Batch Markup Language

General and Site Recipes

BatchML-GeneralRecipe

Version 0500 – March 2011

Table of Contents

Change History:

Change	Date	Person	Description
V0401	Oct 2008	Dennis Brandl	Initial release
V0500	Jun 2010	Dennis Brandl	Updated version documentation and WBF name in copyright. Removed AnyType and ##any extensions.

Material from ANSI/ISA-88 and ANSI/ISA-95 series of standards used with permission of ISA - The Instrumentation, Systems, and Automation Society, www.isa.org

1 Schema Scope

This document defines the information about General Recipes, Site Recipes, Equipment Requirement Libraries (*represented as Resource Constraint Libraries*), and Process Element Libraries. This information is based on the data models and attributes defined in the ANSI/ISA 88.00.03-2003 Batch Control Part 3: General and Site Recipe Models and Representation 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.

1.1 Key Information Assumptions

The data represented in these schemas is derived from the UML model below, defined in ANSI/ISA 88.03, Figure 8, and shown below in Figure 1.
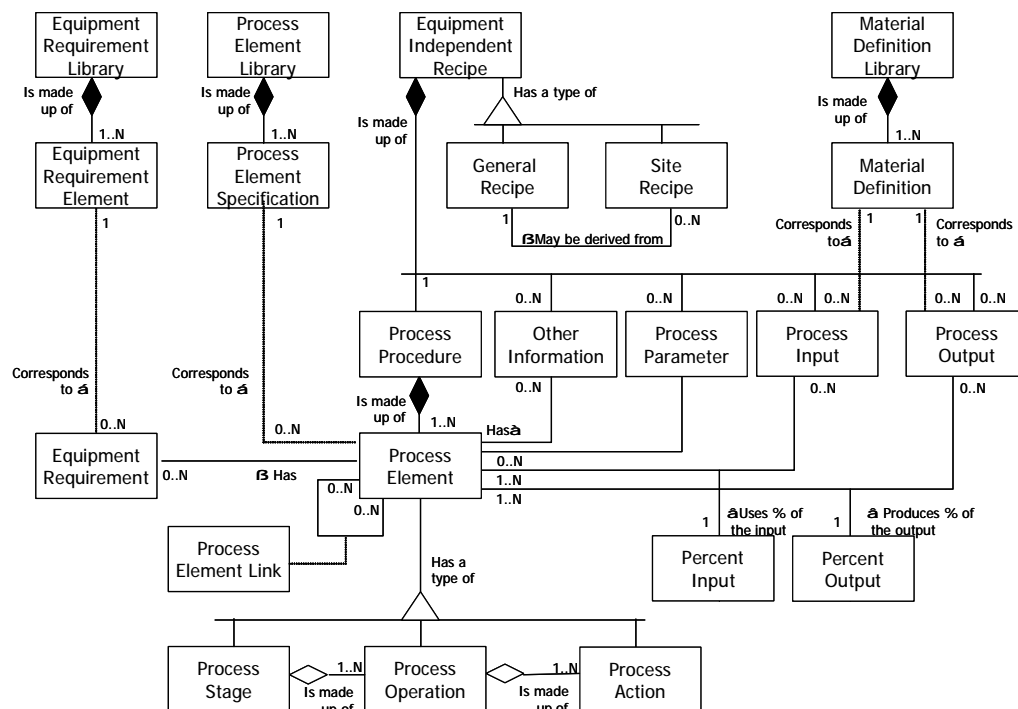
Figure 1 – Model of Exchanged Equipment Information[1]

There are three libraries and one hierarchical element defined in the standard. The BatchML-GRecipe schemas define two libraries and general recipes, but do not define the Material Library. The Material Definition Library can be exchanged using the **B2MML-Material** schema, also defined by WBF (www.wbf.org).

BatchML-GRecipe is defined in the **B2MML** name space in order to pick up common type definitions. As a result a couple of ISA 88 Part 3 elements have been renamed in order to reduce the chance for name conflicts. The data model implemented in BatchML-GRecipe is shown in Figure 2. The gray boxed highlight the differences between the models.

---

[1] From ANSI/ISA 88 Part 3

There are several top level elements defined, as depicted in Figure 2:

1. GRecipeInformation – A container for multiple general and/or site recipes. This element has transactions defined to allow exchange of ISA 88 Part 3 information.

2. ResourceConstraintLibrary – An resource constraint (equipment requirement) library. This element has transactions defined to allow exchange of ISA 88 Part 3 information.

3. ProcessElementLibrary – A process element library. This element has transactions defined to allow exchange of ISA 88 Part 3 information.

4. GRecipe – A single general or site recipe. This element is defined to allow a single XML file to represent a single recipe, such as building a directory of general recipes.

5. ResourceConstraint – A single resource constraint (equipment requirement). This element is defined to allow a single XML file to represent a single resource constraint.

6. ProcessElement – A single process element. This element is defined to allow a single XML file to represent a single process element specification, such as building a directory of process action specifications.

Figure 2 - Exchanged Elements

1.2 Key Use Assumptions
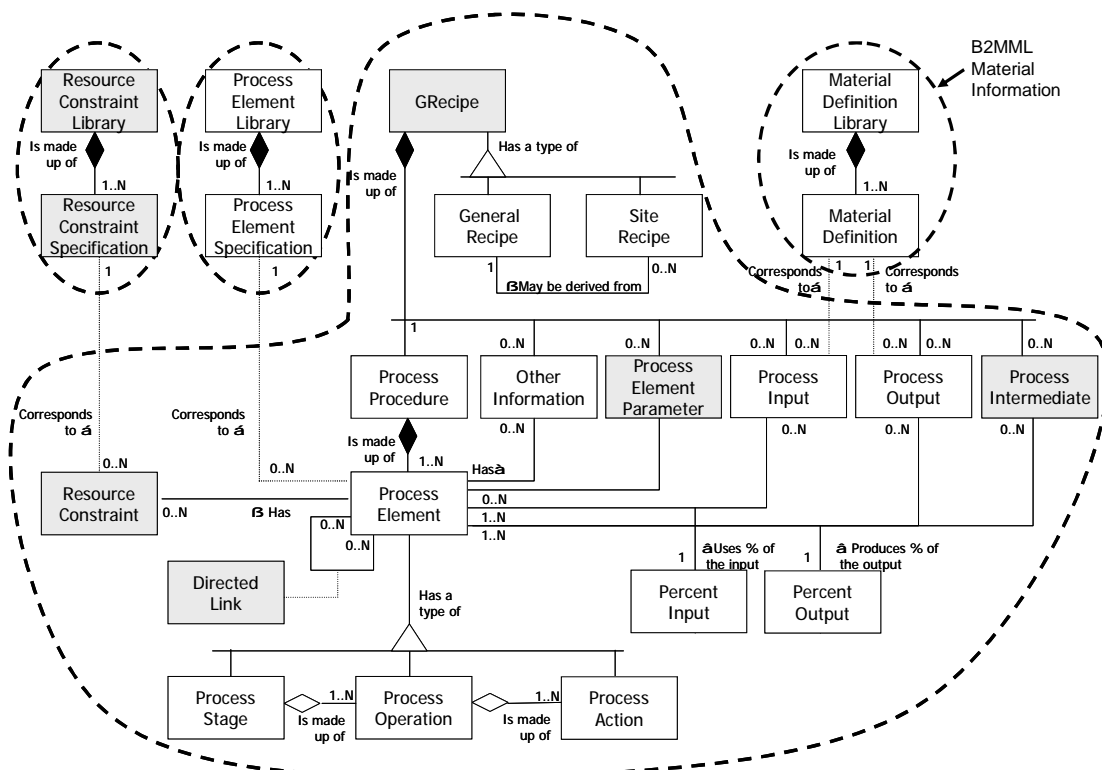
The key use assumption for **BatchML-GRecipe** is that it will be used to exchange General and Site Recipes between different systems. Material information may be exchanged using the **B2MML-Material** schemas, also from WBF.

The schemas may also be used to exchange libraries of Resource Constraints (Equipment Requirement specifications) that may be referenced by a process element

The schemas may also be used to exchange libraries of Process Element Specifications. For example there is usually a process element library that would contain all of the available Process Actions that could be used in a General Recipe.

Much of the structure of the **BatchML-GRecipe** schema is based on the structure of used to depict the recipes, as defined in ISA 88.03, Clause 7. This representation requires that elements not on the object model (in Clause 6) must also be included. See below for additional details.

The **BatchML-GRecipe** schema includes the **B2MML-Common** schema to pick up types that are the same in both schemas and to pick up the UN/CEFACT Core Component types defined in B2MML

1.3 Type Names

The XML schema uses a model that defines simple and complex data types for each element. The data types all follow the convention of a suffix of "**Type**" added to the element name.   Some element types also have a "GRecipe" that prefixes the type name in order to reduce the chance for a namespace type conflict with B2MML–Common.

Schema definition:

```
<xsd:element name = "ProcessElement"  type = "ProcessElementType"/>



<xsd:complexType name=" ProcessElementType">
        . . .
</xsd:complexType>
```

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 BatchML-GRecipe elements.

1.4 User Element Extensibility

Recipes and Resource Constraints have "Properties", similar to the properties defined in the ISA 95 standard for Equipment, Equipment Specifications, Equipment Requirements, Process Segment Requirements, and other elements related to equipment and process segments. The properties provide a method to define and document project and company specific information without changes to the schemas. Process Elements contain an "OtherInformation" element that serves the same purpose for process elements.

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.

See the document **B2MML-V04-Extensions.doc** for a complete explanation of user extensibility. The extensions for **BatchML-GRecipe** and defined in the **BatchML-GExtensions.xsd** file and follow the same model.

1.5 Use of IDs in Schema Definitions

Many types have an ID element. Check the documentation below, these IDs may be local in scope (only defined so that elements in the exchanged file may identify other elements) or have an external scope and serve to uniquely identify the exchanged object.

1.6 Common Data Types

The BatchML BatchInformation schema used the B2MML Common schema to pick up common data types.

See the documentation for the Core Compoenents and B2MML Common Types used in BatchML in the file:

B2MML-V0401-Common.doc

1.7 Core Components

The BatchML BatchInformation schema used the B2MML Core Component schemas.

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.

The core components are defined in the schema file:

B2MML-v0401-CoreComponents.xsd

1.8 Resource Constraint Library

From ANSI/ISA 88.03, Clause 5.6.3 – "*Equipment requirements define the constraints to be placed on target equipment, usually where the constraints impact the chemical or physical processing of the material. For example, the chemistry of a process stage could require that the operations occur in glass-lined reactors and Teflon-lined pipes, because materials being processed will interact with normal steel containers and pipes.*"

The term Resource Constraint is used in this schema to avoid name conflicts with other WBF schemas. An equipment requirement may be more than simply related to physical production equipment. They can be used to define all constraints, such as environmental (clean room requirements), personnel safety (such as gowning or protection requirements), physical constraints (such as maximum room humidity), or any other constraint that may be applied to a Process Stage, Process Operation, or Process Action.

The terminology Resource Constraint is used to contain equipment requirements. These are defined in ResourceConstraintLibrary (*ResourceConstraintLibraryType*) elements. An ResourceConstraintLibrary contains a set of individual equipment requirements in ResourceConstraintSpec (*ResourceConstraintType*) elements. Each ResourceConstraint element contains a requirement ID, one or more descriptions, and an optional range of the requirement.

