Business to Manufacturing Markup Language

B2MML V0700 Documentation

Version 0700 - August 2020

B2MML-BatchML Doc

Table of Contents

1 DOCUMENTATION SCOPE

This document defines the overall structure and use of XML to define the XML implementations of the ISA-95, IEC 62264, ISA-88, and IEC 61412 standards.

This information is based on the data models and attributes defined in the ANSI/ISA-95 Enterprise/Control System Integration standard. Contact ISA (The International Society of Automation) for copies of the standard. Additional information on the standard is available at www.isa.org.

This information is based on the data models and attributes defined in the ANSI/ISA-88.00.02 Batch Control standard Part 2. Contact ISA (The Instrumentation, Systems, and Automation Society) for copies of the standard. Additional information on the standard is available at www.isa.org.

This document does not define the details of each of the B2MML and BatchML schemas, the details of the meaning of the elements are defined in the associated ISA-95 and ISA-88 standards. The latest release of ISA -95 and B2MML have been coordinated, so that all attribute names used in ISA-95 objects are the same as the element names in B2MML, with spaces removed.

1.1 Key Information Assumptions

The schemas define exchanged information and do not define the use of the information or encapsulation of the information in any defining transactions. These schemas are intended to be used to create XML documents used to exchange batch data as well as serve as the basis for corporate, system or application specific schemas that may be derived from the B2MML and BatchML schemas.

1.2 Schema Includes and Imports

The xsd:import and xsd:include structure is carefully constructed to allow for the reuse of the common schemes, the extension schemas, and the UN/CEFACT core component types. Any modifications to the includes and imports may invalidate the schema rules.

Under normal use the only edited schemes should be

- B2MML-CommonExtensions.xsd ,
- B2MML-Extensions.xsd,
- BatchML-GeneralRecipeExtensions.xsd,
- BatchML-BatchInformationExtensions.xsd, and
- BatchML-BatchProductionRecordExtensions.xsd.

1.3 Type Names

The XML schema uses a model that defines data types for each complex element, some simple types, and each defined enumeration. The data types all follow the convention of a suffix of "Type" added to the element name.

The method is the "Venetian Blind Model", defined in the book Professional XML Schemas, 2001, published by WROX (ISBN 1-861005-47-4). It makes all of the type names global and usable in user derived works, without a loss of context or additional information required to identify the element as of being of the same type as related B2MML elements.

Schema definition of a simple type:

```
<xsd:complexType name="DescriptionType">
    <xsd:simpleContent>
        <xsd:restriction base="TextType"/>
    </xsd:simpleContent>
</xsd:complexType>
```

Schema definition of an enumeration type:

```
<xsd:complexType name="Action1Type">
    <xsd:simpleContent>
        <xsd:restriction base="CodeType">
            <xsd:enumeration value="Added"/>
            <xsd:enumeration value="Deleted"/>
            <xsd:enumeration value="Changed"/>
            <xsd:enumeration value="Observed"/>
            <xsd:enumeration value="Other"/>
        </xsd:restriction>
    </xsd:simpleContent>
</xsd:complexType>
<!--     -->
<xsd:complexType name="ActionType">
     <xsd:simpleContent>
        <xsd:extension base="Action1Type">
            <xsd:attribute name="OtherValue" type="xsd:string"/>
        </xsd:extension>
    </xsd:simpleContent>
</xsd:complexType>
```

1.4 User Element Extensibility

In order to make the schemas more useful, selected elements include the ability for elements to be extended. The extended elements are not defined in this standard and should not be considered understandable between applications without prior agreement. The extension mechanism is defined Chapter 4.

1.5 Use of IDs in Schema Definitions

The use of IDs (IdentificationType) in the schema definition is based on the definition of IDs in the ANSI/ISA-95 standard. Many elements in the exchanged information require unique IDs. These IDs should be considered unique only within the scope of the exchanged information. They may have not meaning beyond the scope of exchanged information. In the latest release of ISA-95 and B2MML all objects have IDs, and these IDs can be used to identify elements which may be defined in other exchange messages.

The element IDs are defined only to identify objects within related exchanged information sets. The element IDs are not intended to act as global object IDs or database index attributes.

This will sometimes require translation of the element IDs from one system's internal identification into a standard representation. For example, a unit may be identified as resource "R100011" in the scheduling system and "East Side Reactor" in the manufacturing system. A unique identification set must be agreed to in order to exchange information. Often the IDs may be the IDs of one or the other system in an exchange, assuming that the IDs are unique (within the scope of the exchanged information)

Generally elements that are elements of aggregations, and are not referenced elsewhere in the model, do not require unique IDs.

1.1 ISA-95 to B2MML Mapping

The following table lists the ISA-95 Part, the associated object model, and the B2MML schemas that implement the models.

ISA-95 Part and Object Model	Related B2MML Schema Files
Part 2, Clause 5.1 Hierarchy Scope	B2MML-Common.xsd
Part 2, Clause 5.2 Spatial Definition	B2MML-Common.xsd
Part 2, Clause 5.3 Operational Location Information	B2MML-OperationalLocation.xsd
Part 2, Clause 5.4 Personnel Information	B2MML-Personnel.xsd
Part 2, Clause 5.5 Role Based Equipment Information	B2MML-Equipment.xsd
Part 2, Clause 5.6 Physical Asset Information	B2MML-PhysicalAsset.xsd
Part 2, Clause 5.7 Material Information	B2MML-Material.xsd
Part 2, Clause 5.8 Process Segment Information	B2MML-ProcessSegment.xsd
Part 2, Clause 5.9 Operations Test Information	B2MML-OperationsTest.xsd
Part 2, Clause 5.10 Operations Record Information	B2MML-Common.xsd
Part 2, Clause 5.11 Operations Event Information	B2MML-OperationsEvent.xsd
Part 2, Clause 6.1 Operations Definition Information	B2MML-OperationsDefinition.xsd

ISA-95 Part and Object Model	Related B2MML Schema Files
Part 2, Clause 6.2 Operations Schedule Information	B2MML-OperationsSchedule.xsd B2MML-OperationsPerformancTypes.xsd
Part 2, Clause 6.3 Operations Performance Information	B2MML-OperationsPerformance.xsd B2MML-OperationsPerformancTypes.xsd
Part 2, Clause 6.4 Operations Capability Information	B2MML-OperationsCapability.xsd
Part 2, Clause 6.5 Process Segment Capability Information	B2MML-OperationsCapability.xsd
Part 2, Clause 6.6 Operations Segment Capability Information	B2MML-OperationsCapability.xsd
Part 4, Clause 5 Resource Relationship Network Information	B2MML-ResourceRelationshipNetwork.xsd
Part 4, Clause 6 Work Definition Information	B2MML-WorkDefinition.xsd B2MML-WorkflowSpecification.xsd
Part 4, Clause 7 Work Schedule Information	B2MML-WorkSchedule.xsd
Part 4, Clause 8 Work Performance Information	B2MML-WorkPerformance.xsd
Part 4, Clause 9 Work Capability Information	B2MML-WorkCapability.xsd
Part 4, Clause 10 Work Master Capability Information	B2MML-WorkCapability.xsd
Part 4, Clause 11 Work KPI Information	See the MESA KPIML documentation. www.mesa.org
Part 4, Clause 12 Work Alert Information	B2MML-WorkAlert.xsd
Part 4, Clause 13 Work Calendar Information	B2MML-WorkCalendar.xsd
Part 4, Clause 15 Work Record Information	B2MML-WorkRecord.xsd
Part 5, Clause 5.24 Transaction Profile	B2MML-TransactionProfile.xsd
TR01 Master Data Profile Template	B2MML-MasterDataProfile.xsd

1.2 Additional Schemas

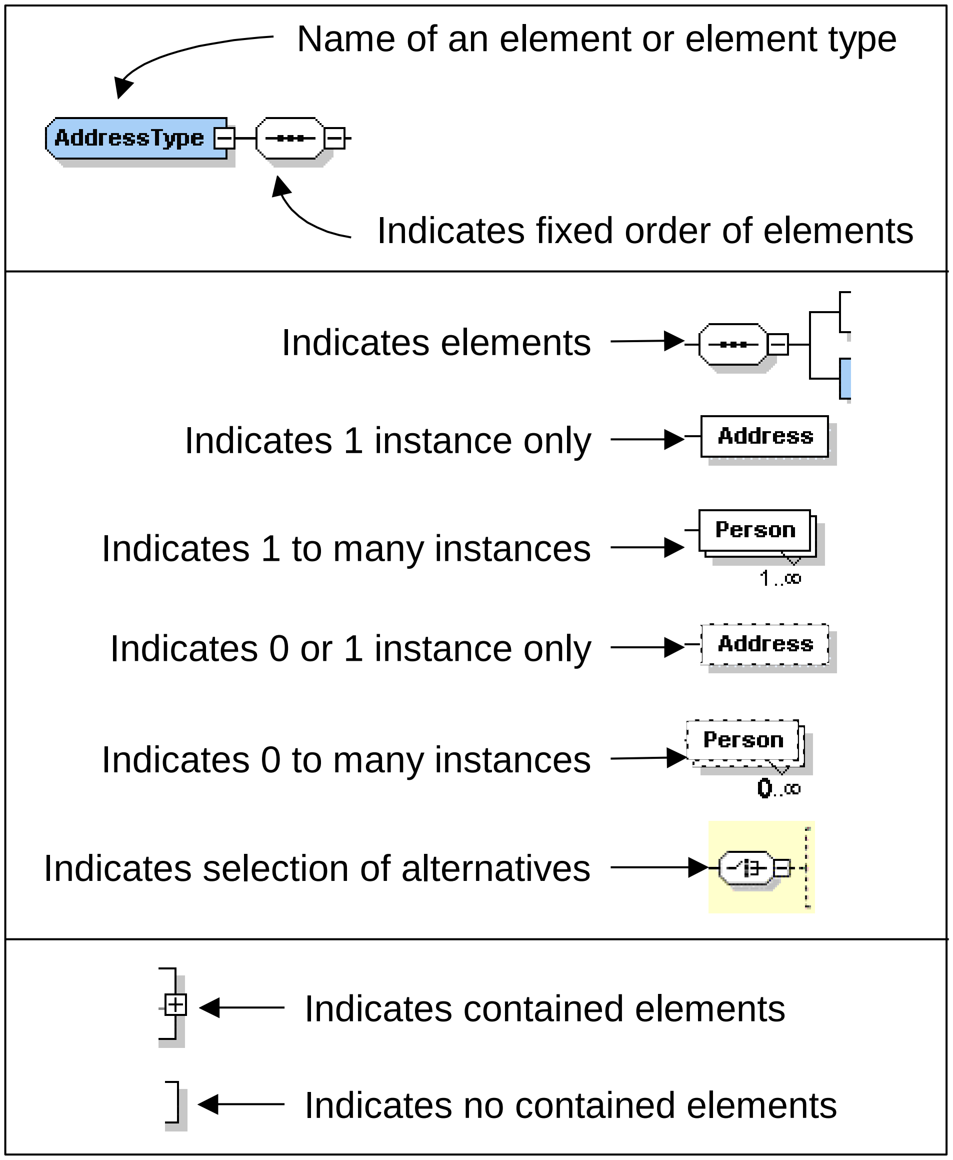
In addition to the schema listed above the following schema are includes with B2MML.

