Business To Manufacturing Markup Language

B2MML - Extensions

Version 0500 – March 2011

Extension Documentation

Table of Contents

Change History:

Change	Date	Person	Description
V03	26 Aug 2005	Dennis Brandl Dave Emerson	• Initial version to explain the extension methods. V0300 added substitution groups. One group added just before each Any element.
V04	04 June 2007	Dennis Brandl	• Added Transaction elements
V0401	Oct 2008	Dennis Brandl	• Added chapter 5 to show a custom schema
V0500	Mar 2011	Dennis Brandl	• Updated version for ISA 95.02-2010 changes • Removed the ##any extensions, except in back supported Production schemas

1 Schema Scope

This document defines the information that is used in the B2MML schemas to provide user defined extensions to the schemas.

This information is based on the data models and attributes defined in the ANSI/ISA 95.00.02 Enterprise/Control System Integration standard. Contact ISA (The Instrumentation, System, and Automation Society) for copies of the standard. Additional information on the standard is available at www.isa.org.

2 Extension Methods

The B2MML schemas follow the ANSI/ISA 95 standard in user extensions. There are four methods for user extensions which are defined within the standard itself;

1. properties
2. segments
3. process and product parameters
4. user extended enumerations

Each method is described below, even though they are not really schema extensions. These four methods are the recommended method for the addition of project or company specific information.

However, there are situations where the four methods will not provide the required flexibility, readability, or validation. Therefore in order to make the schemas more useful, selected elements include the ability for the elements to be extended. The two methods for extending schemas are defined in this document, but the user defined extended elements are not defined here and should not be considered understandable between applications without prior agreement.

The recommended extension mechanisms are listed below:

1. Properties **J J J**
- Define properties where available to avoid requiring schema extensions

2. Process and Product Parameters **J J J**
- Define parameters for process segment specific information or for product specific information

3. Segments **J J J**
- Define process segments to represent activity sets for business reasons

4. User Enumerations **J J**
- Define project or company specific extensions to standard name lists
- No way to formally document user enumerations within a single schema

5. Substitution groups **J J**
- New extension method in v0300
- Allows for XML validation of extended schemas
- Recommended method for schema extensions

6. Any element **J**
- Original B2MML extension method
- Validation turned off in v0200 in favor of substitution groups
- Removed in V0500 except in Production schemas

7. Customize base schemas **L L L**
- Changing core schemas is possible, the license allows any modification
- However, it makes interoperability impossible
- Not recommended

There are two types of schema extension methods defined (5 & 6), the first method uses substitution groups and provides a method for extension that allows for validation of the extensions against a schema. It is the recommended method to be used, if extensions are absolutely required.

The second extension method uses the XSD ##any definition and does not allow validation of the extensions against a schema. This method is defined for those cases where validation of the extensions is not required or possible.

3 Extensions with no Schema Changes

3.1 Properties

ANSI/ISA 95 defines "properties" as the standard method to add project or company specific information. Each project, or company, may define the properties that are important to their information exchange. Properties are the method to define the specific attributes of exchanged information about personnel, equipment, and material.

Properties are defined by either the end user (specifying what information they want to exchange), or by the ERP or MES vendor (specifying what information they export and import), or by a combination of both

> For example: Important information may be the equipment status (clean, sterile, not available) and equipment location (production, clean room, warehouse). An *equipment* B2MML document may define the current value of the status and current equipment location. A *production schedule* B2MMLdocument may specify that equipment to be used has a certain status (sterile). A *production performance* B2MML document may specify the activity where the status and location changed.
>
> For example: Important information may be the Octane level of a "Fuel" material class. In this case there may be multiple properties defined:

```
<MaterialClass>
    <ID "Fuel" />
    <MaterialClassProperty>    <ID "Octane.Target" />      </MaterialClassProperty>
    <MaterialClassProperty>    <ID "Octane.LowLimit" />    </MaterialClassProperty>
    <MaterialClassProperty>    <ID "Octane.HighLimit" />   </MaterialClassProperty>
</MaterialClassProperty>
```

> A *production schedule* could then specify the target, high limit, and low limit for the fuel to be produced.
>
> NOTE: There are no B2MML restrictions on property names, but there may be restrictions on sending or receiving systems. The example above shows the use of a hierarchical name space using a "." as the element separator.