Note: The term **Resource Constraint** is used to minimize the opportunity for namespace conflicts with the B2MML equipment element definitions.

Resource Constraint Elements

Constraint ID	Description	Range
Stainless Steel Vessel	Vessel and all associated piping are stainless steel	True, False
Personnel Protection Class 3	Personnel required to have protection (gloves, hoods, gowns) as required for Class 3 environments.	True, False
High Shear Blending	Equipment must be able to blend with high shearing force to break apart internal material composition,	True, False
Environment Humidity	Maximum humidity if product or raw materials exposed to environment.	0% - 100%

1.9 Process Element Library

From ANSI/ISA 88.03, Clause 5.5.10.2 – "*Process action definitions should be maintained in a process action library in order to ensure that only accepted and broadly understood process action definitions are used in equipment independent recipes. In this context, the term* library *is used to indicate the collection of available process action definitions, and not meant to imply any specific storage or management mechanism.*"

Process Actions and other process elements are defined in a Process Element Library (*ProcessElementLibrary*). A library element contains a set of Process Elements specifications. The specification may be a Process Action, Process Operation, or Process Stage. Normally there would be at least one library of Process Actions that are used to construct General Recipes.

1.10 Material Definition Library

The material specifications in a General or Site Recipe identify a material, using either a Material Definition or Material Class (as defined in ANSI/ISA 95.01). The General and Site Recipes only contain an ID. The Material Definition Library should be exchanged using the B2MML-Material schemas.

1.11 Recipes

A Recipe (*GRecipeType*) may be a General Recipe or a Site Recipe, specified in the RecipeType (*GRecipeTypeType*) element. Each recipe has an identification and version, specified in the ID element, a description, and a lifecycle state. A Recipe also contains the following elements:

1.11.1   Header

The recipe header information is contained in the Header (*GRecipeHeaderType*) element. This contains the information used to identify the recipe and the product(s) produced by the recipe.

1.11.2   Formula

The Formula (*GRecipeFormula*) contains a list of material specifications that define input materials, output materials, and intermediate materials.

1.11.3   Procedure

The recipe's Procedure (ProcessElementType) is a Process Element Type, with ElementType set to "Process".   See the section on Process Elements.

1.11.4   Resource Constraints

The Resource Constraints (*ResourceConstraint*) define specific constraints for the entire recipe. Each ResourceConstraint element defines the ConstraintID (should match a ConstraintID from the Resource Constraint Library), an optional description of the instance of use of the equipment requirement, and the range of acceptable values for the requirement. For example:

Recipe – Resource Constraint Elements

Constraint ID	Description	Range
Stainless Steel Vessel	Vessel and all associated piping are stainless steel	True
Personnel Protection Class 3	Personnel required to have protection (gloves, hoods, gowns) as required for Class 3 environments.	False
High Shear Blending	Equipment must be able to blend with high shearing force to break apart internal material composition,	False
Environment Humidity	Maximum humidity if product or raw materials exposed to environment.	30% - 50%

1.11.5   Other Information

From ANSI/ISA 88.03 Clause 5.7 – "*A general recipe is a container of production information required for manufacturing, including the process definition, material identification and amounts, material quality information, and references to test definitions and test standards.   Examples of other information often included with equipment independent recipes are:*

— *Spreadsheets detailing known process sensitivity models*

— *Complete process models*

— *Pictures of good products*

— *Pictures of bad products and possible failure reasons*

— *References to Test Methods & Test Specifications*

— *References to material data safety sheets*

— *Additional health and safety information*

— *Packaging information*"

The OtherInformation (GR*OtherInformationType*) elements contain an OtherValue (*ValueType*) element that can be used to hold any type of information (Extended:Value type).   This is illustrated in Figure 3.
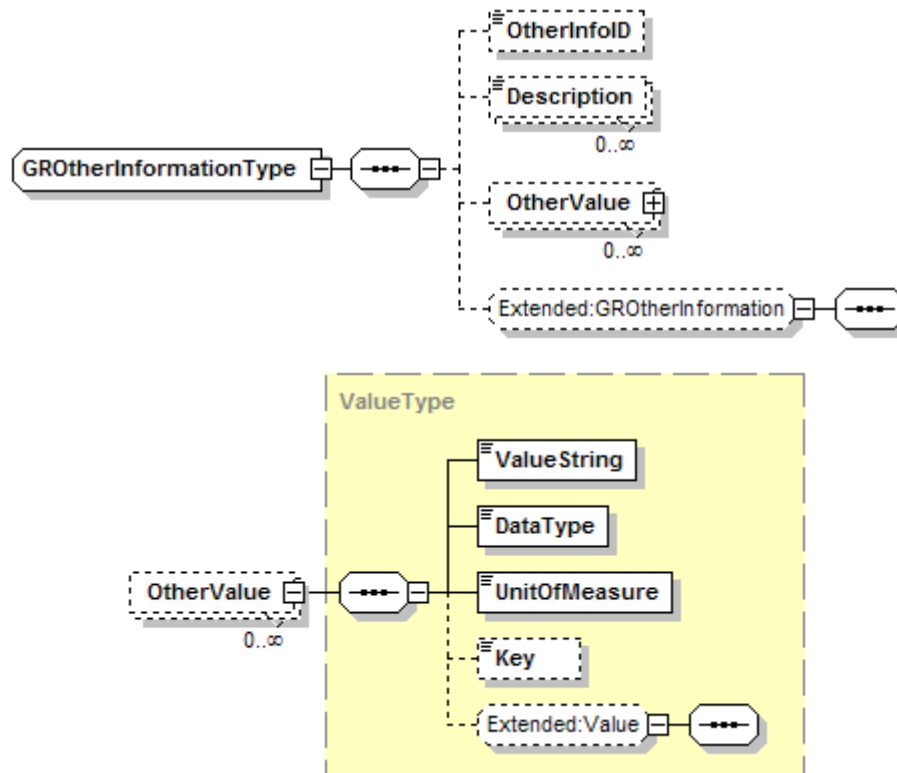

Figure 3 – Other Information Structure

1.12 Process Element

A ProcessElement (GProcessElementType) is used to hold information about a Process, Process Stage, Process Operation, or Process Action.

A Process Element contains an optional identification and an element type. The information in the Process Element is based on the ElementType attribute:

- **Process** – Indicates the Recipe's Procedure (ANSI/ISA 88.03 Clause 5.5.2).

  The ID is generally optional, the ProcessElement typically contains Materials elements, DirectedLink elements, Procedure Chart elements, and Process Stage elements. See Section 3 for an example.

- **Process Stage** – Indicates a Process Stage (ANSI/ISA 88.03 Clause 5.5.4).

  The ID should identify the stage. The ProcessElement typically contains Materials elements, DirectedLink elements, Procedure Chart elements, EquipmentReq, and Process Operations elements.   See Section 3 for an example.

- **Process Operation** – Indicates a Process Operation (ANSI/ISA 88.03 Clause 5.5.5).

  The ID should identify the operation. The ProcessElement typically contains Materials elements, DirectedLink elements, Procedure Chart elements, EquipmentReq, and Process Action elements. See Section 3 for an example.

- **Process Action** – Indicates a Process Action (ANSI/ISA 88.03 Clause 5.5.6).

  The ID should identify the specific instance of the process action (for example: using the library name and appended with a ":" and a number, such as "Charge to Adjust pH:2". The ProcessElement typically contains Parameters, EquipmentReq, and Other Information. See Section 3 for an example.

Process Elements in a Process Element Library may have a LifeCycle State.

1.13 Procedure Chart Element

Elements in a Process, Process Stage, and Process Operation which are not material, links, or process elements may be process procedure chart elements (as defined in ANSI/ISA 88.03 Clause 7). These are elements used to support the graphical notation defined in the standard.

- **Previous Operation Indicator** – Indicates a placeholder that on Process Operation and Process Stage diagrams that indicate a previous operation. Usually used to link to an Intermediate Material indication. Typically only contains an ID element (to allow identification in a DirectedLink element).

  Note: This type is not specifically defined in ANSI/ISA 88.03, but is implied in the examples.

- **Next Operation Indicator** – Indicates a placeholder that on Process Operation and Process Stage diagrams that indicate a previous operation. Usually used to link to an Intermediate Material indication. Typically only contains an ID element (to allow identification in a DirectedLink element).

  Note: This type is not defined in ANSI/ISA 88.03, but is implied in the examples.

- **Start Parallel Indicator** – Indicates the start of a set of parallel stages, operations, or actions. As defined in ANSI/ISA 88.03 Clause 7.2.3.2.   Typically only contains an ID element (to allow identification in a DirectedLink element).

- **End Parallel Indicator** – Indicates the end of a set of parallel stages, operations, or actions. As defined in ANSI/ISA 88.03 Clause 7.2.3.3.   Typically only contains an ID element (to allow identification in a DirectedLink element).

- **Start Optional Parallel Indicator** – Indicates the start of a set of optional parallel stages, operations, or actions. As defined in ANSI/ISA 88.03 Clause 7.2.3.4.   Typically only contains an ID element (to allow identification in a DirectedLink element).

- **End Optional Parallel Indicator** – Indicates the start of a set of parallel stages, operations, or actions. As defined in ANSI/ISA 88.03 Clause 7.2.3.5.   Typically only contains an ID element (to allow identification in a DirectedLink element).

- **Annotation** – Indicates an annotation to the process, stage, or operation. As defined in ANSI/ISA 88.03 Clause 7.2.3.6.   Typically only contains an ID element and the annotation in the Description element.

1.14 Directed Links

A Directed Line (*DirectedLinkType*) is used to represent a material flow dependency or time dependency between materials process elements, and procedure chart elements.   Basically a DirectedLink is any line