B2MML Schema File	Description
B2MML-AllExtensions.xsd	Includes all of the extension schemas for B2MML and BatchML.
B2MML-Common.xsd	Defines the data types that are used in more than one schema.
B2MML-CommonExtensions.xsd	Defines the extension placeholders for the B2MML-Common data types.
B2MML-ConfirmBOD.xsd	Defines the CONFIRM message to comply with Part 5 transaction rules.
B2MML-CoreComponents.xsd	Defines the UN/CEFAC core components used as base types.
B2MML-ErrorMessage.xsd	Defines a common error response message, using the Part 5 transaction model and an ErrorMessage object type.

B2MML Schema File	Description
B2MML-Extensions.xsd	Defines the extension placeholders for all schema specific complex types that are not defined in the B2MML-CommonExtensions.
B2MML-InformationObject.xsd	Defines a complex type that can contain any B2MML and BatchML type that is used in Work Records, and the Master Data Profile.

1.6 Diagram Convention

The schema diagrams using the following convention to illustrate the structure of the schema elements, the type of the elements and attributes, and the rules for optional elements and repetition.


Name of an element or element type
AddressType
Indicates fixed order of elements
Indicates elements
Indicates 1 instance only
Address
Indicates 1 to many instances
Person
1..∞
Indicates 0 or 1 instance only
Address
Indicates 0 to many instances
Person
0..∞
Indicates selection of alternatives
Indicates contained elements
Indicates no contained elements

2 UN/CEFACT CORE COMPONENT TYPES

The base types for most elements are derived from core component types that are compatible with the UN/CEFACT core component types. The UN/CEFACT core component types are a common set of types that define specific terms with semantic meaning (e.g. the meaning of a quantity, currency, amount, identifier,…). The UN/CEFACT core components were defined in a Core Components Technical Specification (CCTS) developed by the ebXML project now organized by UN/CEFACT and ISO TC 154.

NOTE: The core components contain optional attributes that may be used to specify the context and source of the associated element value. All attributes are optional in B2MML.

The core components use several international standards for the representation of semantic and standardized information:

Name	Standard
Country Code	ISO 3166.1
Region Code	ISO 3166.2
Language Code	ISO 639: 1988
Currency Code	ISO 4217
Date and Time Representation	ISO 8601
Unit Of Measure Code	UN/ECE Recommendation 20
Unit of Transport or Packaging Code	UN/ECE Recommendation 21

The core components are defined in the schema file:

B2MML-CoreComponents.xsd

1.7 AmountType

AmountType is used to define a number of monetary units specified in a currency where the unit of currency is explicit or implied. It is derived from a **decimal**.

Optional Attribute	Base XML Type	Description
currencyID	normalizedString	An identifier specifying the identification of a currency code. Reference UN/ECE Rec 9, using 3-letter alphabetic codes, also available as ISO 4217.
currencyCodeListVersionID	normalizedString	An identifier specifying the version of the currency code. The version of the UN/ECE Rec.9 code list.

1.8 BinaryObjectType

BinaryObjectType is used to define a data types representing graphics, pictures, sound, video, or other forms of data that can be represented as a finite length sequence of binary octets. It is derived from base64Binary.

Optional Attribute	Base XML Type	Description
format	string	The format of the binary content. No identifiers for standard formats are defined.
mimeCode	normalizedString	The mine type of the binary object. See IETF RFC 2045, 2046, and 2047.
encodingCode	normalizedString	Specifies the decoding algorithm of the binary object. See IETF RFC 2045, 2046, and 2047.
characterSetCode	normalizedString	The character set of the binary object if the mime type is text. See IETF RFC 2045, 2046, and 2047.
uri	anyURI	The Uniform Resource Identifier that identifies where the binary object is located.
filename	string	The filename of the binary object. See IETF RFC 2045, 2046, and 2047.

1.9 CodeType

CodeType is used to define a character string that is used to represent an entry from a fixed set of enumerations. It is derived from the type **normalizedString**.

All of the B2MML enumerations are derived from **CodeType**. Also, B2MML elements that are not identifications of objects or other elements are derived from **CodeType**.

Optional Attribute	Base XML Type	Description
listID	normalizedString	An Identifier specifying the identification of a code list that this is registered with at an agency. For example: UN/EDIFACT data element 3055 code list
listAgencyID	normalizedString	An Identifier specifying the agency that maintains one or more lists of codes. For example: UN/EDIFACT.
listAgencyName	string	Text that contains the name of the agency that maintains the list of codes.
listName	string	Text that contains the name of a code list that this is registered with at an agency.
listVersionID	normalizedString	An Identifier specifying the version of the code list.
name	string	Text equivalent of the code content component.
languageID	language	An Identifier specifying the language used in the code name.
listURI	anyURI	The Uniform Resource Identifier (URI) that identifies where the code list is located.
listSchemaURI	anyURI	The Uniform Resource Identifier (URI) that identifies where the code list scheme is located.

1.10 DateTimeType

DateTimeType is used to define a particular point in time together with the relevant supplementary information to identify the timezone information. It is derived from the type **dateTime**. In B2MML this is a specific instance on time using the ISO 8601 CE (Common Era) calendar extended format and abbreviated versions. For example:

yyyy-mm-ddThh:mm:ssZ for UTC as “2002-09-22T13:15:23Z”

Optional Attribute	Base XML Type	Description
format	string	Not needed in B2MML, but maintained for compatibility with OAGiS. A string specifying the format of the date time content, however the format of the format attribute is not defined in UN/CEFACT specification.

1.11 IdentifierType

IdentifierType is used to define a character string to identify and distinguish uniquely, one instance of an object in an identification scheme from all other objects in the same scheme. It is derived from the type **normalizedString**.

All of the B2MML ID types are derived from **IdentifierType**.

Optional Attribute	Base XML Type	Description
schemaID	normalizedString	An Identifier specifying the identification of the identification schema.
schemaName	string	Text that contains the name of the identification scheme.
schemaAgencyID	normalizedString	An Identifier specifying the identification of the agency that maintains the schema.
schemaAgencyName	string	Text containing the identification of the agency that maintains the schema.
schemaVersionID	normalizedString	The version (as an Identifier) of the schema.
schemaDataURI	anyURI	The Uniform Resource Identifier (URI) that identifies where schema data is located.
schemaURI	anyURI	The Uniform Resource Identifier (URI) that identifies where schema is located.

1.12 IndicatorType

IndicatorType is used to define a list of two mutually exclusive Boolean values that express the only possible states of a property.

Example: “**True**” or “**False**”. It is derived from the type **string**.

For B2MML purposes the defined values for indicator type is “**True**” and “**False**”.

Optional Attribute	Base XML Type	Description
format	string	A string specifying whether the indicator is numeric, textual or binary; however the format of the format attribute is not defined in UN/CEFACT specification.

1.13 MeasureType

MeasureType is used to define a numeric value determined by measuring an object along with the specified unit of measure. It is derived form type **decimal**.

Optional Attribute	Base XML Type	Description
unitCode	normalizedString	The type of unit of measure. See UN/ECE Rec 20. and X12 355.
unitCodeListVersionID	normalizedString	The version of the unit of measure code list.

1.14 NameType

NameType is used to define the name of any element that requires a common name. It is derived from the type **string**.

Optional Attribute	Base XML Type	Description
languageID	language	An Identifier specifying the language used in the content component.

1.15 NumericType

NumericType is used to define a numeric value determined by measuring an object along with the specified unit of measure. It is derived from the type **decimal**.

Optional Attribute	Base XML Type	Description
format	string	Specifies if the number is an integer, decimal, real number, or percentage. No standard identifiers defined.

1.16 QuantityType

QuantityType is used to define a counted number of non-monetary units, possibly including fractions. It is derived from the type **decimal**.

Optional Attribute	Base XML Type	Description
unitCode	normalizedString	The unit of the quantity. May use UN/ECE Rec. 20.
unitCodeListID	normalizedString	The identification of the code list for the quantity unit of measure.
unitCodeListAgencyID	normalizedString	The identification of the agency that maintains the quantity unit code list.
unitCodeListAgencyName	string	The name of the agency that maintains the quantity unit code list.