You can use the class elements (personnel class, equipment class, material class, and material definition) to define the exchanged properties. Property definitions don't have to be dynamically exchanged, they can be statically defined or entered as configuration information. However B2MML documents are a good way to document what definitions have been agreed upon.

> For example: If exchanging an equipment "Status" property, then the equipment class should have a "Status" property.

However, properties are only defined for personnel, equipment, and material objects. Therefore other user extension methods have to be used for other information.

3.2 Process and Product Parameters

When information on specific segments of production must be exchanged, but the information is not directly related to personnel, material, or equipment, then process and product parameters may be used. Process and product parameters provide a simple way to exchange information in *production schedules* and *production performance* without relating it to a resource.

> For example: A process parameter may be the results on environmental tests that were run (room temperature, humidity, last cleaned date, number of people entering during production).

Process parameters are elements of data that are related to a process segment. The allowable process parameters may be defined in a *process segment* B2MML document.

Product parameters are elements of data that are related to a product segment for a specific product. The allowable product parameters may be defined in a *product definition* B2MML document.

> For example: A product parameter may be the color to paint a part during a paint segment, or the tests to run on a part during an inspection segment.

Parameter definitions don't have to be dynamically exchanged. However *ProcessSegment* B2MML documents are a good way to document what product independent parameters have been agreed upon, and *ProductDefinition* B2MML documents are a good way to document what product specific parameters have been agreed upon.

> NOTE: Parameter types are recursive (a parameter may contain a parameter), and each parameter may have multiple values (such as for an array or a set of named elements). This flexibility allows parameters to represent very complex elements of exchanged data (structures, arrays, …)

There are no differences between process parameters and product parameters in production schedules and production responses.

- A *ProductionSchedule* may define the values for parameters in *ProductionParameters* elements in *SegmentReequirementTypes*.
- A *ProductionPerformance* may define values for parameters in *ProductionData* elements in *SegmentResponseTypes*.

3.3 Segments

Process segments are used to identify activities that are visible to the business. These do not have to directly correspond to production activities, but may correspond to summaries of activities, inspection, daily report, shift report, etc …

Instead of adding elements under a segment, such as adding an "InspectionData" element under a *ProductionPerformance – SegmentResponse"*, a new "Inspection" segment can be defined with specific process segment parameters.

> For example: The following B2MML segment defines a process segment and multiple parameters.

```
<ProcessSegment>
    <ID "Inspection" />
    <Parameter>  <ID "InspectionLotNumber" />     </Parameter>
    <Parameter>  <ID "NumberOfInspections" />     </Parameter>
</ProcessSegment>
```

> The following B2MML segment defines a Production Response with parameter values for the "Inspection" segment parameters.

```
<ProductionResponse>
    <ID “2005-07-04AAA” />
    <SegmentResponse>
        <ID “SR5525" />
        <ProcessSegmentID “Inspection” />
        <ProductionData>
            <ID “InspectionLotNumber" />
            <Value “2005-07-04AAA123” />
        </ProductionData>
        <ProductionData>
            <ID “NumberOfInspections" />
            <Value “14” />
        </ProductionData>
</ProductionResponse >
```

3.4 User Enumeration Extensibility

The ANSI/ISA-95.00.02 Enterprise/Control System Integration standard Part 2 defines sets of specific enumerations. B2MML provides support for these enumerations and possible user extensions extension. The extended enumeration values are not defined in this standard and should not be considered understandable between applications without prior agreement.

All types that contain enumerated values have an enumeration value of “Other” and an attribute of “OtherValue”. Any element, in an instance document, with an extended enumeration should use the enumeration value of “Other” and place the actual enumeration in the “OtherValue” attribute.