with an arrowhead on a Process Procedure Chart (PPC) diagram. Figure 4 illustrates the directed links in a process operation diagram.
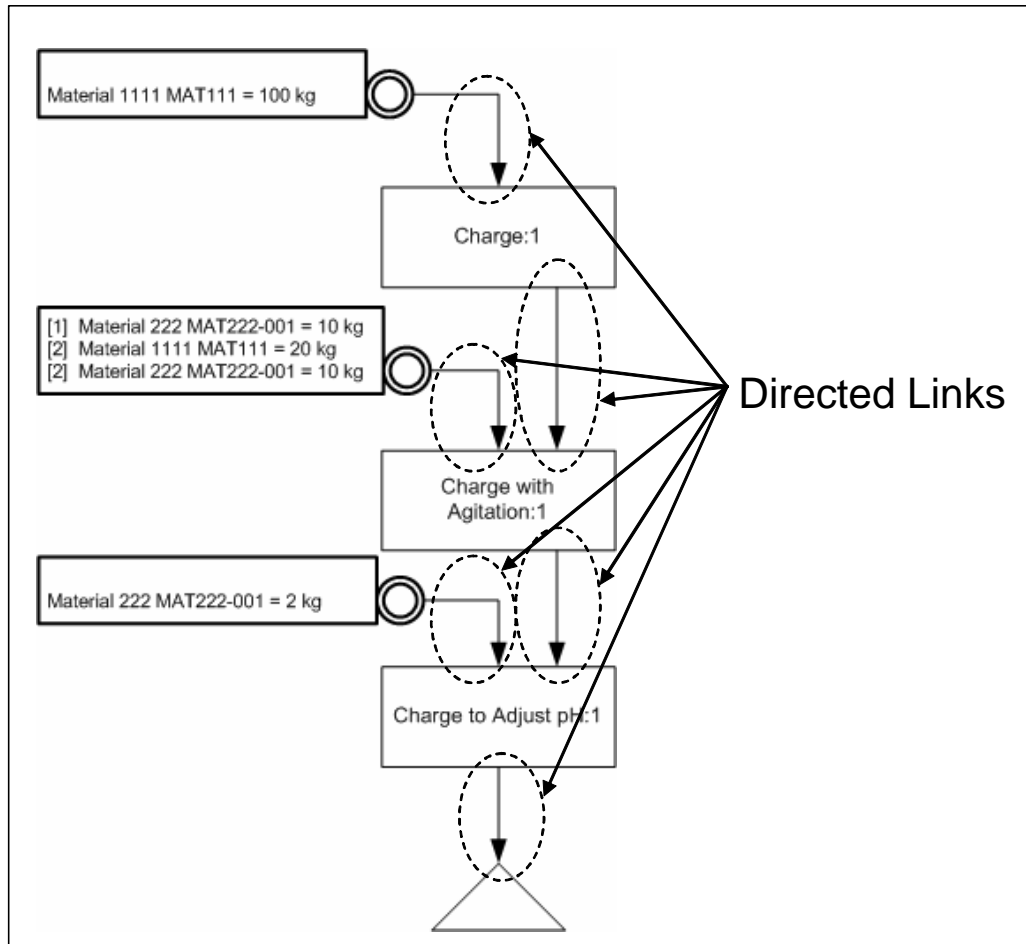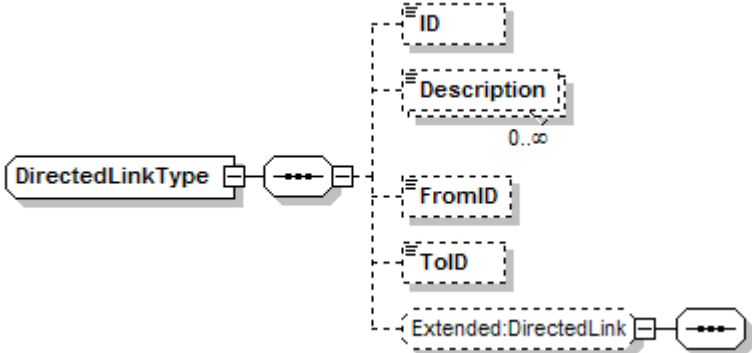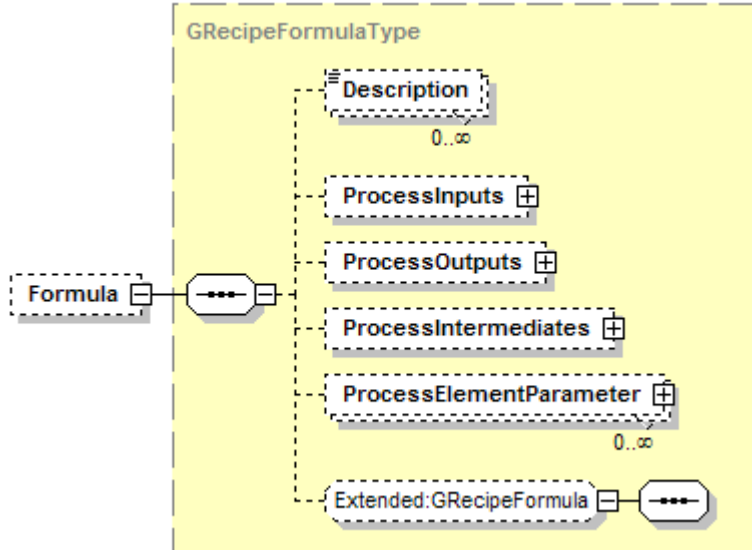

Figure 4 – Directed Links in a PPC
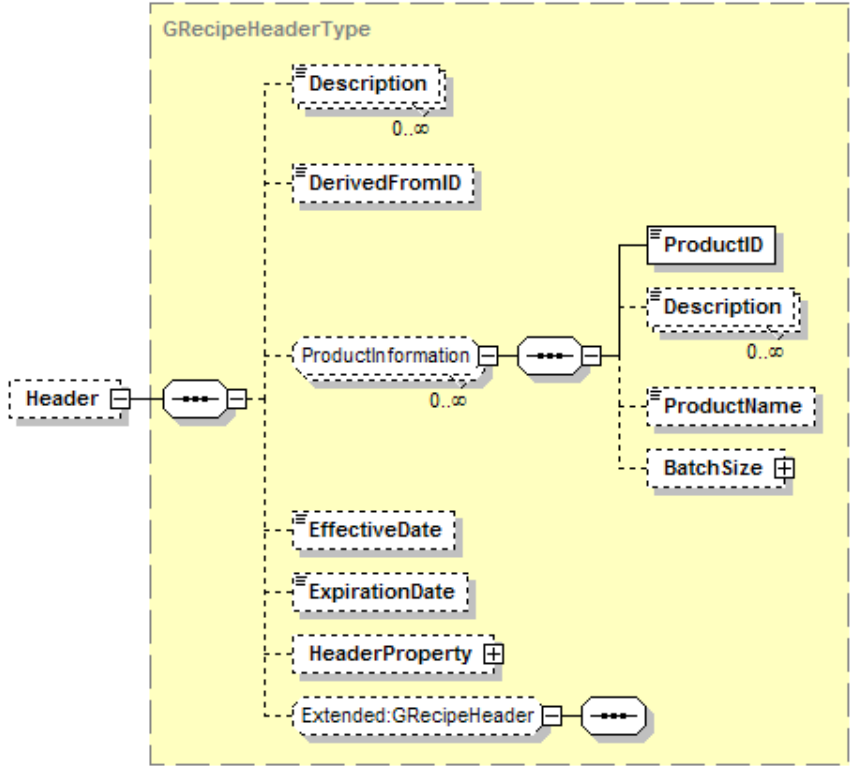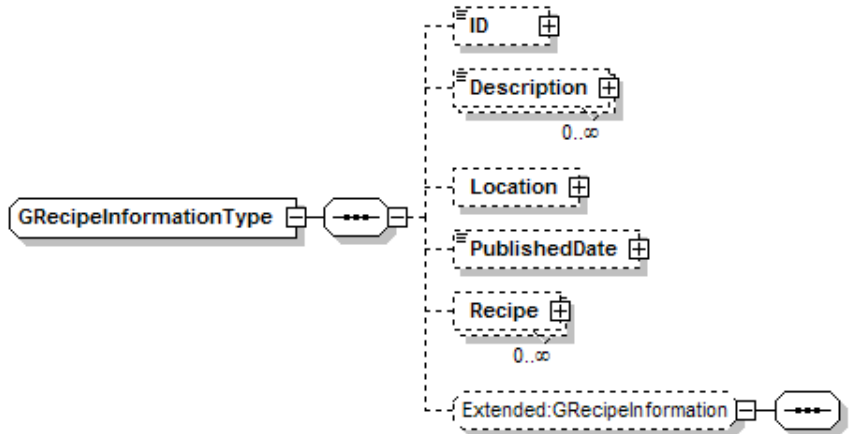
1.15 Materials

Materials (*GRecipeMaterialsType*) is a container object for Material (*GRecipeMaterialType*) elements. The ANSI/ISA standard allows multiple materials to be specified for an input, output, or intermediate material indication. Additionally a recipes Formula (*GRecipeFormulaType*) contains three subsets of materials.

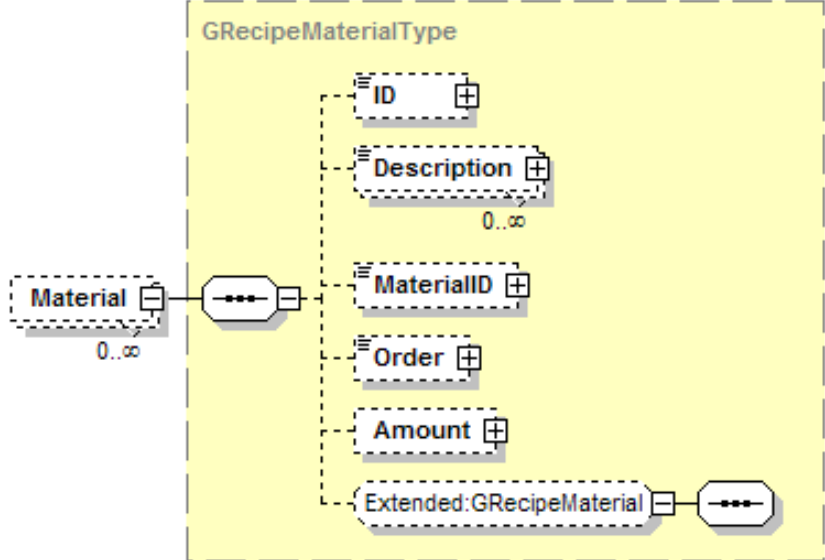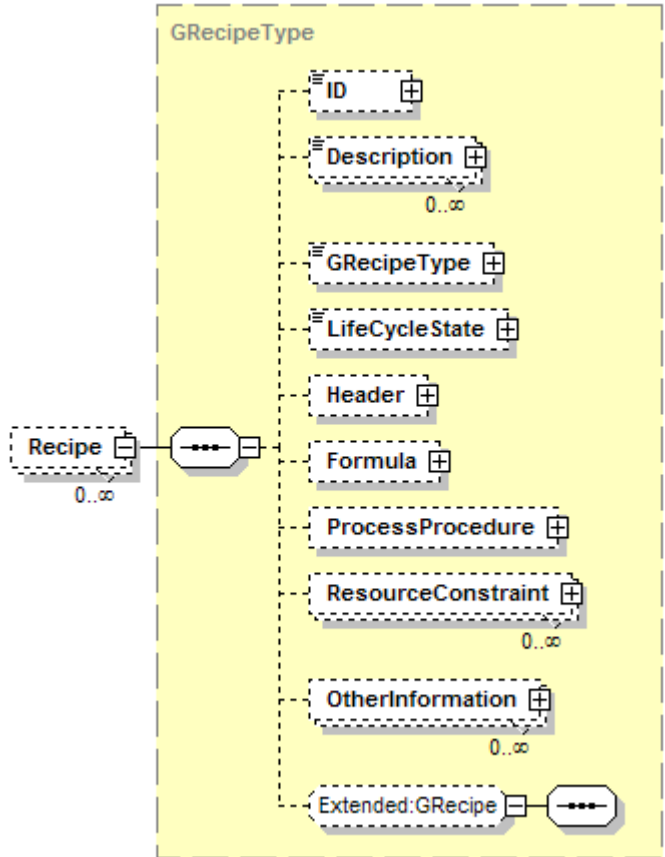The GRecipeMaterialsType is used to identify a set of materials.
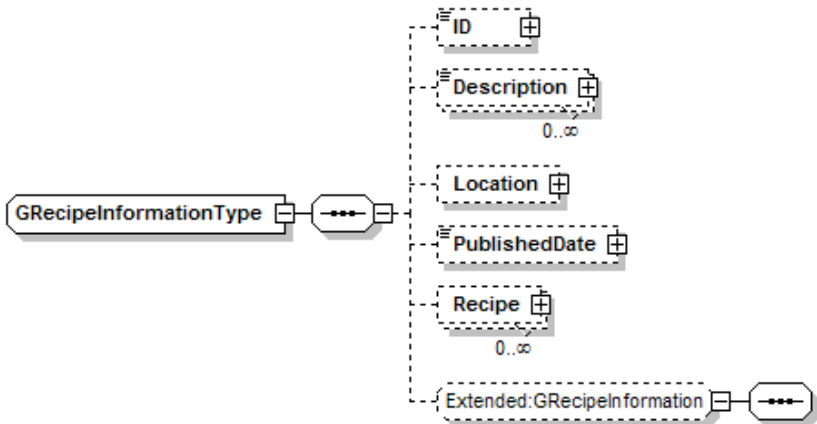
2 Element Definitions

Element/Type	Description
ConstraintTypeType ConstraintType	Identifies the type of a Resource Constraint in a Process Element or Recipe. This may be a standard type or an application specific type. Standard enumerations are: "**Required**", "**Optional**", and '**Other**". • **Required à** Indicates that the requirement is required for proper operation of the Process Element or Recipe • **Optional à** Indicates that the requirement is optional in the Process Element or Recipe If "Other" then the type is an application specific extension and the value is defined in the attribute "OtherValue".
DirectedLinkType DirectedLink	A Directed Link defines a link between elements of a Process Element, linking Materials, Process Elements, and ProcedureChartElements. See the General Recipe Example section. A link contains the ID of the **From** side of the link (the tail of the arrow) and the **To** side of the link (the head of the arrow).