1.17 TextType

TextType is used to define a character string (i.e. a finite set of characters) generally in the form of words of a language. It is derived from the type **string**.

Optional Attribute	Base XML Type	Description
languageID	language	An Identifier specifying the language used in the content component.
languageLocaleID	normalizedString	An Identifier specifying the locale of the language

1.18 String Type Usage

The support for UN/CEFACT core components and compatibility with OAGiS has required the use of three basic string types, each with separate purposes:

1. CodeType is required to be compatible with the core components
2. xsd:normalizedString is required to be compatible with OAGiS transaction processing
3. xsd:string is required to hold special characters (tab, LF, CR)

CodeType

- CodeType is used anyplace there is an enumeration.
- This follows the UN/CEFACT standard, it provides attributes that can be used to identify who "owns" the enumeration.
- This is derived from the xsd:normalizedString.

xsd:normalizedString

- xsd:normalizedString is a string in which line feeds, carriage returns, and tabs have been replaced by blanks. There can be multiple blanks in the string.
- This is used in B2MML for all of the attributes defined in the core component types. This should not be changed because it would no longer match the recommended Core Component types.
- This is also used in the transaction elements in order to match the definition in the OAGiS schemas. If we change it, then we are no longer compatible with the OAGiS transaction model. It would probably not be a problem to change this to xsd:string, **BUT** it could cause very difficult to find problems of compatibility (for example someone uses a tab instead of a space, or has a non-printing CR in a string that causes it not to match the expected string.)

xsd:string

- xsd:string is a string which may contain line feeds, carriage returns, and tabs.
- This is used in the core component types for names and format strings, where tabs, LF, CR may be significant. The B2MML mapping matches the current definition of the Core Component types and should not be changed.
- This is used in the places where the tab, LF, CR characters may be significant. This includes tag delimiters, order delimiters delimited data, and all "otherValue" attributes in enumerated lists. These should not be changed,

because the tabs, LF and CR characters are important.   The “otherValue” types could probably be changed to xsd:normalizedString without any major impact, because these are usually just vendor specific enumerations.

3 TRANSACTION DEFINITIONS

The B2MML-Vxx-common.xsd contains a set of elements used to support the transactions as defined in the ISA-95 Part 5 Business-to-Manufacturing Transaction standard. Transactions define sets of messages that are exchanged between applications according to a specific set of rules. The transaction model follows the OAGiS 9.6 model for transaction messages using the OAGiS XML schema structure, but using data objects (nouns) that are B2MML elements (relating to the ISA-95 data objects).

Transaction messages are based on the concept of VERBS and NOUNS. Verbs define the action to be taken or the response to an action. Nouns define the data objects the actions are taken on. The top level element of a XML document (the message) is named as the combination of the verb and the noun. For example, a “Get” verb on ***OperationsSchedule*** nouns would have an element named ***GetOperationsSchedule***.

Three different transaction models are defined:

1. A PULL model where a user of data requests the data from a provider using a GET verb, and where the provider of the data responds with a SHOW verb.
2. A PUSH model where a provider of data requests an action (processing, changing, or canceling) on the data by another user.
    a) A request to process the attached data is sent using the PROCESS verb and an optional response to the processing is returned using the ACKNOWLEDGE verb.

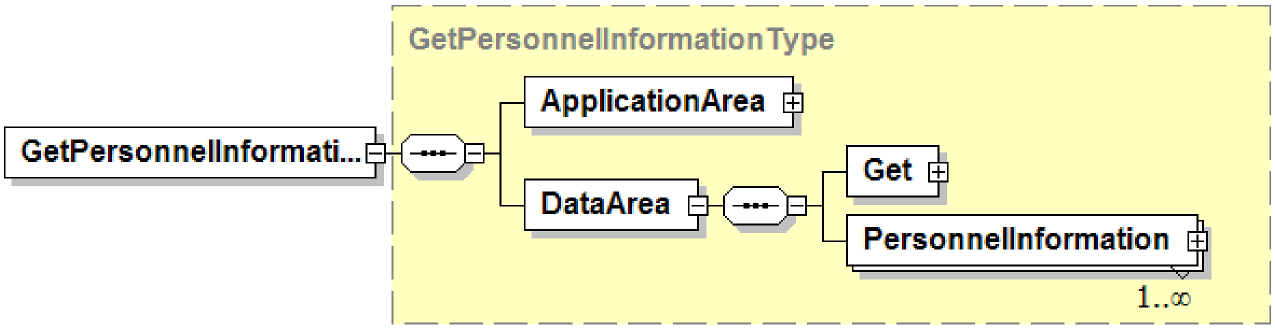
       Note: The definition of word “process” as meaning “deal with” or “handle”. A PROCESS verb is often the equivalent of a command to add the data, but usually the receiving entity performs further actions as a result of receiving the data.

    b) A request to change information is sent using a CHANGE verb and an optional response to the change is returned using the RESPOND verb.
    c) A request to cancel the attached data is sent using the CANCEL verb.

       Note: The request to cancel indicates that the sender no longer needs the data. Because the CANCEL is not sent by the owner of the data, the data are not necessarily deleted.
3. A PUBLISH model where the publisher of data sends to users (subscribers) of the data.
    a) A notification of new data is sent using a SYNC verb and the ADD option.
    b) A notification of changed data is sent using a SYNC verb and the CHANGE option.
    c) A notification of deleted data is sent using a SYNC verb and the DELETE option.

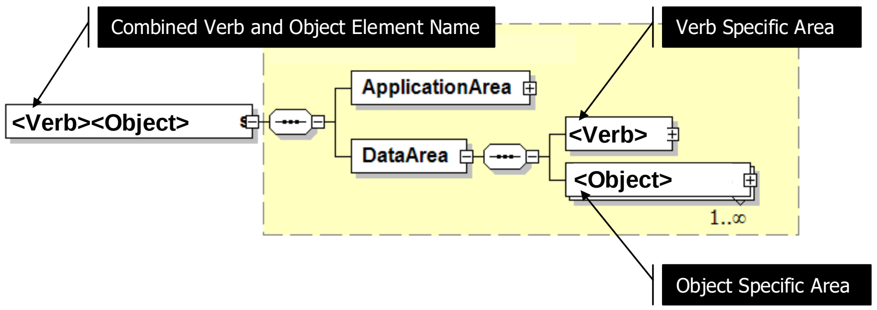
1.19 Standard Transaction Element Structure

The standard structure used for all transaction elements is an element with the verb name prefixed to the element name. For example, the element used to contain a “***Get***” message for a “***PersonnelInformation***” element would be “***GetPersonnelInformation***”. Each transaction element contains two elements, an ***ApplicationArea*** and a ***DataArea***, as shown in the figure and partial XML sample below.



```
<GetPersonnelInformation … releaseID="B2MML-V04RC01">
    <ApplicationArea>
        …
    </ApplicationArea>
    <DataArea>
            <Get>
                    …
            </Get>
            <PersonnelInformation>
                    …
            </PersonnelInformation>
    <DataArea>
```

All transaction elements contain the same ***ApplicationArea*** element (see definition in Section 1.24). Each ***DataArea*** is unique to the specific element type being exchanged. The DataArea contains two elements, an element that is specific to the verb (***Get***, ***Show***, ***Process***, ***Confirm***, ***Acknowledge***, …) and an element that defines the specific exchanged element (***PersonnelInformation***, ***Equipment***, ***MaterialLot***, ***ProductionSchedule***, ***ProcessSegment*****,** …).



All common transaction element types are prefixed with "***Trans***" and postfixed with "***Type***". For example the ***ApplicationArea*** is defined in the type ***TransApplicationAreaType***, and the **Get** is defined in the type ***TransGetType***.

1.20 Message Confirmation

Any message may request confirmation of receipt of the message using a CONFIRM option that is defined in the message's *ApplicationArea*. The confirmation indicates successful processing of the message and returns error conditions if the initiating message could not be processed. The CONFIRM option may specify the following options:

Option	Option Description
Never	No confirmation requested. (*Note: Default value if option not defined.*)
OnError	Send back a confirmation only if an error has occurred.
Always	Always send a confirmation regardless of the local processing.

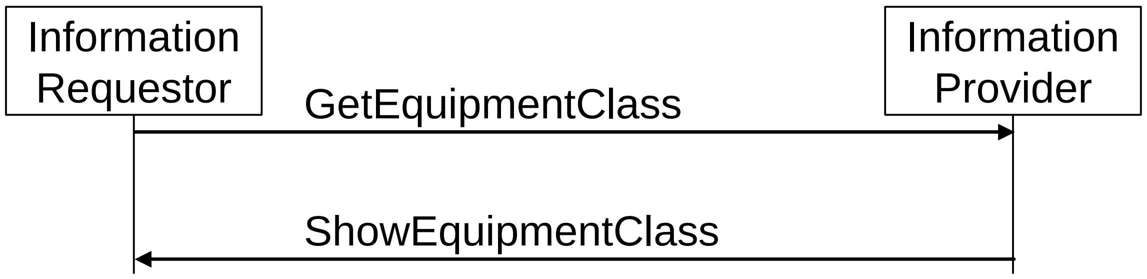
All confirmations are returned in a single message (XML element) type of *ConfirmBOD*. (Note: This follows the OAGiS definitions, where BOD is short for Business Object Document.)

