Appendix C

C. XML POLICY SYNTAX

In MANBoP policies are expressed in XML (eXtensible Markup Language). XML has recently emerged as a widely accepted way of representing and exchanging structured information. Its principal technical strengths are that it has a text-based representation, which imposes few restrictions on network technology or protocols and that, through the use of XML Schemas, it has sufficiently strict syntax to permit automated validation and processing information in an unambiguous way.

XML allows the definition of new markup tags and constraints on the relationship between them through the use of templates. XML Schema allows more control over the way XML documents are specified. Datatypes are supported and there is the ability to specify relationships and constraints between different elements of a document.

The structure of XML documents is dictated by the XML Schema against which that documents are validated. Hence, hereafter we will describe the main aspects of the XML Schema defined for MANBoP. We will just describe those functional domain independent elements. Then, the appendix is concluded with an example of a MANBoP policy in XML fulfilling the described XML Schema.

In MANBoP all XML policies are structured around seven mandatory elements and two optional ones. Table C - 1 below shows the section of the XML Schema defining these elements.

MANBoP policy structure

```
<xsd:complexType name="PolicyRuleType">
      <xsd:sequence>
        <xsd:element name="PRN" type="PRNType"/>
        <xsd:element name="PR" type="PRType"/>
        <xsd:element name="UI" type="UIType"/>

        <xsd:element name="PG" type="PGType" minOccurs="0"/>
        <xsd:element name="Evaluation" type="EvalType"/>
        <xsd:element name="ActEnf" type="ActEType"/>
        <xsd:element name="Cond" type="CRType" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element name="Act" type="ARType" minOccurs="1" maxOccurs="unbounded"/>
      </xsd:sequence>
 </xsd:complexType>
```

Table C - 1. Main policy structure

The first of the seven mandatory fields is the *PRN* element. This element contains information to uniquely identify the policy, as the policy name and a sequence number and the identifier of the component that sent the policy if any.

The *PR* element lists the device roles to which the policy applies. That is, all network elements developing a role listed in the policy is expected to respond to it.

The next field of the policy is the *UI* element. It contains the identifier and password of the user that is introducing the policy in the framework. This identifier is used to select the restricted XML Schema against which the user policy should be validated with.

The policy expiration date is contained within the *PRVP* element. Usually the expiration date is just given with the day and hour the policy starts and finishes being valid. Nevertheless, filters specifying concrete months, days and hours during which the policy is not valid can be also introduced. In the table below we can see the structure of the *PRVP* element, from which the only mandatory element is the *TP* element that includes the start and stop times.

Policy Rule Validity Period specification
<xsd:complexType name="PRVPType"> <xsd:sequence> <xsd:element name="TP" type="xsd:string"/> <xsd:element name="MonthOfYearMask" type="MoYMT" minOccurs="0"/> <xsd:element name="DayOfMonthMask" type="DoMMT" minOccurs="0"/> <xsd:element name="DayOfWeekMask" type="DoWMT" minOccurs="0"/> <xsd:element name="TimeOfDay" type="xsd:string" minOccurs="0"/> <xsd:element name="LocalOrUtcTime" type="LoUTType" minOccurs="0"/> </xsd:sequence> </xsd:complexType>

Table C - 2. PRVP element structure

The *PG* element is used for the correct processing of policy groups. This element is optional. It is just included if the policy pertains to a group. The information contained is shown in the table below.

First, the element is defined with the *PGNum* attribute that contains a unique identifier of the policy group among those introduced by the same user. Then, the *NOfP* element includes the number of policies forming the group. The *where* element specifies at what management level the group must be processed, network or element. The *Pos* element contains the position of the policy within the group. Finally, the two last elements are the *NoS* and *ON* elements that are used to correctly process the policy group even if a group policy has been splitted in many policies during the translation process at higher layers. These fields contain respectively the number of splits suffered by the policy at higher layers and the position of the policy among those potential splits.

Policy Group specification

```
<xsd:complexType name="PGType">
   <xsd:sequence>
      <xsd:element name="NOfP" type="xsd:integer"/>
      <xsd:element name="GES" type="xsd:integer"/>
      <xsd:element name="where" type="xsd:integer"/>
      <xsd:element name="Pos" type="xsd:integer"/>
      <xsd:element name="NoS" type="xsd:integer"/>
      <xsd:element name="ON" type="OrderType" minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
   <xsd:attribute name="PGNum" type="xsd:integer"/>
</xsd:complexType>
```

Table C - 3. PG element structure

The two MANBoP policy elements that follow *Evaluation* and *ActEnf* contain information for the correct evaluation of policy conditions and the correct enforcement of policy actions respectively.