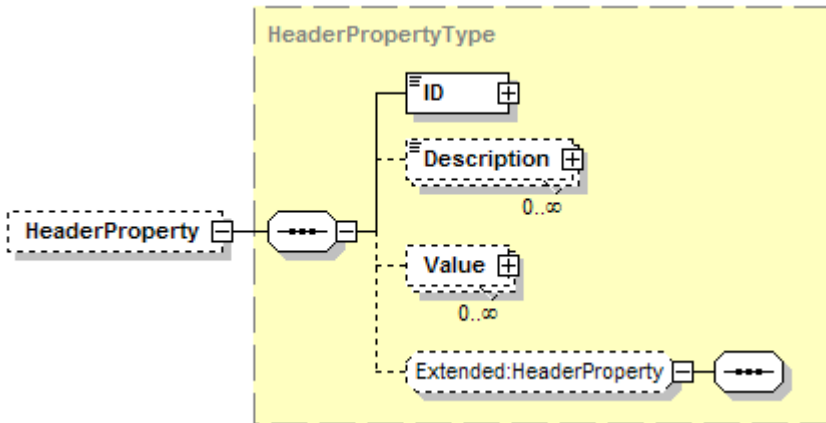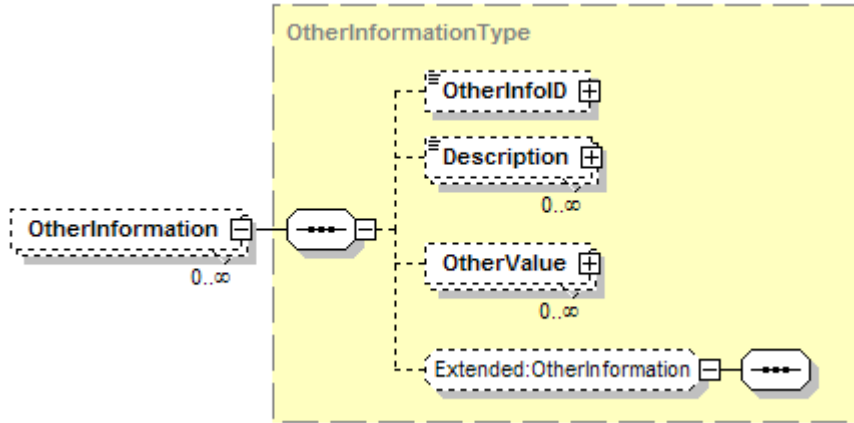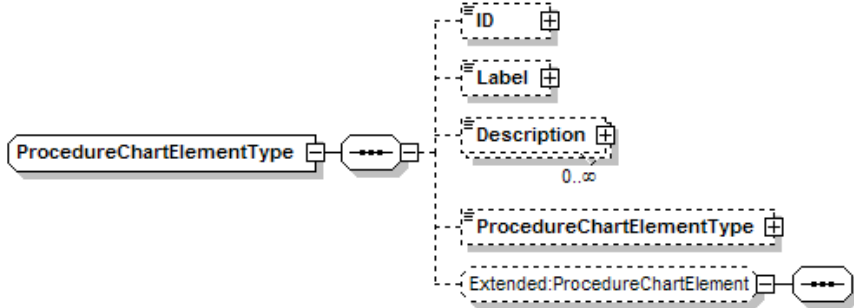| Element/Type	Description
GParameterTypeType ParameterType	Identifies the type of a parameter. (Note: Multiple ParameterTypes may be defined for a single Parameter) This may be either a standard type or an application specific extended type. Standard enumerations are: "**Input**", "**Output**", "**Optional**", "**Required**", and "**Other**". • **Input à** The parameter contains a value used by the associated Process Element during execution. • **Output à** The parameter is to be reported back by the associated Process Element during (or after) execution. • **Optional à** The parameter is optional. • **Required à** The parameter is required. If "**Other**" then the type is an application specific extension and the value is defined in the attribute "OtherValue". For example: • Other à With OtherValue = "**CPP**", an application specific type that indicates that the parameter is a Critical Process Parameter, registered with regulatory authorities. • Other à With OtherValue = "**KQA**", and application specific type that indicates that the parameter defines an Key Quality Attribute of the produced product, rather than a process parameter.
GRecipeFormulaType Formula	A Formula is part of a recipe and contains a set of Process Input (*GRecipeMaterialsType*), Process Outputs (*GRecipeMaterialsType*),Process Intermediates (*GRecipeMaterialsType*), and process parameters.

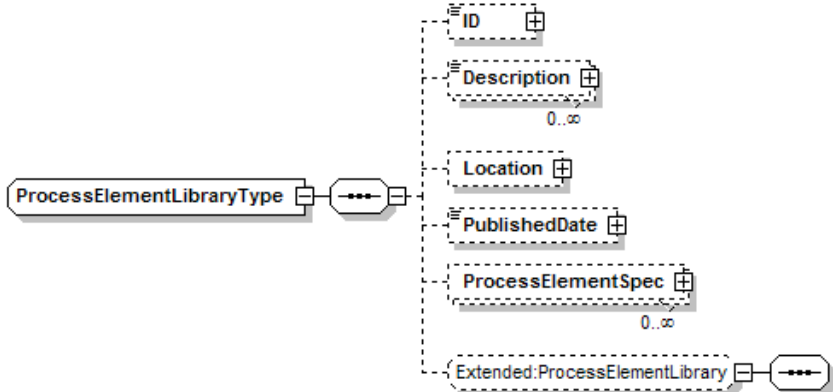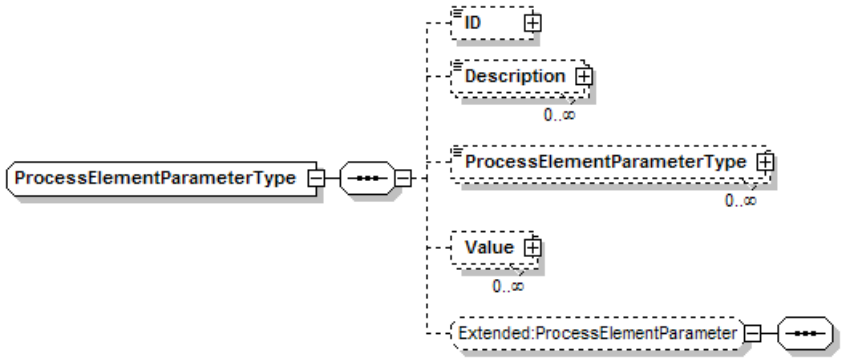| Element/Type	Description
GRecipeHeaderType Header	A Header Element is part of a recipe and contains the identification information for the recipe. It includes an ID of the recipe this recipe was derived from (e.g. a Site Recipe may be derived from a General Recipe). It contains zero or more product identification, name, and size sets (for those cases where a recipe may generate multiple products), effective date of the recipe, an expiration date of the recipe, and a set of properties.
GRecipeInformationType GRecipeInformation	GRecipeInformation is a top level container object that contains a set of General or Site Recipes. The GRecipeInformation contains an identification of the XML document, description, Location (LocationType in B2MML-Common, defines the scope of the definitions – e.g. a Specific Site), and the published date of the XML document.

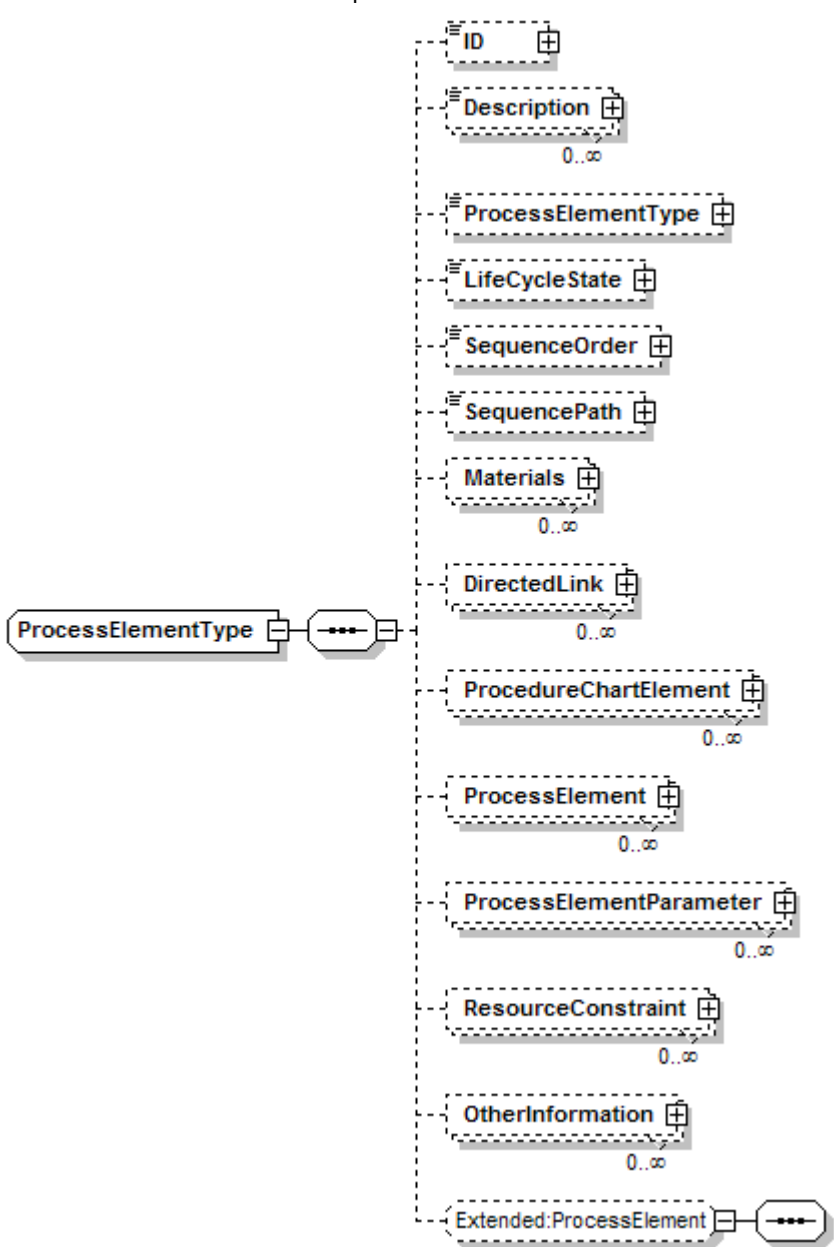| Element/Type	Description
GRecipeMaterialsType Materials ProcessInputs ProcessOutputs ProcessIntermediates	A *GRecipeMaterialsType* is used for Process Inputs, Process Outputs, and Materials (in a Process Element) to define a set of materials. Each *GRecipeMaterialType* element contains an optional identification, a definition of the use of the materials (*MaterialsTypeType*), and a set of Material (*GRecipeMaterialType*) definitions.
GRecipeMaterialType Material	A *GRecipeMaterialType* contains a specification of a Material Definition or Material Class (from the ANSI/ISA 95 and ANSI/ISA 88.03 standards) in a formula Process Input, Output or parameter. . A *GRecipeMaterialType* contains and optional ID of the XML element, an ID of the material (Material Definition ID or Material Class ID), an optional Order element (see the example in Section 4), and an Amount (*ValueType* from B2MML-Common).