***NOTE**: While any message may request confirmation (including a ConfirmBOD message), the recommended use is to only request confirmations for critical messages and only on CANCEL messages. (Confirmation on SYNC messages may lead to a large number of messages that the publisher could take no effective action on, GET messages have SHOW responses, PROCESS messages have ACKNOWLEDGE responses, and CHANGE messages have RESPOND responses.)*

1.21 PULL Transaction Model

The PULL transaction model is used when a user of data requests information from a provider of the data. The request is defined in a message that contains a GET verb and an empty or partially defined element.

For example the following diagram indicates a GET/SHOW transaction.



See the ISA-95 Enterprise-Control System Integration, Part 5, Business to Manufacturing Transaction standard for a complete specification of the empty and partially defined element rules on the GET.

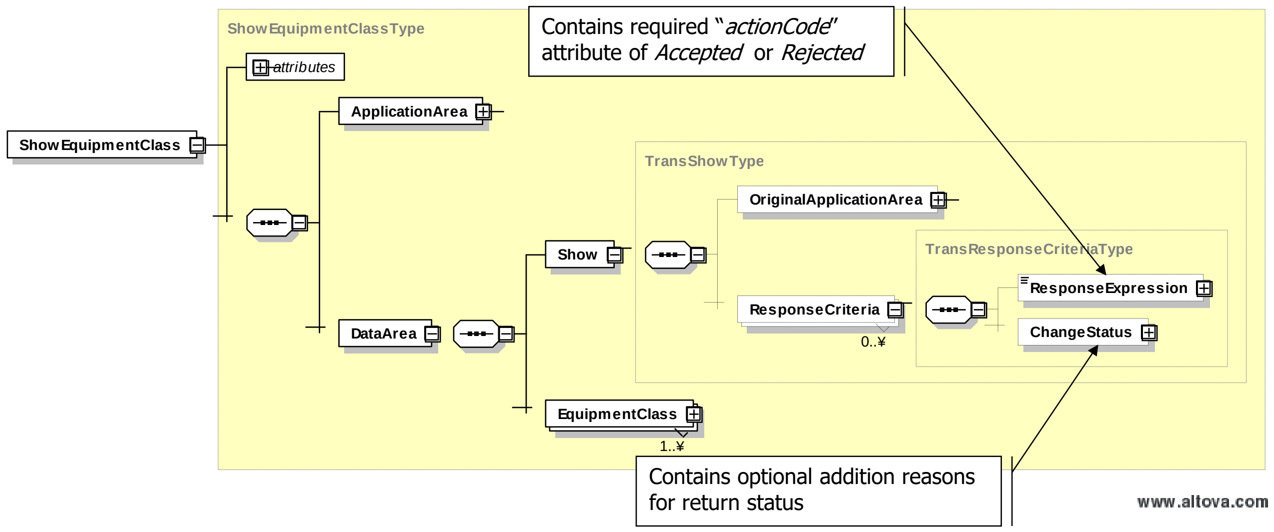
For example, the ***GetEquipmentClass*** message would contain a partially defined ***EquipmentClass*** element, as shown in the following text from the ISA-95 Part 5 standard[1].

Value of Equipment Class ID	Value of Equipment Class Property ID	Equipment Class Property Value	Action on Object(s) Specified for the GET verb

[1] Reprinted with permission of ISA, Research Triangle Park, NC, from the *ISA-95 Enterprise-Control System Integration, Part 5, Business to Manufacturing Transaction* standard.

IDs specified	*not specified*	*not specified*	Defines a request that the receiver is to return, in a SHOW message, all attributes about the specified *Equipment Classes*, all properties and their attributes, and the IDs of *Equipment* that are members of each *Equipment Class*.
IDs specified	IDs specified	*not specified*	Defines a request that the receiver is to return, in a SHOW message, all attributes about the specified *Equipment Classes*, all of the specified *Equipment Class Properties*, and the IDs of *Equipment* that are members of each *Equipment Class*.
IDs specified	IDs specified	Property Values Specified	Defines a request that the receiver is to return, in a SHOW message, all attributes about the specified *Equipment Classes* where the *Equipment Class Property* value matches the specified property value, all of the specified *Equipment Class Properties*, and the IDs of *Equipment* that are members of each *Equipment Class*.
Wildcard specified	*not specified*	*not specified*	Defines a request that the receiver is to return, in a SHOW message, all attributes and properties about the *Equipment Classes* that match the wildcard ID and the IDs of *Equipment* that are members of each *Equipment Class*. To return all *Equipment Classes*, specify a "*" as the wildcard.
Wildcard specified	Wildcard specified	*not specified*	Defines a request that the receiver is to return, in a SHOW message, all attributes of the *Equipment Classes* that match the wildcard IDs, and for each class return all *Equipment Class Properties* that match the property ID wildcards, and the IDs of *Equipment* that are members of each *Equipment Class*. To return a single property, specify the *Equipment Class Property* ID in the property ID wildcard. To return all *Equipment Class Properties*, specify a "*" as the property ID wildcard. To return a single *Equipment Class*, specify the ID in the wildcard ID. To return all *Equipment Classes*, specify a "*" as the ID wildcard.

The SHOW message contains a required response status attribute of either *Accepted* or *Rejected* in the *Show/ResponseCriteria/ResponseExpression* element as shown in the figure below.



Contains required "*actionCode*" attribute of *Accepted* or *Rejected*

Contains optional addition reasons for return status

1.22 PUSH Transaction Model

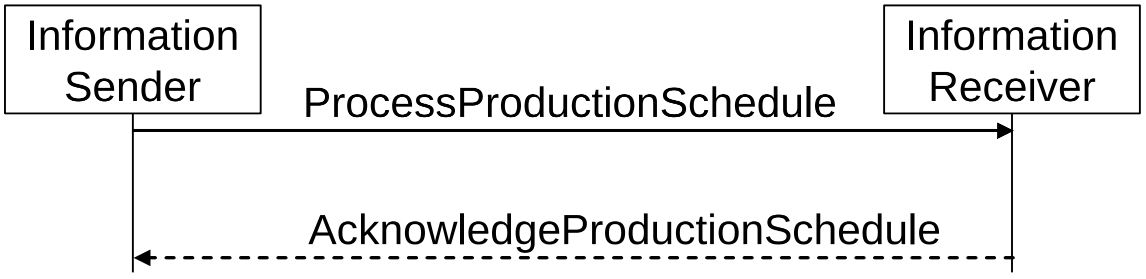
The PUSH model uses PROCESS/ACKNOWLEDGE, CHANGE/RESPOND, and CANCEL messages for an application that is not the owner of data to request processing, changing, or canceling data to the data owner.

For example, the following diagram indicates a Process/Acknowledge transaction for a ***ProductionSchedule*** element. The PROCESS message may contain a CONFIRM option and a ACKNOWLEDGE option, but normally only the ACKNOWLEDGE would be specified (not both).

The PROCESS Acknowledgement option may specify the following options:

Option	Option Description
Never	No acknowledge requested. (*Note: Default value if option not defined.*)
Always	Always send an acknowledge response.

See the ISA-95 Part 5 standard for a complete definition of the action to be performed as the result of receiving a PROCESS message for each object (element) type.

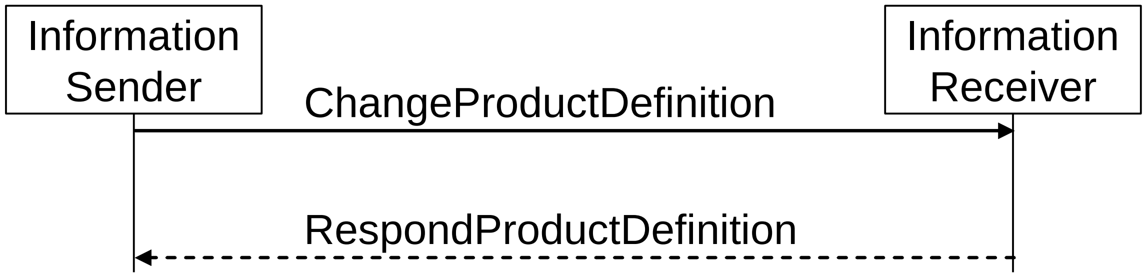


The following diagram illustrates a CHANGE/RESPOND transaction for a ***ProductDefinition*** element. The CHANGE message may contain a CONFIRM option and a RESPOND option, but normally only one or neither would be specified (not both).

The CHANGE respond option may specify the following options:

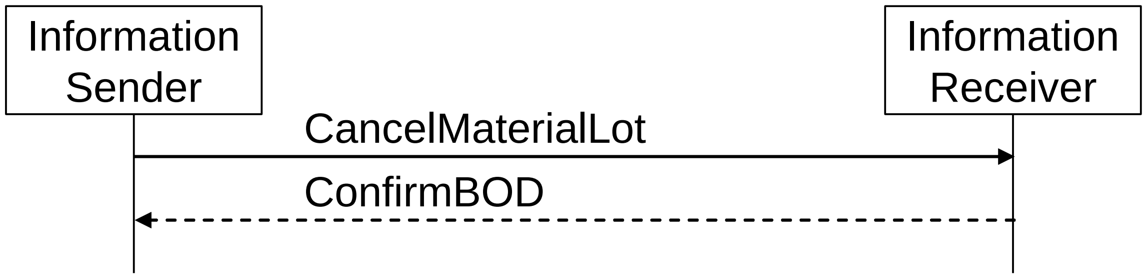
Option	Option Description
Never	No response requested. (*Note: Default value if option not defined.*)
Always	Always send a response.

See the ISA-95 Part 5 standard for a complete definition of the action to be performed as the result of receiving a CHANGE message for each object (element) type.