More specifically, the *Evaluation* element consists of two elements. These elements indicate respectively, whether the policy conditions are specified using the Conjunctive Normal Form or the Disjunctive Normal Form and at what management level must be the conditions evaluated. The possible values for this last field are network-level, element-level or both.

The *ActEnf* element contains four elements. Its structure is shown in the table below. The *ActM* field specifies the objective of the policy action. Its possible values are creation, activation, modification, deactivation or removal. The *EnfT* element indicates how policy actions must be enforced on the target nodes, either in a best effort way or in a guaranteed way (if the policy is not correctly enforced in ALL target nodes the enforcement is not considered successful and hence the policy is uninstalled from all nodes where it was enforced). Then, the *MS* element establishes if the target of the policy action is the management station itself or managed devices. Finally, the *TN* element lists all managed device identifiers where the policy must be enforced.

Action enforcement specification

```
<xsd:complexType name="ActEType">
   <xsd:sequence>
      <xsd:element name="ActM" type="ActMType"/>
      <xsd:element name="EnfT" type="xsd:integer"/>
      <xsd:element name="MS" type="xsd:boolean"/>
      <xsd:element name="TN" type="TNType"/>
   </xsd:sequence>
</xsd:complexType>
```

Table C - 4. ActEnf element structure

The *Cond* policy element includes all policy conditions. Conditions can be either compound or simple and refer to an hour of the day, an IP flow, a concrete notification or a managed device status. The components needed to monitor these conditions, if any, are also extracted from the *Cond* element

information. This element is optional; when not included, the framework interprets that the policy action should be enforced directly. In the Table C - 5 below we can see how MANBoP policy conditions are structured in XML. There is an abstract policy condition from which both compound and simple conditions inherit. This element defines one mandatory element, the *PCN* element that contains the policy condition name.

Compound conditions are defined in the *CFCondType* type. The elements included are the *IsMirrored*, *CRef* and *MMIds*. *IsMirrored* indicates if the filter mirroring the one specified with the condition should be considering as meeting the condition. *CRef* contains the compound and simple conditions that form this compound condition. Two attributes are defined in this element that indicate respectively, the group number of the condition and if the condition is negated, that is if the policy should be enforced when this condition is not met. Finally, the *MMIds* element lists the identifiers of Monitoring Meter components that must be used to monitor this compound condition.

Simple conditions elements have been defined in XML in a domain-independent way, only its values will be domain-dependent. Simple conditions are specified in the *SPCondType* type. This type includes three attributes that indicate respectively, the identifier of the Monitoring Meter component that must monitor this simple condition, the target nodes that must be monitored and whether all target nodes must met the condition to trigger the policy enforcement or just one is enough. The elements included within the simple condition are *DataName, DataType, EMethod* and *Value*. *DataName* contains the name of the data that must be monitored while the *DataType* element contains the type of this data. The *EMethod* element establishes how must be the condition evaluated to decide when the policy must be enforced, either the monitored data must match the value introduced, or it must be bigger or lower, etc. Finally, the *Value* element contains all information needed to evaluate correctly the policy condition, as the threshold that must be reached or matched.

Policy conditions specification

```
<xsd:complexType name="PCondType" abstract="true">
   <xsd:sequence>
      <xsd:element name="PCN" type="xsd:string"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CFCondType">
   <xsd:complexContent>
      <xsd:extension base="PCondType">
         <xsd:sequence>
            <xsd:element name="IsMirrored" type="xsd:boolean"/>
            <xsd:element name="CRef" minOccurs="0" maxOccurs="unbounded">
               <xsd:complexType>
                  <xsd:sequence>
                     <xsd:element name="PCond" type="PCondType"/>
                  </xsd:sequence>
                  <xsd:attribute name="GN" type="xsd:decimal"/>
                  <xsd:attribute name="CN" type="xsd:boolean"/>
               </xsd:complexType>
            </xsd:element>
         </xsd:sequence>
         <xsd:attribute name="MMIds" type="StringList"/>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="SPCondType">
   <xsd:complexContent>
      <xsd:extension base="PCondType">
         <xsd:sequence>
            <xsd:element name="DataName" type="xsd:string"/>
            <xsd:element name="DataType" type="xsd:string"/>
            <xsd:element name="EMethod" type="xsd:string"/>
            <xsd:element name="Value" type="StringList"/>
         </xsd:sequence>
         <xsd:attribute name="MMId" type="xsd:string"/>
         <xsd:attribute name="TN" type="TNType"/>
         <xsd:attribute name="All" type="xsd:boolean"/>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
```

Table C - 5. Cond element structure

The *Act* element contains the action type and parameters as well as information about the component responsible of enforcing this action. At least one *Act* element is mandatory in all policies but there can be more that one. The *Act* element is the only element of the XML policy that is domain-dependant. That is, each XML Schema representing a particular functional domain will contain the policy actions with their corresponding elements specific for that domain. In the table below we can see an example of how domain-dependant actions are specified in the XML Schema.