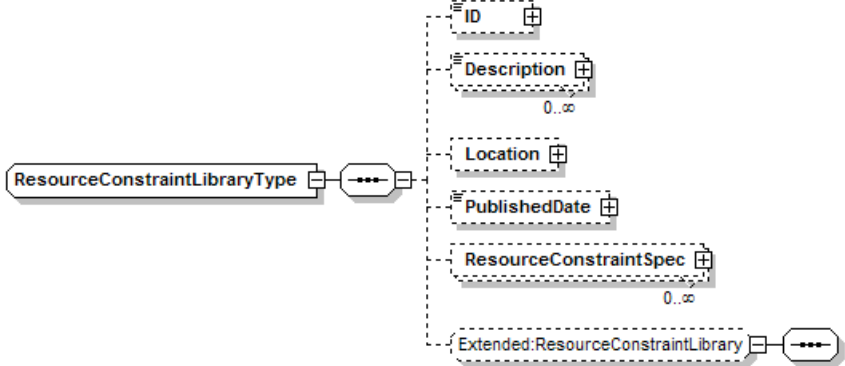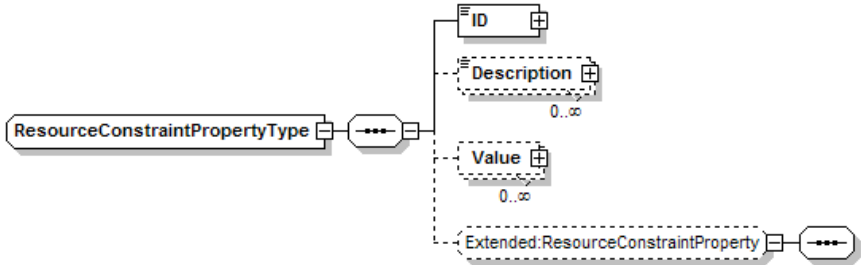| Element/Type	Description
GRecipeType Recipe	A GReipeType defines a General or Site Recipe. See descriptions above.
GRecipeTypeType RecipeType	Identifies the type of a Recipe. This may be a standard type or an application specific type. Standard enumerations are: "**General**", "**Site**", and '**Other**". • **General à** Indicates that the recipe is a General Recipe. • **Site à** Indicates that the recipe is a Site Recipe. If "Other" then the type is an application specific extension and the value is defined in the attribute "OtherValue".

Element/Type	Description
HeaderPropertyType HeaderProperty	Contains a definition of a recipe property, consisting of an ID, a description, and a value. This element provides a method for defining and documenting project and company specific recipe properties without requiring changes to the schemas.
LifeCycleStateType LifeCycleState	Identifies the lifecycle state of a Recipe, Process Element Specification, and Resource Constraint. These types are defined in ANSI/ISA 88.03, Clause 5.8. This may be a standard type or an application specific type. Standard enumerations are: "**Draft**", "**Approved**", "**Released**", "**Effective**", "**Withdrawn**", and '**Other**". • **Draft à** Indicates that the element definition is under development or is available for review but is not yet available for use in normal production. • **Approved à** Indicates that the element definition is complete and has been approved by all pertinent authorities. • **Released à** Indicates that the element definition has been approved and has been distributed, but it has not yet become effective. • **Effective à** Indicates that the element definition is available for use. • **Withdrawn à** Indicates that the element definition is no longer effective and is not available for use. If "**Other**" then the type is an application specific extension and the value is defined in the attribute "OtherValue".
MaterialsTypeType MaterialsType	Identifies the use of a set of materials. This may be either a standard type or an application specific extended type. Standard enumerations are: "**Input**", "**Output**", "**Intermediate**", or "**Other**". • **Input à** Material that is consumed by the process. • **Output à** Material produced as a result of execution of the process. • **Intermediate à** Material that is produced as a result of the process and is later consumed by the process. If "**Other**" then the type is an application specific extension and the value is defined in the attribute "OtherValue".

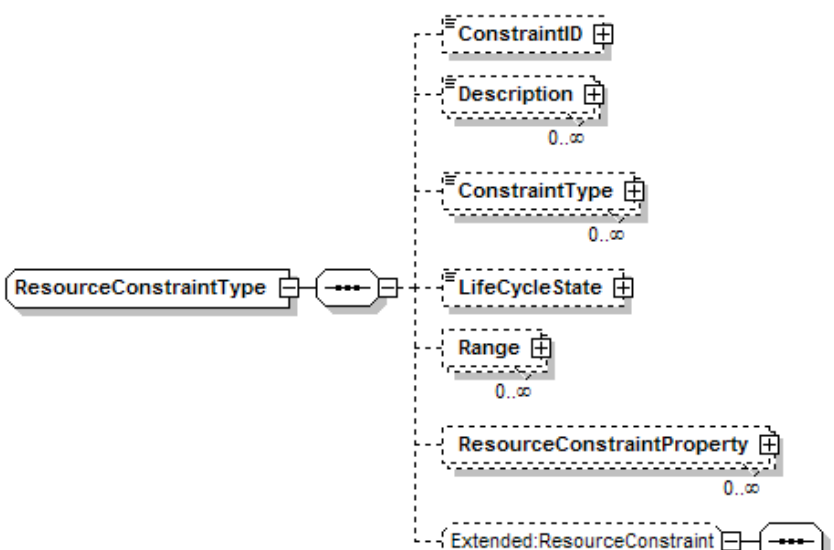| Element/Type	Description
OtherInformationType OtherInformation	OtherInformation is a container for other information associated with a Process Element or Recipe. It contains an identification (OtherInfoID), zero or more descriptions, and zero or more pieces of data (OtherValue of type *ValueType*).
ProcedureChartElementType ProcedureChartElement	A Procedure Chart Element defines a Process Procedure Chart graphical element, such as a start of parallel or annotation. Each Procedure Chart Element contains an ID (used to identify the element in a Link element), an optional Label (TextType) for the annotation, a description, and the type of the element (*ProcedureChartElementTypeType*).

Element/Type	Description
ProcedureChartElementType ProcedureChartElementType	Identifies the type of a Procedure Chart Element. This may be a standard type or an application specific type. Standard enumerations are: "**Previous Operation Indicator**", "**Next Operation Indicator**", "**Start Parallel Indicator**", "**End Parallel Indicator**", "S**tart Optional Parallel Indicator**", "**End Optional Parallel Indicator**", "**Annotation**", and 'Other". • **Previous Operation Indicator** à Not specifically defined in ISA 88.03, but is a symbol used to indicate material or control flow from a previous operation on another diagram. • **Next Operation Indicator** à Not specifically defined in ISA 88.03, but is a symbol used to indicate material or control flow to the next operation on another diagram. • **Start Parallel Indicator** à Defined in ISA 88.03 Clause 7.2.3.2, indicates a start of parallel Process Elements. • **End Parallel Indicator** à Defined in ISA 88.03 Clause 7.2.3.3 , indicates an end to parallel Process Elements. • **Start Optional Parallel Indicator** à Defined in ISA 88.03 Clause 7.2.3.4, indicates a start of optionally parallel Process Elements. • **End Optional Parallel Indicator** à Defined in ISA 88.03 Clause 7.2.3.5, indicates an end to optionally parallel Process Elements. • **Annotation** à Defined in ISA 88.03 Clause 7.2.3.6, indicates a comment on a process procedure chart. If "**Other**" then the type is an application specific extension and the value is defined in the attribute "OtherValue".
ProcessElementLibraryType ProcessElementLibrary	An ProcessElementLibrary defines a set of Process Element Specifications (ProcessElementSpec as *ProcessElementType*). It contains an identification of the library (e.g. ID and version) and zero or more Descriptions

Element/Type	Description
ProcessElementParameterType ProcessElementParameter	A *ProcessElementParameterType* defines either a specification of a parameter (in a Process Element Library), or an instance of the use of a parameter in a Process Element in a Recipe. The parameter contains an identification of the parameter (ID), an optional description, zero or more a ProcessElementParameterType (*ProcessElementParameterTypeType*) elements, and zero of more Value elements. (Note: Multiple ProcessElementParameterTypes may be defined for a single Parameter)

Element/Type	Description
ProcessElementType ProcessElement Procedure	A *ProcessElementType* defines a procedure, process stage, process operation, or process action. See details in the previous section.

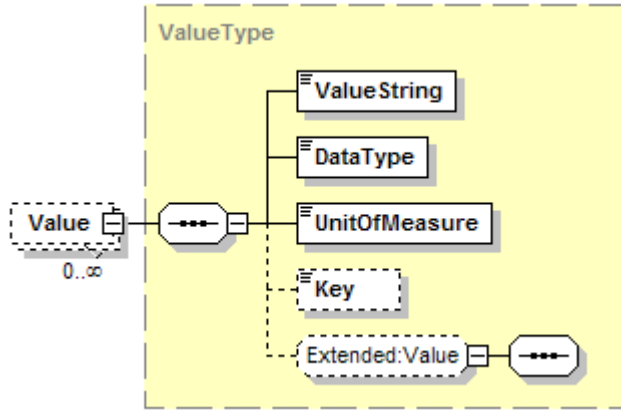| Element/Type	Description
ProcessElementTypeType ProcessElementType	Identifies the type of a Process Element This may be either a standard type or an application specific extended type. Standard enumerations are: "**Process**", "**Process Stage**", "**Process Operation**", "**Process Action**" and "**Other**" • **Process** à Indicates that the Process Element is a Process Description and usually contains Process Stages. • **Process Stage** à Indicates that the Process Element is a Process Stage and usually contains Process Operations. • "**Process Operation** à Indicates that the Process Element is a Process Operation and usually contains Process Actions. • **Process Action** à Indicates that the Process Element is a Process Action and usually contains Parameters. If "**Other**" then the type is an application specific extension and the value is defined in the attribute "OtherValue".
ResourceConstaintLibraryType ResourceConstraintLibrary	A Resource Constraint Library defines a set of Resource Constraint Specs (ResourceConstraintSpec as *ResourceConstraintType*). It contains an identification of the library (e.g. ID and version) and zero or more Descriptions.
ResourceConstraintProperty ResourceConstraintProperty	Contains a definition of Resource Constraint property, consisting of an ID, a description, and a value. This element provides a method for defining and documenting project and company specific Resource Constraint without requiring changes to the schemas.