The enumerations and “OtherValue” attribute are added in two steps in the schemas. This is done due to a restriction in W3C schemas that prevent restrictions (enumeration values) and extensions (adding an attribute) at the same time. The complex type naming convention used is a “1” in the name of temporary complex type (complex type name = ‘element name’ + ‘1’ + ‘Type’) and the same name without the ‘1’ for the final complex type name. The two step process can be ignored by XML authors because the temporary type is not intended for use in XML documents.

Schema definition:

```
  <xsd:simpleType name = "CapabilityType1Type">
      <xsd:restriction base = "xsd:string">
        <xsd:enumeration value = "Committed" />
        <xsd:enumeration value = "Available" />
        <xsd:enumeration value = "Unattainable" />
        <xsd:enumeration value = "Other" />
      </xsd:restriction>
  </xsd:simpleType>

  <xsd:complexType name = "CapabilityTypeType">
    <xsd:simpleContent>
      <xsd:extension base = "CapabilityType1Type">
        <xsd:attribute name = "OtherValue" type = "xsd:string"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
```

Used in XML document:

```
<b2mml:CapabilityType>
 Committed
</b2mml: CapabilityType >


<b2mml: CapabilityType OtherValue = "MayBePurchased">
 Other
</b2mml: CapabilityType >
```

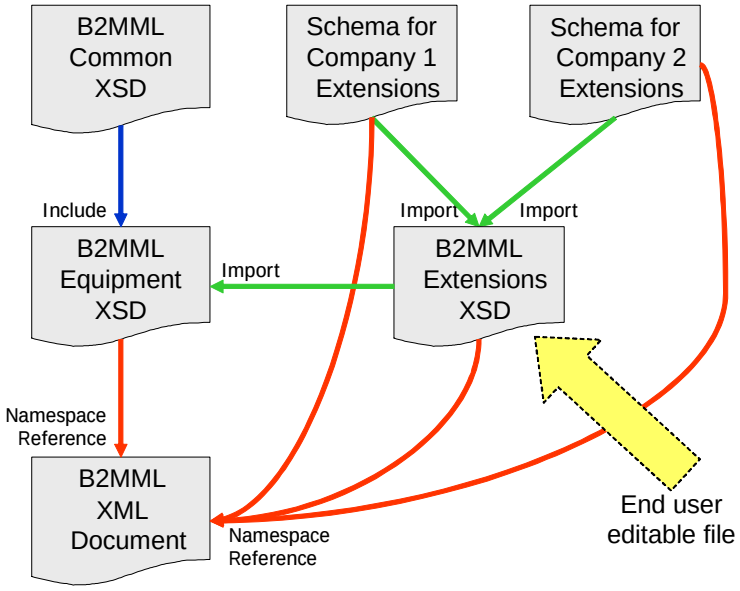
4 Extension using Schema Changes

4.1 Extended Namespace

When none of the previous extension methods will meet the requirements, then an extension mechanism using an end user managed schema file has been defined. The extension mechanism provides for fully validated B2MML documents.

Each complex type that is not an enumerated set contains an XSD:GROUP reference. The reference is to the element name in the "Extended" name space. The extended namespace is defined in the user editable b2mml-v0300-extensions.xsd file. The user extensions may be the specific schema extensions, or the extensions file could import company specific extensions.

The figure below shows how "Imports" and "Includes" are used in managing the extended name spaces.



Basically the end user may edit the b2mml-v0401-extensions.xsd file to contain the added elements. It may include company specific extensions or vendor supplied extensions.