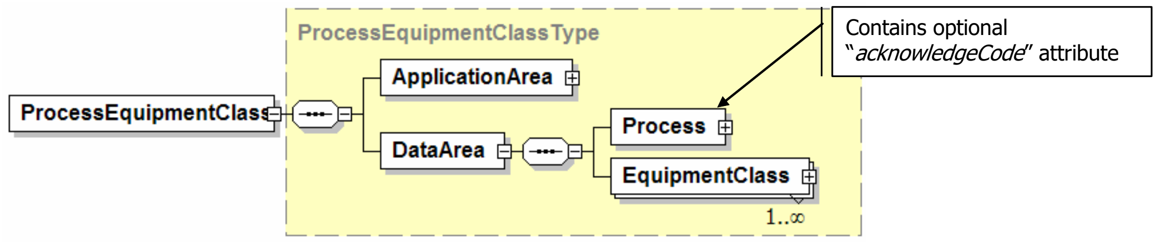
The following diagram illustrates a CANCEL transaction for a ***MaterialLot*** element. The CANCEL message may contain a confirmation option.

See the ISA-95 Part 5 standard for a complete definition of the action to be performed as the result of receiving a CANCEL message for each object (element) type.



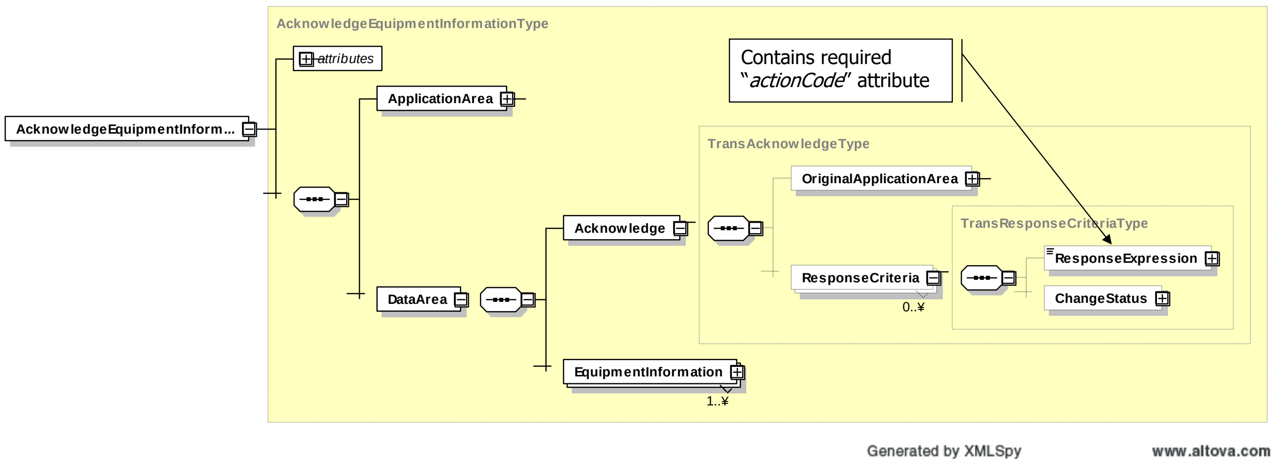
1.22.1 Process Acknowledgment

The acknowledge option is defined using optional attributes in PROCESS messages. The PROCESS verb contains an optional *acknowledgeCode* attribute, as shown below in the *Process* element.


ProcessEquipmentClassType
ProcessEquipmentClass
ApplicationArea
DataArea
Process
EquipmentClass
1..∞
Contains optional "*acknowledgeCode*" attribute

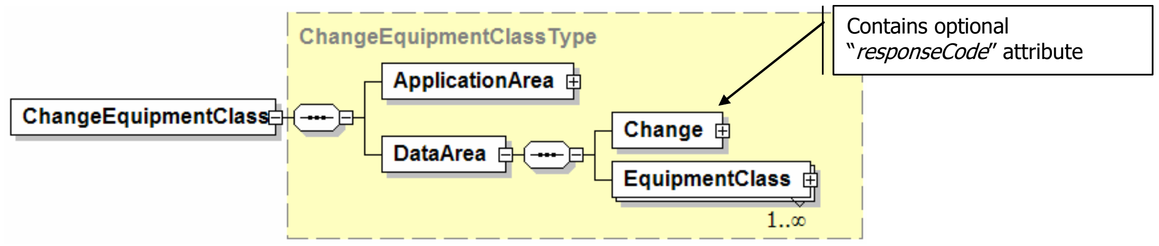
For consistency with OAGiS verb definitions, the PROCESS verb also contains an ActionCode element with ActionExpression elements. This element is not used in the B2MML transactions.

The ACKNOWLEDGE message contains a required status attribute in the *Acknowledge/ResposeCriteria/ResponseExpression* element, as shown below.



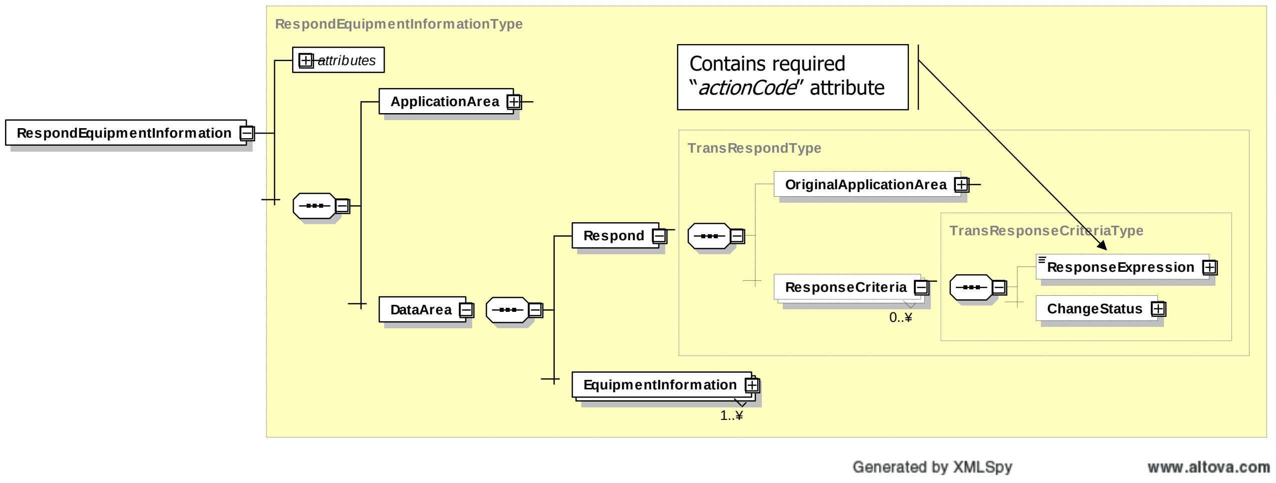
1.22.2 Change Response

The CHANGE verb contains an optional *responseCode* attribute, as shown below in the *Change* element.



For consistency with OAGiS verb definitions, the CHANGE verb also contains an ActionCode element with ActionExpression elements. This element is not used in the B2MML transactions.

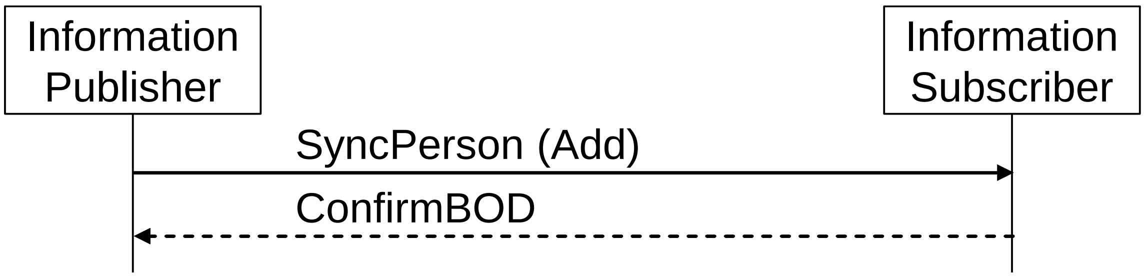
The RESPOND message contains a required status attribute in the *Respond/ResponseCriteria/ResponseExpression* element.



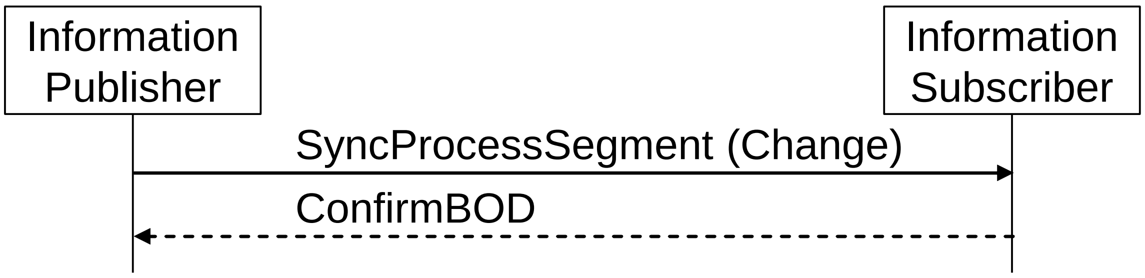
1.23 PUBLISH Transaction Model

The PUBLISH model uses SYNC messages with ADD, CHANGE, or DELETE options to indicate the action to be taken on the published data.