Element/Type	Description
ResourceConstraintType ResourceConstraint ResourceConstraint Spec	An ResourceConstraint defines an Resource Constraint (Equipment Requirement) to be applied to the associated Recipe, or Process Element, or an Resource Constraint defined in Resource Constraint Library. An Resource Constraint contains an identification (*ConstraintID*), zero or more Descriptions, ConstraintType (*ConstraintType*), LifeCycle State, and Range (*ValueType*) of the requirement. • In a library element the Range is the valid range (e.g. True, False or 0%-100%). • In a Process Element the Range is the specific range for that instance of the requirement (e.g. True or 5%-10%).

Element/Type	Description
SequenceOrderType SequenceOrder	Identifies the type of a process element in the execution sequence. The order of execution in a table format is from top to bottom, unless modified by the sequence order and sequence path value. The sequence order indicates sequential execution, parallel branches, or sequential execution under parallel branches. Note: The use of symbols within the table format allows simple parallel constructs to be quickly interpreted. Table representations should not be used for complex sequences of parallels and nested parallels. This may be a standard type or an application specific type. Standard enumerations are: "**First Parallel**", "**Middle Parallel**", "**End Parallel**", "**First In Path**", "**Last In Path**", "**Not In Parallel**", "**Single First Parallel**", "**Single Middle Parallel**", "**Single End Parallel**", and "**Other**". • **First Parallel** à Indicates that the element is first in first series under a parallel (first action in first path) • **Middle Parallel** à Indicates that the element is action in middle of series under a parallel (not first or last in path) • **End Parallel** à Indicates that the element is last in last series of actions under a parallel (last action in last path) • **First In Path** à Indicates that the element is first in a series of actions under a parallel (first action in path) • **Last In Path** à Indicates that the element is last of series of actions under a parallel (last action in path) • **Not In Parallel** à Indicates that the element is an action not under a parallel • **Single First Parallel** à Indicates that the element is a single action at start of a parallel (only action in path) • **Single Middle Parallel** à Indicates that the element is a single action under a parallel (only action in path) • **Single End Parallel** à Indicates that the element is a single action at end of parallel (only action in path) If "**Other**" then the type is an application specific extension and the value is defined in the attribute "OtherValue".
SequencePath	A number, that indicates the sequence within a set of parallel sequences. Sequences should be numbered sequentially, starting from the left. All actions under the same sequence path execute sequentially.

Element/Type	Description
ValueType Amount OtherValue Range Value	From the B2MML-Common schema. A ValueType contains a value (in ValueString), the datatype, and unit of measure. A Key element is defined for identification of where multiple ValueTpes are used, such as in a Range where one Key Value may be "High Limit" and another would be "Low Limit".

3 General Recipe Example

The following diagrams illustrate selected parts of an example General Recipe.

3.1 Process Example

Figure 5 illustrates the process within a general recipe using a Process Procedure Chart (PPC).
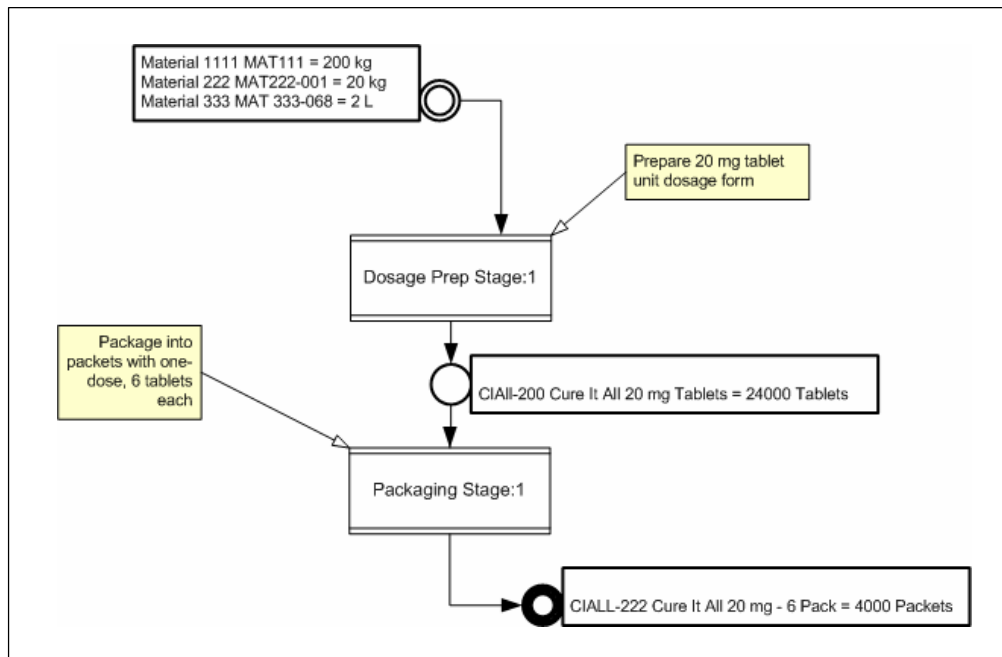
Figure 5 – Sample Process Procedure Chart

3.2 Elements of the Process

Figure 6 illustrates the materials, link, annotation, and child Process Element that make up the Process description. There would also be elements for the Dosage Prep Stage:1 and Packaging Stage:1.
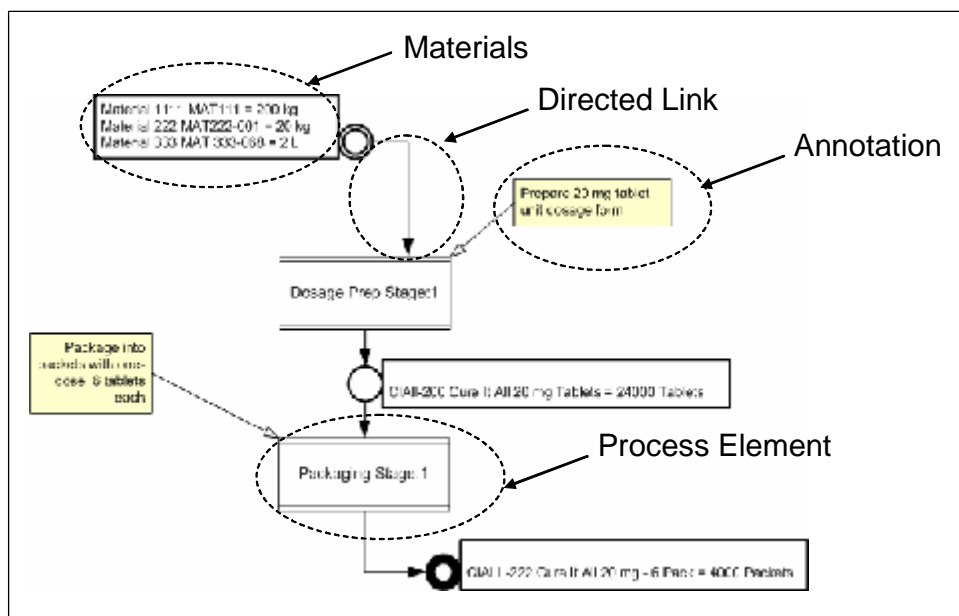
Figure 6 – Elements of a Process and PPC

3.3 Process Stage

Figure 7 illustrates the "Dosage Prep Stage" with optionally parallel operations that precede the Tableting Operation:1
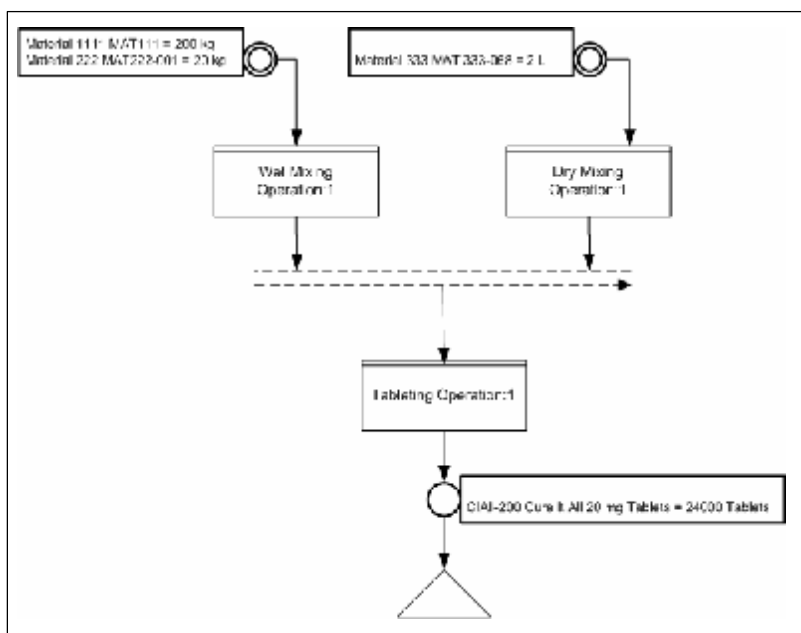
Figure 7 – Sample Dosage Prep Stage

3.4 Elements of the Process Stage

Figure 8 illustrates the elements of the Process Stage, including the links, End Parallel Indicator and Next Operation Indicator.
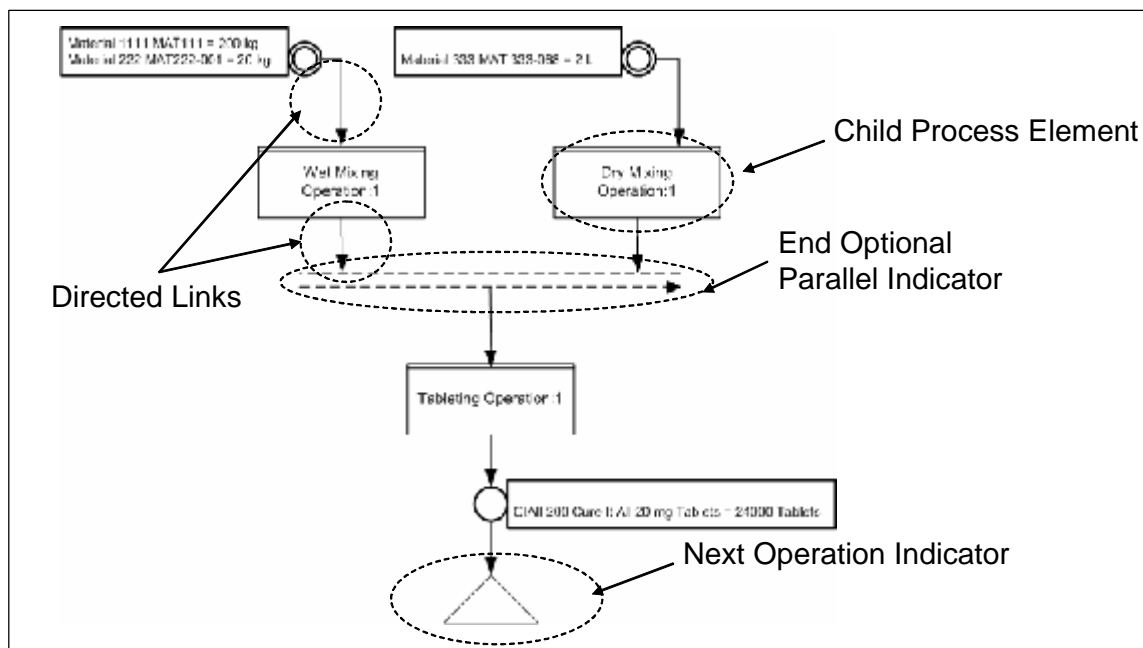
Figure 8 - Elements of Dosage Prep Stage