As with the *Cond* element, there is an abstract type from which all policy actions defined in the XML Schema must inherit. This abstract type defines one mandatory element and one mandatory attribute. The element contains

the name of the policy action while the attribute includes the identifier of the Policy Consumer component that must be used to enforce the policy.

Domain-specific actions are defined by inheriting from the abstract type. Table C - 6 below shows an example of domain-specific policy action defining five elements. As these elements are domain-specific we will not describe them here. More information about the meaning of these elements can be found in the implementation chapter.

Policy actions specification

```
<xsd:complexType name="PActType" abstract="true">
    <xsd:sequence>
       <xsd:element name="PActName" type="xsd:string"/>
    </xsd:sequence>
    <xsd:attribute name="PCId" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="QoSAlloc">
    <xsd:complexContent>
       <xsd:extension base="PActType">
          <xsd:sequence>
                <xsd:element name="VNId" type="xsd:string"/>
                <xsd:element name="QoSClass">
                      <xsd:simpleType>
                         <xsd:restriction base="xsd:string">
                            <xsd:enumeration value="gold"/>
                            <xsd:enumeration value="silver"/>
                            <xsd:enumeration value="bronze"/>
                         </xsd:restriction>
                      </xsd:simpleType>
                </xsd:element>
                <xsd:element name="TrafficProfile" type="xsd:integer" minOccurs="0"/>

                      <xsd:simpleType>
                         <xsd:restriction base="xsd:string">
                            <xsd:enumeration value="gold"/>
                            <xsd:enumeration value="silver"/>
                            <xsd:enumeration value="bronze"/>
                         </xsd:restriction>
                      </xsd:simpleType>
                </xsd:element>
                <xsd:element name="EE" type="xsd:string"/>
          </xsd:sequence>
       </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
```

Table C - 6. Act element structure

After having a clear picture of the structure of a MANBoP XML policy we provide in the table below an example. The XML policy shown has been used during the proof-of-concepts implementation scenario. More specifically, it is used to create VEs for a service provider as part of the process of creating his VAN. More information can be found in the implementation chapter.

XML Policy

```
<?xml version="1.0" encoding="UTF-8"?>
<a:PolicyRule xmlns:a="http://Schema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://Schema QoSPC_0.xsd">
    <PRN InstanceNumber="1" HLPCId="QoSPC">NLVANCreationPolicy</PRN>
    <PR>AA</PR>
    <UI>wtv wtvPass</UI>
    <PRVP>
        <TP>20020101T080000/20040631T120000</TP>
    </PRVP>
    <PG PGNum="1">
        <NOfP>4</NOfP>
        <GES>3</GES>
        <where>0</where>
        <Pos>1</Pos>
        <NoS>0</NoS>
    </PG>
    <Evaluation>
        <CL>2</CL>
        <where>1</where>
    </Evaluation>
    <ActEnf>
        <ActM>0</ActM>
        <EnfT>0</EnfT>
        <MS>false</MS>
        <TN>147.83.106.104 10.0.4.4</TN>
    </ActEnf>
    <Cond>
        <CRef GN="1" CN="false">
            <PCond xsi:type="a:CFCondType" MMIds="">
                <PCN>VANSitesInfo</PCN>
                <IsMirrored>false</IsMirrored>
                <CRef GN="1" CN="false">
                    <PCond xsi:type="a:SPCondType" MMId="null" TN="" All="true">
                        <PCN>VANSiteInfo</PCN>
                        <DataName>IPAddr</DataName>
                        <DataType>IPv4Addr</DataType>
                        <EMethod>Match</EMethod>
                        <Value>147.83.106.111</Value>
                    </PCond>
                </CRef>
                <CRef GN="1" CN="false">
                    <PCond xsi:type="a:SPCondType" MMId="null" TN="" All="true">
                        <PCN>VANSiteInfo</PCN>
                        <DataName>IPAddr</DataName>
                        <DataType>IPv4Addr</DataType>
                        <EMethod>Match</EMethod>
                        <Value>172.31.255.3</Value>
                    </PCond>
                </CRef>
            </PCond>
        </CRef>
    </Cond>
    <Act>
        <PAct xsi:type="a:QoSAlloc" PCId="QoSPC">
            <PActName>NLAlloc</PActName>
            <VNId>wtv</VNId>
            <QoSClass>gold</QoSClass>
            <CompQoSClass>silver</CompQoSClass>
            <EE>JVM</EE>
        </PAct>
  </Act>
</a:PolicyRule>
```

Table C - 7. XML policy example