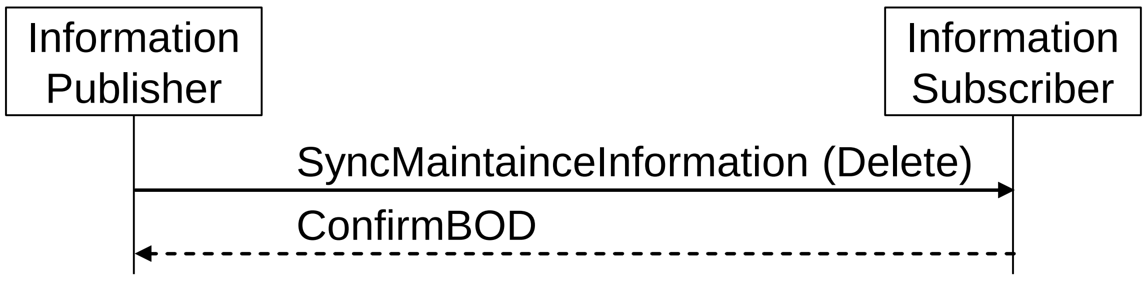
The following diagram illustrates a SYNC ADD transaction for a ***Person*** element. The ***SyncPerson*** (Add) message may be sent to multiple information subscribers when a new ***Person*** object is defined, but only one is shown in the example. The SYNC message may contain a confirmation option, but this is not normally used in PUBLISH transactions. See the ISA-95 Part 5 standard for a complete definition of the action to be performed as the result of receiving a SYNC ADD message for each object (element) type.



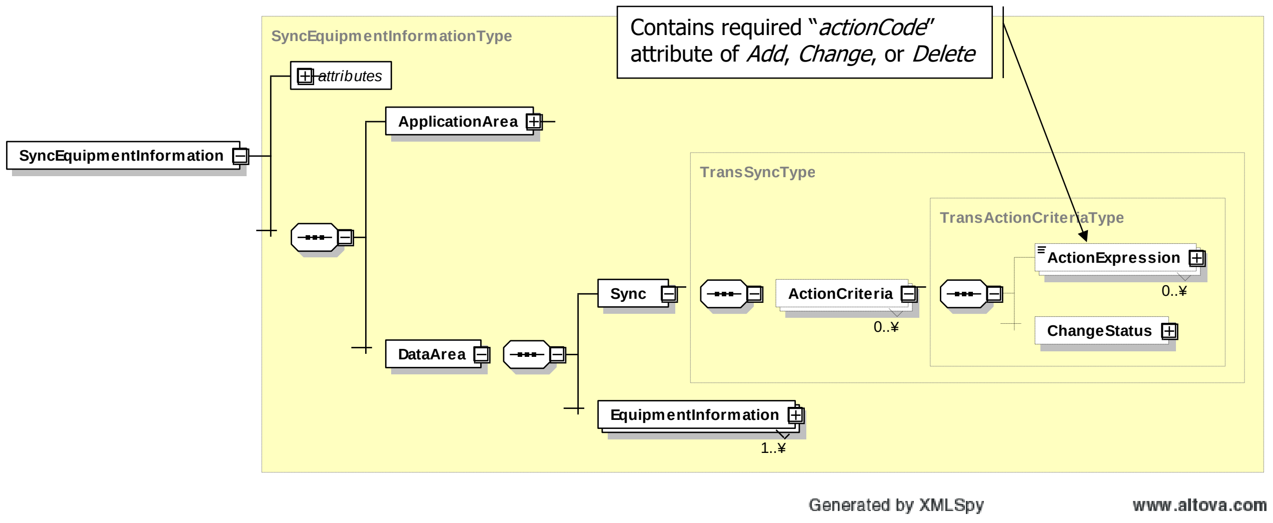
The following diagram illustrates a SYNC CHANGE transaction for a ***ProcessSegment*** element. The SyncPerson (Change) message may be sent to multiple information subscribers when a ***ProcessSegment*** object is changed, but only one is shown in the example. The SYNC message may contain a confirmation option, but this is not normally used in PUBLISH transactions. See the ISA-95 Part 5 standard for a complete definition of the action to be performed as the result of receiving a SYNC CHANGE message for each object (element) type.



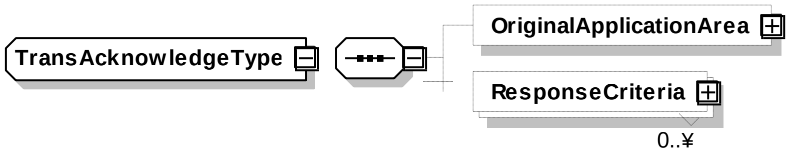
The following diagram illustrates a SYNC DELETE transaction for a ***MaintenanceInformation*** element. The SyncPerson (Change) message may be sent to multiple information subscribers when ***MaintenanceInformation*** is deleted, but only one is shown in the example. The SYNC message may contain a confirmation option, but this is not normally used in PUBLISH transactions. See the ISA-95 Part 5 standard for a complete definition of the action to be performed as the result of receiving a SYNC DELETE message for each object (element) type.



The SYNC options are specified in the required attribute of the *Sync/ActionCriteria/ActionExpression* element. There are multiple *ActionCriteria* elements allowed by the schema, but only one is used in the B2MML transactions.



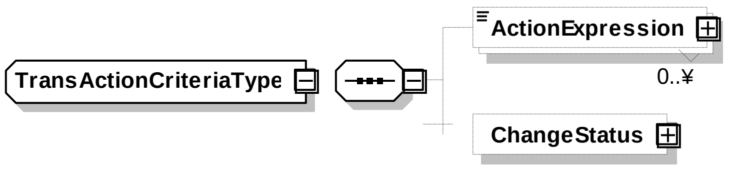
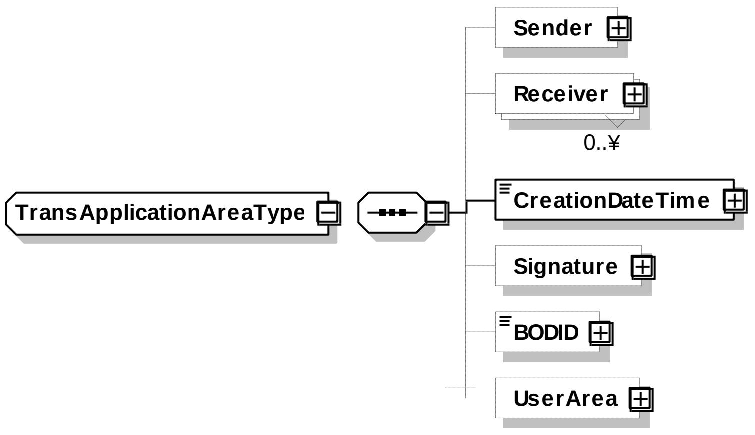
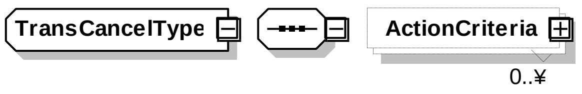
1.24 Common Transaction Elements

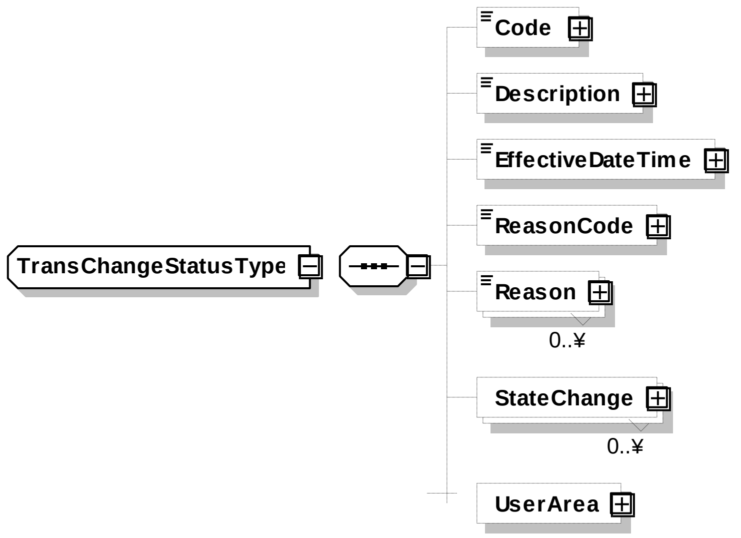
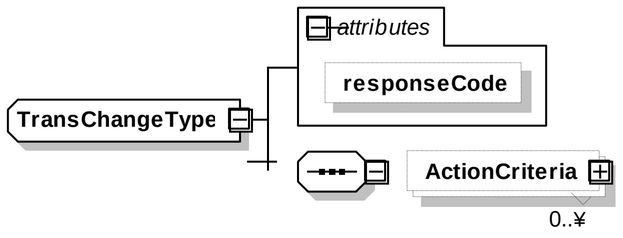
Type Name **Element Name**	**Description**
TransAcknowledgeType Acknowledge	Complex data type for an ACKNOWLEDGE verb in an *Acknowledge<Object>* message. A complex type that contains two elements: • **OriginalApplicationArea:** An optional copy of the ApplicationArea from the requesting Process message (see ***TransApplicationAreaType***). This is included to assist in error handling by the requesting application. • **ResponseCriteria:** Zero or more elements (See ***TransResponseCriteriaType***) that contain additional acknowledge information (including an action code). o If no **ResponseCriteria** is present, then the action code of "**Accepted**" is the default. o The first **ResponseCriteria** element defines the acknowledgement option. The meanings of any additional **ResponseCriteria** are not defined in B2MML.  TransAcknowledgeType OriginalApplicationArea ResponseCriteria 0..¥