3.5 Process Operation

Figure 9 illustrates the three Process Actions that make up the Wet Mixing Operation.
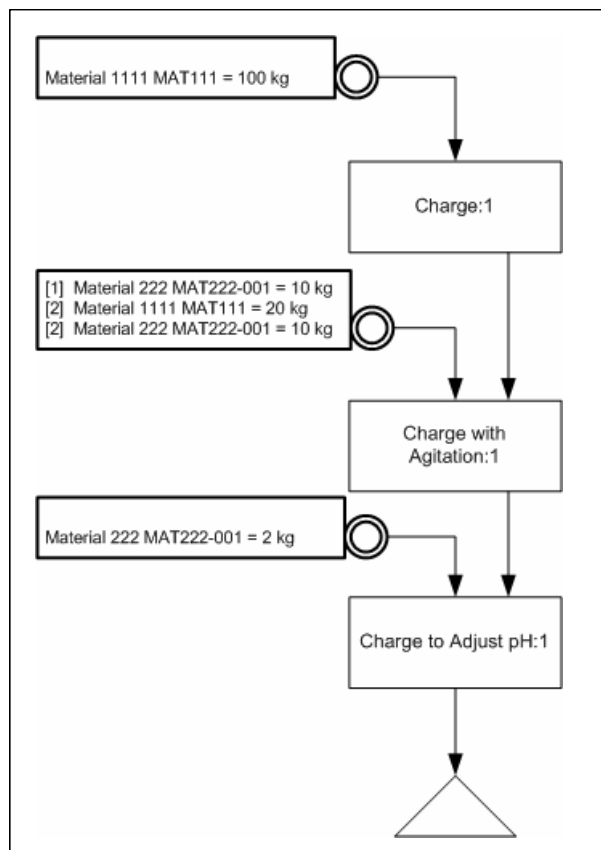
Figure 9 – Sample Wet Mixing Operation

3.6 Elements of the Process Operation

Figure 10 illustrates some of the elements within the Wet Mixing Operation, illustrating the links and the materials (not highlighted are the three Process Actions and the Next Operation Indicator).

Figure 10 - Elements of the Process Operation

3.7 XML Representation

The previous figure could be represented as the following XML snippet:

```
…
<ProcessElement>
   <ID "Wet Mixing Operation" />
   <ElementType "Process Operation" />
   <Materials>
      <ID "M20045" />  <MaterialType "Input" /> . . .
   </Materials>
   <Materials>
      <ID M20046 />     <MaterialType "Input" /> . . .
   </Materials>
   <Materials>
      <ID M20047 />     <MaterialType "Input" /> . . .
   </Materials>
   <DirectedLink>
     <FromID "M20045" />
     <ToID   "Charge:1" />
   </DirectedLink >
   <DirectedLink >
      <FromID "Charge:1" />
      <ToID   "Charge with Agitation:1" />
   </DirectedLink >
   <DirectedLink >
      <FromID "M20046" />
      <ToID   "Charge with Agitation:1" />
   </DirectedLink >
   <DirectedLink >
      <FromID "Charge with Agitation:1" />
      <ToID "Charge to Adjust pH:1" />
   </DirectedLink >
   . . .
   <ProcessElement>
      <ID "Charge:1" />
      <ElementType "Process Action" />
      ...
   </ProcessElement>
   <ProcessElement>
      <ID "Charge with Agitation:1" />
      <ElementType "Process Action" />
      . . .
   </ProcessElement>
   <ProcessElement>
      <ID "Charge to Adjust pH:1" />
      <ElementType "Process Action" />
      . . .
   </ProcessElement>
   . . .
</ProcessElement>
…
```

3.8 Materials and Material Elements

The second *Materials* indication in Figure 10 shows multiple materials. The "Order" element in the GMaterialType to indicate the specific order that materials are to be used, and is illustrated in Figure 11. Each GMaterialType also includes amount of material to be used.
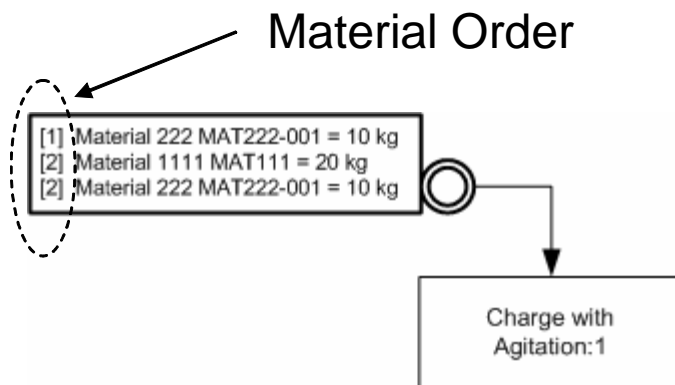
Figure 11 - Multiple Material Element

A use of the "Order" element is as follows:

- When there is no number order element specified, then the order of material addition is not critical to the process chemistry or physics, and may be optimized at the site level. For example materials may be added simultaneously or layered based on optimization of later blending. For example, the following list of materials have no specified order of addition.
    - o   Water
    - o   Sucrose
    - o   Lemon Extract
- When an order is specified, then the order defines the order of addition. For example, the following indicates that water is charged first and then Sucrose and then Lemon Extract,
    - o   [1] Water
    - o   [2] Sucrose
    - o   [3] Lemon Extract
- When the order number is the same for multiple materials, then the materials should be added simultaneously. For example the following indicates that Water and Sucrose are simultaneously added.
    - o   [1]: Water
    - o   [1]: Sucrose
    - o   [2]: Lemon Extract
- If the same material is added multiple times in the sequence, then the amount or percentage of material added per order number may be specified with each use of the material, as shown in the example below.

- o   [1] Water – 70%
- o   [2] Sucrose
- o   [3] Water – 30%
- o   [4] Lemon Extract

3.9 XML Representation of Materials

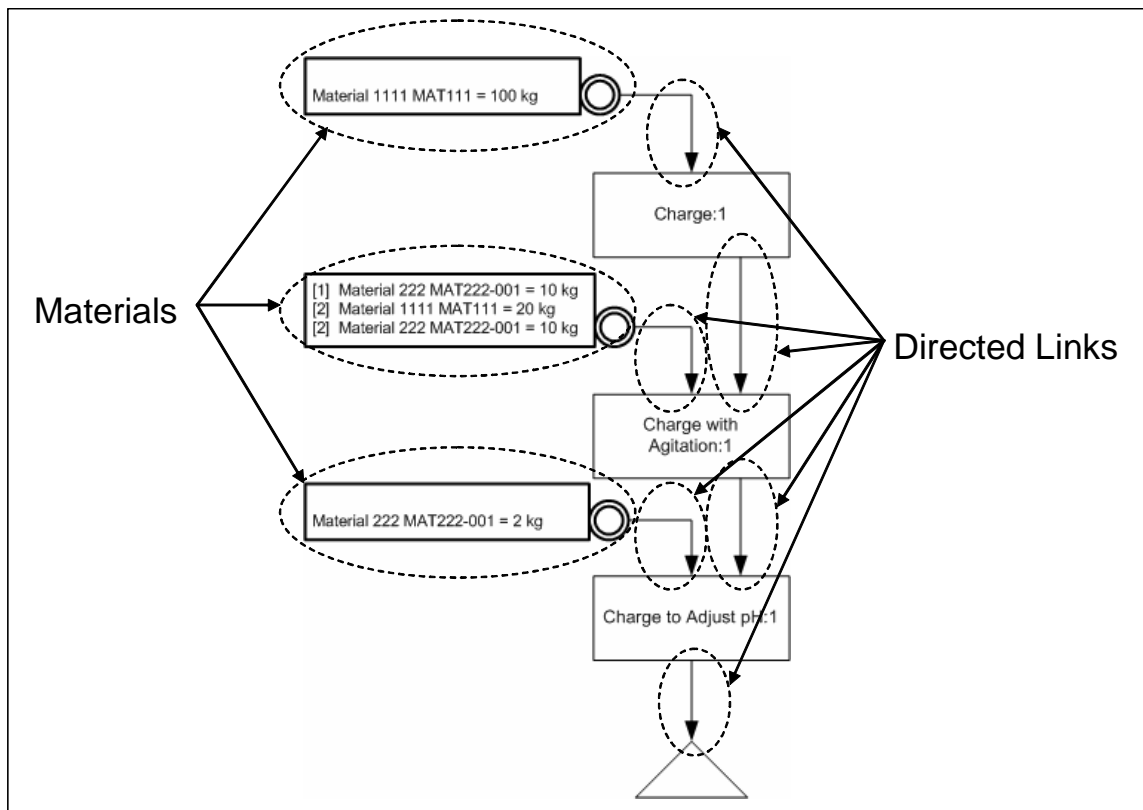The previous figure could be represented as the following XML snippet:

```
 . . .
<Materials>
   <ID M20045 />
   <MaterialType "Input" />
   <Material>
      <ID "Material 222" />
      <MaterialID "MAT222-01" />
      <Order "1" />
      <Amount>
         <ValueString "10"/> <DataType "Measure"/> <UnitofMeasure "KG"/>
      </Amount>
   </Material>
   <Material>
      <ID "Material 1111" />
      <MaterialID "MAT111" />
      <Order "1" />
      <Amount>
         <ValueString "20"/> <DataType "Measure"/> <UnitofMeasure "KG"/>
      </Amount>
   </Material>
   <Material>
      <ID "Material 222" />
      <MaterialID "MAT222-01" />
      <Order "2" />
      <Amount>
         <ValueString "10"/> <DataType "Measure"/> <UnitofMeasure "KG"/>
      </Amount>
   </Material>
</Materials>
 . . .
```

3.10 Table Format

The ANSI/ISA 88 Part 3 standard also defines a table format for situations that may not require the complexity of the graphical format. The table allows a shorter representation of predominantly sequential sequences, but there is a graphical format for any table format.

This table representation also allows the definition of sequential and parallel paths through annotation of each row, but is limited to a single parallel definition (no nesting of parallel actions).

The following table is the table representation of the Process Operation depicted in Figure 9.

Sequence Order	Sequence Path	Operations & Actions	Material Definition
▽	0	Change:1	*Material IN* Material 111, Mat111, 100 kg
▽	0	Charge with Agitation:1	*Material IN* [1] Material 222, Mat222-001, 10 kg [2] Material 111, Mat111, 20 kg [2] Material 222, Mat222-001, 10 kg
▽	0	Change to Adjust pH:1	*Material IN* Material 222, Mat222-001, 2 kg
▽	0	Next Operation	

The order of execution in table is from the top to bottom, unless modified by the sequence order symbol and sequence path value. There is a directed link between "Operations & Actions" cells, and directed links between the "Material Definition" and the "Operations & Actions" cells.