For example: The following schema segment defines the "Extended:MaterialProducedRequirement" element within a "MaterialProducedRequirementType".

```
<xsd:schema    targetNamespace    = "http://www.wbf.org/xml/b2mml-v0300"
               xmlns              = "http://www.wbf.org/xml/b2mml-v0300"
               xmlns:Extended     = "http://www.wbf.org/xml/b2mml-v0300-extensions"
               xmlns:xsd      = "http://www.w3.org/2001/XMLSchema"
               elementFormDefault = "qualified"
               attributeFormDefault   = "unqualified">

<xsd:complexType name = "MaterialProducedRequirementType">
  <xsd:sequence>
    <xsd:element name = "MaterialClassID"type = "MaterialClassIDType"
                                    minOccurs = "0" maxOccurs = "unbounded"/>
    <xsd:element name = "MaterialDefinitionID"   type = "MaterialDefinitionIDType"
                                    minOccurs = "0"/>
    <xsd:element name = "MaterialLotID"      type = "MaterialLotIDType"
                                    minOccurs = "0"/>
    <xsd:element name = "MaterialSubLotID"   type = "MaterialSubLotIDType"
                                    minOccurs = "0"/>
    <xsd:element name = "Description"        type = "DescriptionType"
                                    minOccurs = "0" maxOccurs = "unbounded"/>
    <xsd:element name = "Location"    type = "LocationType"
                                    minOccurs = "0"/>
    <xsd:element name = "Quantity"    type = "QuantityType"
                                    minOccurs = "0" maxOccurs = "unbounded"/>
    <xsd:element name = "MaterialProducedRequirementProperty"
                                    type = "MaterialProducedRequirementPropertyType"
                                    minOccurs = "0" maxOccurs = "unbounded"/>
    <xsd:element name = "RequiredByRequestedSegmentResponse"
                                   type = "RequiredByRequestedSegmentResponseType"
                                   minOccurs = "0"/>
    <xsd:group   ref  = "Extended:MaterialProducedRequirement" minOccurs="0" maxOccurs="1"/>
    <xsd:element name = "Any"         type="AnyType"
                                    minOccurs = "0" maxOccurs = "unbounded"/>
 </xsd:sequence>
</xsd:complexType>
```

The following schema segment defines a sample user extension to MaterialProducedRequirement and includes two company specific extensions:

```
<xsd:group name="MaterialProducedRequirement">
    <xsd:sequence>
        <xsd:element name="ByProduct"            type="xsd:string" minOccurs="0"/>
        <xsd:element name="ReservationNumber"    type="xsd:string" minOccurs="0"/>
        <xsd:element name="ReservationItem"      type="xsd:string" minOccurs="0"/>
        <xsd:group ref="AAA-Control:SpecialKey"  minOccurs="0" maxOccurs="1"/>
        <xsd:group ref="XYZ-Eng:SpecialKey"      minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
</xsd:group>
```

NOTE: When company specific extensions are added to an extended element, then the extended names spaces should be added in alphabetical order with a minOccurs of 0, and all end users extended elements should be listed first. This allows the mixing of schemas from different vendors, so that any B2MML document may have none, some, or all of the extensions without any element sequencing problems.

The use of the extended namespace construct allows for intra-vendor or intra-company extensions to the schemas with full schema validation, extending the number of places that they may be applied. The extended namespace does this without effecting the ability to perform inter-vendor or inter-company information exchange as defined by the standards for exchanged information in ANSI/ISA-95.

ANSI/ISA-95 Part 2 includes a list of objects and attributes, mapped as elements in the schemas, which can be used to measure the **completeness** of an implementation. Excessive use of extended namespace constructs combined with minimal coverage of supported objects will result in a minimal measure of completeness of an implementation.

4.2 ##any Extensions

Each complex type in the Production schemas (ProductionSchedule, ProductionPerformance, ProductionCapability, Productdefinition) that is not an enumeration set contains the XSD element Any, which contains the element ##any. This allows any user-defined element to be included at the end of the element sequence. The additional elements must be defined in a schema whose namespace is referenced in order to create a valid XML document.