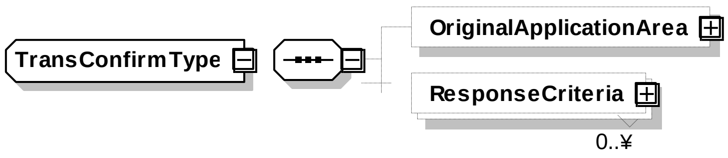
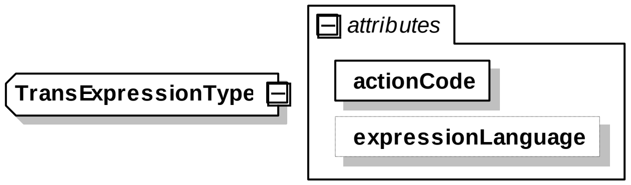
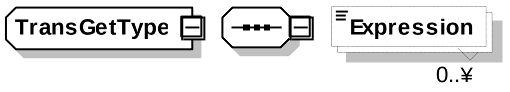
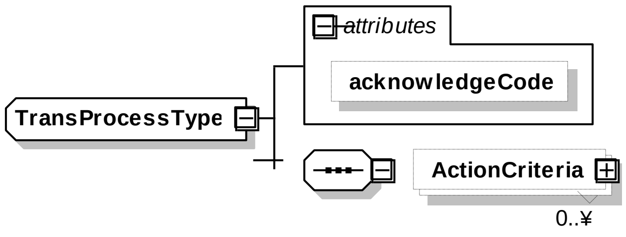
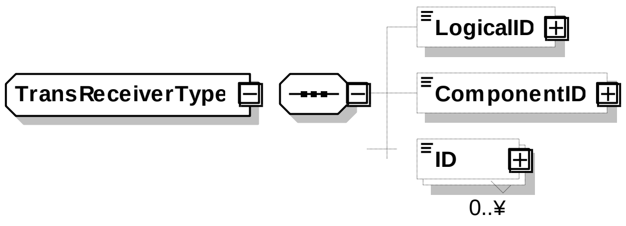
TransActionCodeEnumerationType	A string that contains an identification of the action to be performed as part of a verb. The action codes are used for SYNC messages and as responses in CHANGE, CONFIRM, ACKNOWLEDGE and RESPOND messages. The following enumerations are defined:

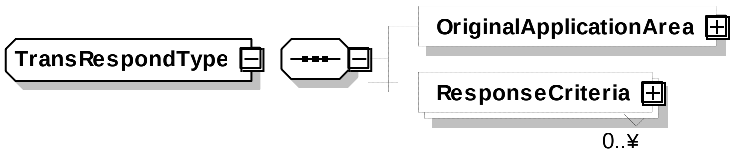
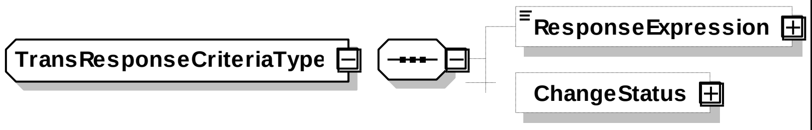
Enumeration	Action Meaning
Add	Used in a SYNC message to indicate a SYNC ADD action to be performed.
Change	Used in a SYNC message to indicate a SYNC CHANGE action to be performed.
Delete	Used in a SYNC message to indicate a SYNC DELETE action to be performed.
Replaced	Used in a RESPOND message to indicate that the change was performed
Accepted	Used in an ACKNOWLEDGE message to indicate that the PROCESS was performed Used in a CONFIRM message to indicate that the action was performed. Used in a SHOW message to indicate that the GET was performed and all data was returned. This may include a SHOW message with no data elements, if the section criteria identified no data to be returned.
Modified	Used in an ACKNOWLEDGE message to indicate that the PROCESS was performed but that the information was modified. Used in a RESPOND message to indicate that the CHANGE was performed but that the information was modified. Not used in a SHOW message.
Rejected	Used in an ACKNOWLEDGE message to indicate that the PROCESS was rejected. Used in a RESPOND message to indicate that the CHANGE was rejected. Used in a CONFIRM message to indicate that the action was rejected. Used in a SHOW message to indicate that the GET was not performed and that not data was returned.

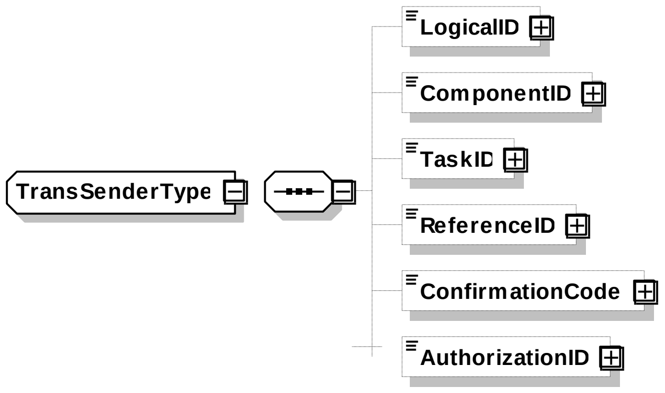
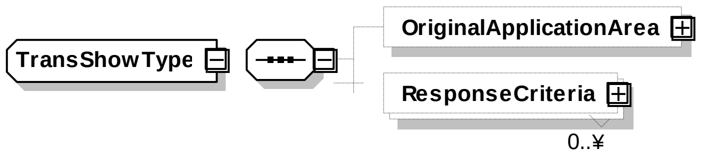
TransActionCodeType	A union type of an enumeration (see **TransActionCodeEnumerationType**) and a **normalizedString**. This allows either a defined enumeration value (see above) or a user defined string. The meanings of any user defined action code strings are not defined in B2MML.

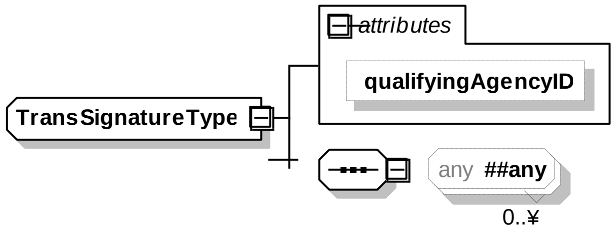
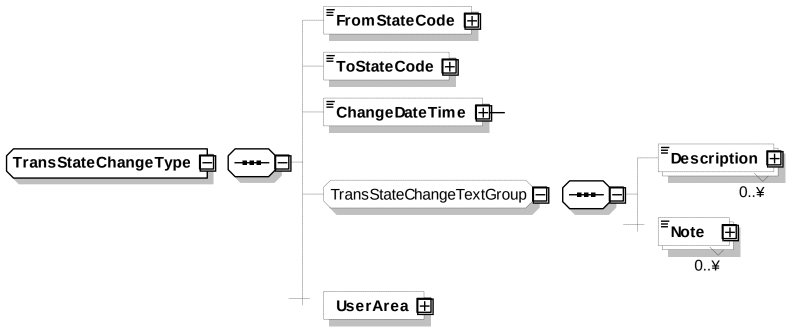
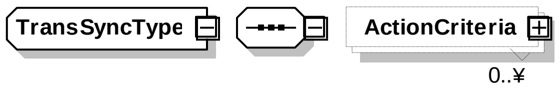
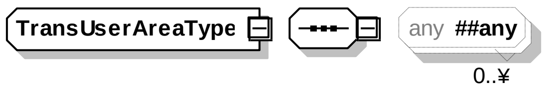
TransActionCriteriaType ActionCriteria	Data Type for a SYNC, PROCESS, CHANGE, and CANCEL message. It contains one optional element **ActionExpression** (see ***TransExpressionType***) that contains an action code for SYNC messages. It also contains an optional **ChangeStatus** (see **TransChangeStatusType**) element for a definition of the change. If no **ActionExpression** is defined for a SYNC message, then the action code of "**Add**" is the default.  TransActionCriteriaType ActionExpression 0..¥ ChangeStatus
TransApplicationAreaType ApplicationArea OriginalApplicationArea	A complex type that contains: • An optional identification of the sender of the message (see **TransSenderType**) • Zero or more optional identifications of the receiver of the message (see **TransReceiverType**) • A required element with the creation date & time of the message. • An optional electronic signature that can be used to sign the transaction message • An optional ID (**BODID**) to be applied to exchanged data object. This should be a GUID (Globally Unique Identifier) that uniquely identifies the data object. • An optional user area for user extended data.  Sender Receiver 0..¥ TransApplicationAreaType CreationDateTime Signature BODID UserArea
TransCancelType	Data type for a CANCEL verb in a *Cancel<Object>* message. Contains an optional attribute **responseCode** of type **TransResponseCodeType**. The responseCode specifies if a response is required. The complex type also that contains an optional ActionCriteria (see **TransActionCriteriaType**) element for compatibility with OAGiS; however the ActionCriteria elements are not defined in a **TransCancelType** element in B2MML. 

TransChangeStatusType ChangeStatus	Defines the description for a response. **Code** (CodeType): A user defined element for communication of all codes. No standard code types are defined. **Description** (DescriptionType): A text description of the overall reason for the response. **EffectiveDateTime** (DateTimeType): The effective date and time that response was generated, to allow backtracking of the reason for the response. **ReasonCode** (CodeType): Identifies the reason for the response activity. **Reason** (TextType): Text description of the reasons for the response. **StateChange**: Information about any state changes associated with the response. **UserArea**: User defined ##any type.  Code Description EffectiveDateTime ReasonCode Reason 0..¥ StateChange 0..¥ UserArea TransChangeStatusType
TransChangeType	Data type for a CHANGE verb in a *Change<Object>* message. Contains an optional attribute **responseCode** of type **TransResponseCodeType**. The responseCode specifies if a response is required. The complex type also that contains an optional ActionCriteria (see **TransActionCriteriaType**) element for compatibility with OAGiS; however the ActionCriteria elements are not defined in a **TransChangeType** element in B2MML.  attributes responseCode TransChangeType ActionCriteria 0..¥
TransConfirmationCodeType	A string used to indicate a confirmation, acknowledge, or respond code option in a message. The value must be one of the following standard enumerations: **Always**, **Never**, **OnError** • Always → Always return a confirm, acknowledge, or respond message. • Never → Never return a confirm, acknowledge, or respond message. • OnError → Only return a confirm, acknowledge, or respond message if an error occurred during processing of the message's action.