In the ISA 88 Part 3 standard: "*The sequence order symbol indicates sequential execution, parallel branches, or sequential execution under parallel branches. The sequence order symbols shall be indicated as shown by Figure 12. The use of symbols within the table format allows simple parallel constructs to be quickly interpreted. Table representation of complex sequences of parallels and nested parallels are not defined in this standard.*"

Symbol	Description
	First in first series under a parallel (first action in first path)
	Action in middle of series under a parallel (not first or last in path)
	Last in last series of actions under a parallel (last action in last path)
	First in a series of actions under a parallel (first action in path)
	Last of series of actions under a parallel (last action in path)
	Action not under a parallel
	Single action at start of a parallel (only action in path)
	Single action under a parallel (only action in path)
	Single action at end of parallel (only action in path)

Figure 12 - Sequence order annotations for table representation[2]

The sequence path column indicates the sequence within a set of parallel sequences. Sequences should be numbered sequentially, starting from the left. All actions under the same sequence path execute sequentially.

---

[2] From ANSI/ISA 88 Part 3 General and Site Recipes Models and Representation, www.isa.org

4 Transaction Elements

The following elements are defined to support the transactions on General Recipe information, following the model defined in the ANSI/ISA 95.05 Business to Manufacturing Transaction standard. BatchML-GRecipe uses the transaction data types defined in the B2MML-Common.xsd schema.

4.1 Resource Constraint Library

The following table defines the recommended actions for a transaction message that contains a ResourceConstraintLibrary specification:

Resource Constraint Library	Description
GetResourceConstraintLibrary	• Returns the Resource Constraint Library and its resource constraint specifications in a ShowResourceConstraintLibrary message.
	• If no Resource Constraint Library ID is specified, then the receiver determines which library to return.
	• If a Resource Constraint Library ID is specified, then the receiver should return the library.
	• If no Resource Constraint Specifications (ResourceConstraintSpec) are defined, then return all resource constraint specifications in the library.
	• If a Resource Constraint Specification ID contains a wildcard character, then return all resource constraint specifications that match the wildcard.
	• If a Resource Constraint Specification ID is specified, then return all information about the identified resource constraint specification.
ProcessResourceConstraintLibrary	• "Process" the defined resource constraint specifications. This usually involves adding them to the Resource Constraint Library.
	• If no Resource Constraint Library ID is specified, then the receiver determines which Resource Constraint Library to be used.
	• If a Resource Constraint Library ID is specified, then the receiver should process the resource constraint specifications into the identified library.
	• No wildcards care allowed in resource constraint specification IDs.

Resource Constraint Library	Description
ChangeResourceConstraintLibrary	• Replace the defined resource constraint specifications in the Resource Constraint Library. *Note: The syntax does not specify a method to change library elements, so the sent Resource Constraint Specifications should contain all information with an assumption that the new information replaces all of the old information.* • If no Resource Constraint Library ID is specified, then the receiver determines which Resource Constraint Library to be used. • If a Resource Constraint Library ID is specified, then the receiver should change the resource constraint specifications into the identified library. • No wildcards care allowed in resource constraint specification IDs.
CancelResourceConstraintLibrary	• Cancel the defined resource constraint specifications in the Resource Constraint Library. *Note: This usually involves removing the resource constraint specification or marking them for removal.* • If no Resource Constraint Library ID is specified, then the receiver determines which Resource Constraint Library to be used. • If a Resource Constraint Library ID is specified, then the receiver should cancel the resource constraint specifications into the identified library. • If a Resource constraint Specification ID contains a wildcard character, then cancel all resource constraint specifications that match the wildcard.
SyncGRecipeInformation	• Sync with the ADD option should operate the same as a ProcessResourceConstraintLibrary message. • Sync with a CHANGE option should operate the same as a ChangeResourceConstraintLibrary message. • Sync with a DELETE option should operate the same as a CancelResourceConstraintLibrary message.

4.2 Process Element Library

The following table defines the recommended actions for a transaction message that contains a ProcessElementLibrary specification:

Process Element Specification	Description
GetProcessElementLibrary	• Returns the process element library and its process element specifications in a ShowProcessElementLibrary message.
	• If no Process Element Library ID is specified, then the receiver determines which library to return.
	• If a Process Element Library ID is specified, then the receiver should return the library.
	• If no Process Element Specifications (ProcessElementSpec) are defined, then return all process element specifications in the library.
	• If a Process Element Specification ID contains a wildcard character, then return all process element specifications that match the wildcard.
	• If a Process Element Specification ID is specified, then return all information about the identified process element specification.
ProcessProcessElementLibrary	• "Process" the defined process element specifications. This usually involves adding them to the process element library.
	• If no Process Element Library ID is specified, then the receiver determines which Process Element Library to be used.
	• If a Process Element Library ID is specified, then the receiver should process the process element specifications into the identified library.
	• No wildcards care allowed in process element specification IDs.
ChangeProcessElementLibrary	• Replace the defined process element specifications in the process element library. *Note: The syntax does not specify a method to change library elements, so the sent Process Element Specifications should contain all information with an assumption that the new information replaces all of the old information.*
	• If no Process Element Library ID is specified, then the receiver determines which Process Element Library to be used.
	• If a Process Element Library ID is specified, then the receiver should change the process element specifications into the identified library.
	• No wildcards care allowed in process element specification IDs.

Process Element Specification	Description
CancelProcessElementLibrary	• Cancel the defined process element specifications in the process element library. *Note: This usually involves removing the process element specification or marking them for removal.* • If no Process Element Library ID is specified, then the receiver determines which Process Element Library to be used. • If a Process Element Library ID is specified, then the receiver should cancel the process element specifications into the identified library. • If a Process Element Specification ID contains a wildcard character, then cancel all process element specifications that match the wildcard.
SyncGRecipeInformation	• Sync with the ADD option should operate the same as a ProcessProcessElementLibrary message. • Sync with a CHANGE option should operate the same as a ChangeProcessElementLibrary message. • Sync with a DELETE option should operate the same as a CancelProcessElementLibrary message.

4.3 GRecipe

The following table defines the recommended actions for a transaction message that contains General or Site Recipes.

GRecipe Information Elements	Description
GetGRecipeInformation	• If no Recipe ID is specified, then the receiver determines which recipes to return in a ShowGRecipeInformation message. • If a Recipe ID is specified, then the receiver returns the recipe information that it has for the specified recipe in a ShowGRecipeInformation message. • If a Recipe ID contains a wildcard, then the receiver returns the recipe information that it has for all recipes whose ID's match the wildcard in a ShowGRecipeInformation message.
ProcessGRecipeInformation	• If no Recipe ID is specified, then an error should be returned. • If the Recipe ID contains a wildcard, then an error should be returned. • If a Recipe ID is specified, then the receiver should process the associated recipe (typically placing it in local storage for later use).

GRecipe Information Elements	Description
ChangeGRecipeInformation	• If no Recipe ID is specified, then an error should be returned. • If the Recipe ID contains a wildcard, then an error should be returned. • If a Recipe ID is specified, then the receiver should process the changes to the associated recipe. *Note: The syntax does not specify a method to remove elements of the recipe, so the sent recipe should contain all information with an assumption that the new information replaces all of the old information.*
CancelGRecipeInformation	• If no Recipe ID is specified, then an error should be returned. • If the Recipe ID contains a wildcard, then all recipes that match the wildcard should be canceled. • If a Recipe ID is specified, then the receiver should cancel the associated recipe. *Note: This may not involve deleting the recipe from local storage, and may just be a marking for possible future delete.*
SyncGRecipeInformation	• Sync with the ADD option should operate the same as a ProcessGRecipeInformation message. • Sync with a CHANGE option should operate the same as a ChangeGRecipeInformation message. • Sync with a DELETE option should operate the same as a CancelGRecipeInformation message.

4.4 Transaction Profile

The recommended method for use of the Transaction Profile is to use the "Other" element for the Transaction Noun, and to use:

- GRECIPEINFORMATION for the GRecipe messages
- PROCESSELEMENTLIBRARY for the process element library messages
- RESOURCECONSTRAINTLIBRARY for the resource constraint library messages.

The following XML snippet illustrates how a transaction profile for the GRecipe elements could be defined.

```
<TransactionProfile>

    <SupportedAction>

        <TransactionVerb>GET</TransactionVerb>

        <TransactionNoun>"Other" /OtherValue="GRECIPEINFORMATION</TransactionNoun>

        <InformationUser>true</InformationUser>

        <InformationProvider>true</InformationProvider>

        <InformationSender>false</InformationSender>

        <InformationReceiver>true</InformationReceiver>

        <ObjectWildcardSupported>false</ObjectWildcardSupported>

    </SupportedAction>

</TransactionProfile>
```

5 Name Spaces

The General Recipe schemas are defined in the WBF's B2MML name space. The schemas include the B2MML Common types and B2MML Core Component types.

The user extensions to the General Recipe elements are defined in the BatchML GRExtensions using the *Extended* name space.

The following figure illustrates the Import and Includes of the files used in B2MML and BatchML GRecipe.
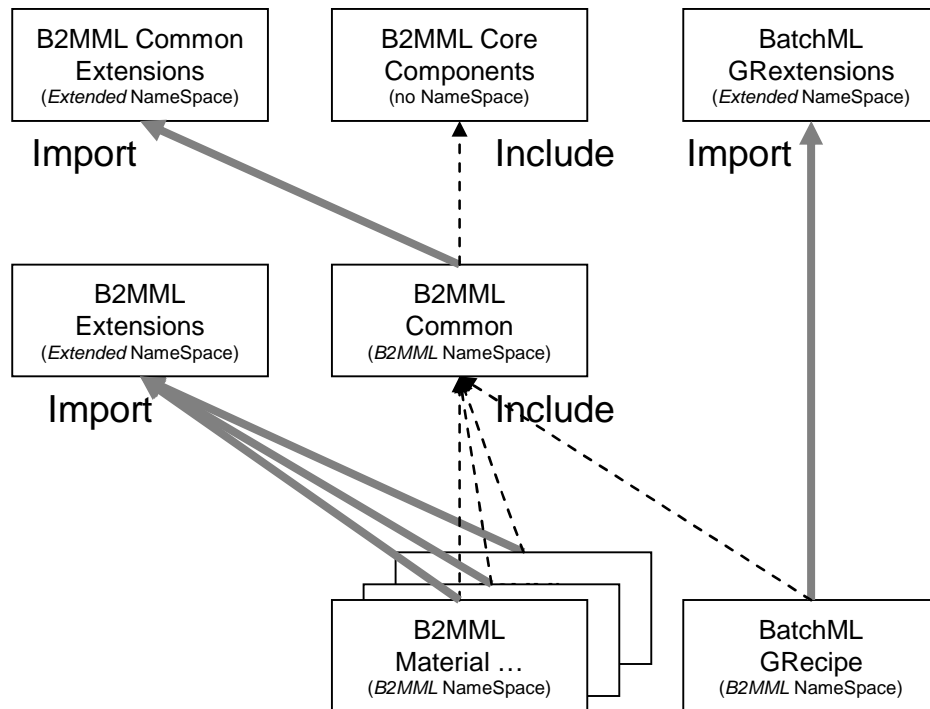
6 Diagram Convention

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.