Schema definition:

```
<xsd:complexType name="AnyType" >
   <xsd:sequence>
      <xsd:any namespace="##any" processContents="skip"
                        minOccurs = "0" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name = "PersonnelCapabilityPropertyType">
  <xsd:sequence>
   <xsd:element name = "ID"          type = "IdentifierType" />
   <xsd:element name = "Description" type = "DescriptionType"
                                     minOccurs = "0" maxOccurs = "unbounded"/>
   <xsd:element name = "Value"       type = "ValueType"
                                     minOccurs = "0" maxOccurs = "unbounded"/>
   <xsd:element name = "Quantity"    type = "QuantityValueType"
                                     minOccurs = "0" maxOccurs = "unbounded"/>
   <xsd:group   ref  = "Extended:PersonnelCapabilityProperty"
                                      minOccurs = "0" maxOccurs = "1"/>
<!-- Added in V04 - The Any element should not be used in new schemas.  This extension
     schema should be used to add user defined elements to B2MML documents.
     In a future release the Any element will be removed from B2MML.  -->
   <xsd:element name = "Any"          type="AnyType"
                                      minOccurs = "0" maxOccurs = "unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
```

Used in XML document:

```
<PersonnelCapabilityPropertyType>
    <ID>XYZ Weigh Scale Certified</ID>
    <Description>”Training in use and certified to use the XYZ Weigh scale”</Description>
    <Any>
          <mySchema:CD>Class Date</mySchema:CD>
          <mySchema:CG>Class Grade</mySchema:CG>
    </Any>
</ PersonnelCapabilityPropertyType >
```

The use of the ##any construct allows for intra-vendor or intra-company extensions to the schemas, extending the number of places that they may be applied. However the extensions can not be validated with the standard “AnyType” definition. The ##any does this without effecting the ability to perform inter-vendor or inter-company information exchange as defined by the standards for exchanged information in ANSI/ISA-95.

ANSI/ISA-95 Part 2 includes a list of objects and attributes, mapped as elements in the schemas, which can be used to measure the **completeness** of an implementation. Excessive use of ##any constructs combined with minimal coverage of supported objects will result in a minimal measure of completeness of an implementation.

Many of the transaction elements contain ##any constructs for compatibility with OAGiS transaction rules. The use if the ##any constructs in the transaction elements is not recommended in B2MML.

5 Example of a Custom Schema

The following schema is an example custom schema developed using the infrastructure of the WBF. It illustrates how the UN/CEFAC core components, the WBF common and extension schemas can be used in company or project specific schemas.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- This is a DRAFT    -->
<xsd:schema
     xmlns:xsd           ="http://www.w3.org/2001/XMLSchema"
     xmlns:b2mml         ="http://www.wbf.org/xml/B2MML-V0401"
     xmlns               ="http://www.wbf.org/xml/B2MML-V0401-Extensions"
     targetNamespace     ="http://www.wbf.org/xml/B2MML-V0401-Extensions"
     elementFormDefault  ="qualified"
     attributeFormDefault="unqualified">

     <!-- Import the Common schema   -->
     <xsd:import namespace="http://www.wbf.org/xml/B2MML-V0401" schemaLocation="B2MML-V0401-Common.xsd"/>

     <!-- Include the Extension Schema         -->
     <xsd:include schemaLocation="B2MML-V0401-Extensions.xsd"/>


     <xsd:annotation>
          <xsd:documentation>

        This custom schema is developed within the infrastructure of the WBF Work
        (including specifications, documents, software, and related items)
        referred to as the Business To Manufacturing Markup Language (B2MML).

        It belongs to the entire responsibility of the user to write consistent code
        in this document in order to provide a seamless extension to the original
        B2MML structure, matching the intended specific integration needs.

        This particular schema provide a mean for exchanging information about
        Engineering Change Management.
        If used in another context of its original use in a specific custom application,
        the user must make sure that the global elements names added to the generic
        "http://www.wbf.org/xml/B2MML-V0401-Extensions" namespace do not conflict
        with existing user extensions of the target application.

        "The Business To Manufacturing Markup Language (B2MML) is used
        courtesy of the WBF."