TransConfirmType	Data type for a CONFIRM verb in a *ConfirmBOD* message. See the ConfirmBOD documentation for the full description of the confirm message.  TransConfirmType OriginalApplicationArea ResponseCriteria 0..¥
TransExpressionType ActionExpression ResponseExpression	A complex type with: • A simple type of "**token**" • A required attributed of **actionCode** (see **TransActionCodeType**) • An optional **expressionLanguage** attribute that specifies the language that may be used to interpret the tokens in the expression. The meaning of any text included in the token in a **TransExpressionType** is not defined in B2MML.  attributes TransExpressionType actionCode expressionLanguage
TransGetType Get	Data type for a GET verb in a *Get<Object>* message. There are no attributes. 
TransProcessType Process	Data type for a PROCESS verb in a *Process<Object>* message. Contains an optional attribute **acknowledgeCode** of type **TransResponseCodeType**. The responseCode specifies if a response is required.  attributes acknowledgeCode TransProcessType ActionCriteria 0..¥
TransReceiverType Receiver	Contains information about the expected receiver of the message. This contains an optional **LogicalID** of the server and application for which the BOD is intended. This contains an optional **ComponentID** of the server and application for which the BOD is intended. It provides a finer level in addition to the LogicalID. This contains zero or more optional **ID**s for the receiver of the message. 

TransRespondType Respond	Data type for a RESPOND verb in a *Respond<Object>* message.  TransRespondType OriginalApplicationArea ResponseCriteria 0..¥
TransResponseCodeType	A string used to indicate if response is requested for the sent. The value must be one of the following standard enumerations: **Always**, **Never** • Always → Always return a response message. • Never → Never return a response message.
TransResponseCriteriaType ResponseCriteria	Data Type for an ACKNOWLEDGE, CONFIRM, SHOW, and RESPOND response. It contains one optional element **ResponseExpression** (see ***TransExpressionType***) that contains an action code. If no **ResponseExpression** is defined, then the action code of "**Accepted**" is the default. **ChangeStatus** is an optional element that contains the reason for the response.  TransResponseCriteriaType ResponseExpression ChangeStatus

TransSenderType	A complex type that identifies characteristics and control identifiers that relate to the application that created the transaction message. The sender area can indicate the logical location of the application and/or database server, the application, and the task that was executing to create the data object. • The format for the **LogicalID** is not defined in B2MML. May contain a logical (not physical) identification of the sending task. • The format for the **ComponentID** is not defined in B2MML. May contain additional detail for the **LogicalID**. • The format for the **TaskID** is not defined in B2MML. It may describe the business event that initiated the need for the Business Object Document to be created. • The format for the **ReferenceID** is not defined in B2MML. It may contain additional information that enables the sending application to indicate the instance identifier of the event or task that caused the data to be created. • The format for **ConfirmationCode** is defined in **TransConfirmationCodeType**. • The format for the **AuthorizationID** is not defined in B2MML. It may identify the authorization level of the user or application that is sending the data. This authorization level may indicate to the receiving system indicates what can be done on request.  TransSenderType LogicalID ComponentID TaskID ReferenceID ConfirmationCode AuthorizationID
TransShowType	Data type for a SHOW messages.  TransShowType OriginalApplicationArea ResponseCriteria 0..¥

TransSignatureType	A **##any** type that is used if the message is to be signed. It supports any digital signature that maybe used by an implementation. The optional qualifyingAgencyID attribute identifies the agency that provided the format for the signature. In order to support digital signature specifications currently available or that will be developed in the future. The Signature element is defined to have any content from any other namespace. The choice of which digital signature to use is left up to the specific implementation.  attributes qualifyingAgencyID TransSignatureType any ##any 0..¥
TransStateChangeType StateChange	Defines any state change associated with the response, such as a change from effective to obsolete. • **FromStateChange** (CodeType): Old state • **ToStateChange** (CodeType): New state • **ChangeDateTime** (DateTimeType): Date and time the change occurred. • **Description** (TextType): Descriptions of the change. • **Note** (TextType): Secondary notes associated with the change. • **UserArea**: User ##any type  FromStateCode ToStateCode ChangeDateTime TransStateChangeType TransStateChangeTextGroup Description 0..¥ Note 0..¥ UserArea
TransSyncType	Data type for SYNC messages. 
TransUserAreaType UserArea	A **##any** type that is used to contain user data in the application area. 

4 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 ☺☺☺
  - Define properties where available to avoid requiring schema extensions

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

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

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

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

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

There are one type of schema extension method defined (5), the 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.

4.1 Extensions with no Schema Changes

4.1.1 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, and not available) and equipment location (production, clean room, and 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 do not 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. Also see the B2MML-MasterDataProfile.xsd file and the associated TR95.01 Master Data Profile Template for a method to configure property names.

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.

4.1.2 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. Also see the B2MML-MasterDataProfile.xsd file and the associated TR95.01 Master Data Profile Template for a method to configure parameter names.

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 *SegmentRequirementTypes*.
- A *ProductionPerformance* may define values for parameters in *ProductionData* elements in *SegmentResponseTypes*.

4.1.3 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. Also see the B2MML-MasterDataProfile.xsd file and the associated TR95.01 Master Data Profile Template for a method to configure segment names.

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 >
```

4.2 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. Also see the B2MML-MasterDataProfile.xsd file and the associated TR95.01 Master Data Profile Template for a method to configure “Other” value names.

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 >
```

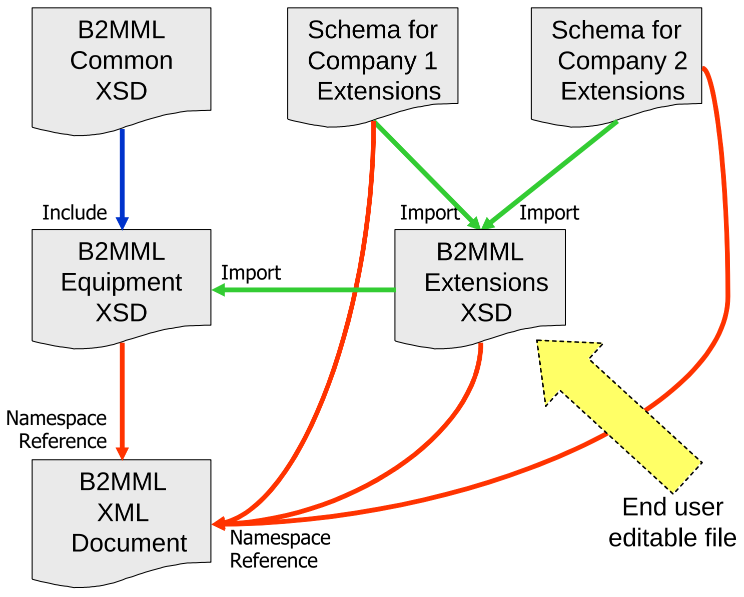
4.3 Extension using Schema Changes

4.3.1 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-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-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.mesa.org/xml/b2mml-v0300"
               xmlns              = "http://www.mesa.org/xml/b2mml-v0300"
               xmlns:Extended     = "http://www.mesa.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 affecting 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.

4.4 Example of a Custom Schema

The following schema is an example custom schema developed using the infrastructure of MESA. It illustrates how the UN/CEFAC core components, the 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.mesa.org/xml/B2MML-V0700"
  xmlns    ="http://www.mesa.org/xml/B2MML-V0700-Extensions"
  targetNamespace  ="http://www.mesa.org/xml/B2MML-V0700-Extensions"
  elementFormDefault  ="qualified"
  attributeFormDefault="unqualified">

  <!-- Import the Common schema   -->
  <xsd:import namespace="http://www.mesa.org/xml/B2MML-V0700"
              schemaLocation="B2MML- Common.xsd"/>

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

    <xsd:annotation>
    <xsd:documentation>

        This custom schema is developed within the infrastructure of the MESA 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 means 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.mesa.org/xml/B2MML-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 MESA."

    </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>
```

5 B2MML AND OAGIS DIFFERENCES

B2MML is designed to implement a subset of the OAGiS 9.6 message rules that are consistent with the ISA-95 Part 5 definitions. Specifically the B2MML ***ApplicationArea*** and verb elements are subsets of the full OAGiS 9.6 specifications, with the differences listed in the following table.

Element	Differences
Show	The OAGiS 9.6 "*Show*" specification includes a set of optional attributes (recordSetStartNumber, recordSetCount, recordSetTotal, recordSetCompleteIndicator, and recordSetReferenceId) that are used by the responding task to indicate the status of the request and to define the scope of the information returned. These attributes are not defined for B2MML. The Show message should return all elements of the Get request.
Get	The OAGiS "Get" specification includes a set of optional attributes (uniqueIndicator, maxItems, recordSetSaveIndicator, recordSetStartNumber, and recordSetReferenceId) that are used by the requesting application to control how many elements are returned. These attributes are not defined for B2MML. The Show message should return all elements of the Get request.
ActionExpression **ResponseExpression**	In B2MML this contains a required attribute that contains an actionCode.
ConfirmBOD	B2MML contains only the Original Application Area, a free form text group, and user data. OAGiS 9.6 also contains BOD Failure Message, BOD Success Message, and Partial BOD Failure Message areas.
DateTimeType	B2MML has the DateTimeType derived from the xsd:dateTime type. OAGiS 9.6 has the DataTimeType derived from xsd:string. B2MML is more restrictive than OAGiS, but OAGiS recommends the use of ISO 8601 CE format.