   </xsd:documentation>
          <xsd:documentation>

      Revision History

        Ver     Date            Person          Note
        ---     ----            ------          ----
        v01   27 June 2007     J. Vieille

   </xsd:documentation>
     </xsd:annotation>
     <!-- Global Elements -->
     <xsd:element name="EngineeringChangeOrderInformation">
          <xsd:complexType>
               <xsd:sequence>
                    <xsd:element name="ID" type="b2mml:IdentifierType" minOccurs="0"/>
                    <xsd:element name="Description" type="b2mml:DescriptionType" minOccurs="0" maxOccurs="unbounded"/>
                    <xsd:element name="Location" type="b2mml:LocationType" minOccurs="0"/>
                    <xsd:element name="PublishedDate" type="b2mml:PublishedDateType" minOccurs="0"/>
                    <xsd:element name="EngineeringChangeOrder" minOccurs="0" maxOccurs="unbounded">
                         <xsd:complexType>
                              <xsd:sequence>
                                   <xsd:element name="ID" type="b2mml:IdentifierType" minOccurs="0"/>
                                   <xsd:element name="Description"
                                                  type="b2mml:DescriptionType" minOccurs="0" maxOccurs="unbounded"/>
                                   <xsd:element name="EngineeringChangeOrderState"
                                                  type="EngineeringChangeOrderStateType" minOccurs="0"/>
                                   <xsd:element name="EngineeringChangeOrderReason" type="b2mml:ReasonType" minOccurs="0"/>
                                   <xsd:element name="Location" type="b2mml:LocationType" minOccurs="0"/>
                                   <xsd:element name="Priority" type="b2mml:PriorityType" minOccurs="0"/>
                                   <xsd:element name="ApprovalDateTime" type="b2mml:DateTimeType" minOccurs="0"/>
                                   <xsd:element name="ImplementationDateTime" type="b2mml:DateTimeType" minOccurs="0"/>
                                   <xsd:element name="ApprovalRequiredBy" type="b2mml:DateTimeType" minOccurs="0"/>
                                   <xsd:element name="Requester" type="b2mml:IdentifierType" minOccurs="0"/>
                                   <xsd:element name="EngineeringChangeDetail" minOccurs="0" maxOccurs="unbounded">
                                             <xsd:complexType>
                                                  <xsd:sequence>
                                                       <xsd:element name="Description"
                                                            type="b2mml:DescriptionType" minOccurs="0" maxOccurs="unbounded"/>
                                                       <xsd:element name="ObjectType"
                                                            type="b2mml:IdentifierType" minOccurs="0"/>
```

```
                                                    <xsd:element name="ObjectTypeActiveForChangeNumber"
                                                         type="b2mml:IdentifierType" minOccurs="0"/>
                                                    <xsd:element name="ManagementRecordRequiredForObjectIndicator"
                                                         type="b2mml:IndicatorType" minOccurs="0"/>
                                                    <xsd:element name="ObjectManagementRecordGeneratedIndicator"
                                                         type="b2mml:IndicatorType" minOccurs="0"/>
                                                    <xsd:element name="ObjectTypeLockedForChangesIndicator"
                                                         type="b2mml:IndicatorType" minOccurs="0"/>
                                               </xsd:sequence>
                                          </xsd:complexType>
                                     </xsd:element>
                                </xsd:sequence>
                           </xsd:complexType>
                      </xsd:element>
                 </xsd:sequence>
            </xsd:complexType>
       </xsd:element>
       <xsd:complexType name="TypeOfChangeType">
            <xsd:simpleContent>
                 <xsd:extension base="b2mml:CodeType"/>
            </xsd:simpleContent>
       </xsd:complexType>
       <xsd:complexType name="EngineeringChangeOrderStateType">
            <xsd:simpleContent>
                 <xsd:extension base="b2mml:CodeType"/>
            </xsd:simpleContent>
       </xsd:complexType>
</xsd:schema>
```