# MathAssess Specifications February 2009

At the MathAssess Specification Meeting, held at Kingston University on 15-16 January 2009, and subsequently, the following specifications were agreed:

## Authoring MathML in the itemBody

- Authors can input the LaTeX form of the Presentation MathML
- The LaTeX is converted to Presentation MathML using SnuggleTeX
- The LaTeX is inserted as an annotation inside the Presentation MathML, for use in subsequent editing:

  `<m:annotation encoding='LaTeX'>\[\frac{2x-3}{x+4}\]</m:annotation>`
- The Presentation MathML (including annotation) is embedded directly into the itemBody
- Where a variable is needed in the LaTeX string, its identifier is enclosed within \qv{}:

```
<m:annotation encoding='LaTeX'>\[\frac{\qv{iA}x-
\qv{iB}}{\qv{mX}\qv{mSignC}\qv{iAbsC}}\]</m:annotation>
```

- Variable substitution is allowed only in <mi> and <ci> elements; in Presentation MathML, if the content of the variable is an operator, the <mo> element will be substituted during rendering.

### Example

```
<m:math display="block">
        <m:semantics>
                <m:mrow>
                        <m:mfrac>
                                <m:mrow>
                                        <m:mi>iA</m:mi>
                                        <m:mo>-</m:mo>
                                        <m:mi>iB</m:mi>
                                </m:mrow>
                                <m:mrow>
                                        <m:mi>mX</m:mi>
                                        <m:mi>mSignC</m:mi>
                                        <m:mi>mDen</m:mi>
                                </m:mrow>
                        </m:mfrac>
                </m:mrow>
                <m:annotation encoding='LaTeX'>
                        \[\frac{\qv{iA}x-\qv{iB}}{\qv{mX}\qv{mSignC}\qv{iAbsC}}\]
                </m:annotation>
        </m:semantics>
</m:math>
```

note <m:mi>, rather than <m:mo>

## Maths Content

The use of a Computer Algebra System (CAS) is essential for response Processing and in templateProcessing, where it is needed particularly for randomisation.

### Uses of Maths Content

The MathAssess project has identified five main uses for Maths Content; these are shown in the table below:

| *Required Use* | *Form of Data* | *Field Name* |
|---|---|---|
| **Visual form**<br>expressions are displayed as mathematics | **Presentation MathML** | **PMathML** |
| **Semantic meaning**<br>expressions have a mathematical meaning | **Content MathML**<br>**CAS code** – Maxima | **CMathML**<br>**Maxima** |
| **Computation**<br>expressions can be manipulated and combined to form other expressions | **CAS code** – Maxima | **Maxima** |
| **Input form for students**<br>as "natural" as possible | **Input string** - (if any) | **CandidateInput** |
| **Input form for authors**<br>expresses meaning and hence appearance | **CAS code** - Maxima | **Maxima** |

### Form of Maths Content

The QTI form of Maths Content is thus an itemVariable with

- cardinality = "record"
- Four fields:
    - **CandidateInput** containing the string entered by the student (if any)
    - **PMathML** containing the Presentation MathML form of the expression
        - This is sent to a separate string variable for display
        - Authors must use class="literal" inside printedVariable to display the Presentation MathML rather than the code.
    - **CMathML** containing the Content MathML form of the expression
    - **Maxima** containing the Maxima CAS code representing the expression

### Authoring Maths Content

It is possible to set defaultValue and correctResponse for maths content variables; however, this involves inserting all the forms of the variable, including both Presentation and Content MathML. In any case, this is usually needed only if there is no randomisation.

It is better to use the customOperator CasProcess or ScriptRule for setting values in templateProcessing or responseProcessing.

## MathAssess customOperators for Using CAS

| *Operation* | *customOperator* |
|---|---|
| Computing groups of maths expressions | ScriptRule |
| Computing the value of one variable | CasProcess |
| Comparing two QTI expressions | CasCompare |
| Evaluating a condition | CasCondition |

### Names of itemVariables

The form of variable names permitted in Maxima is more restrictive than in QTI, hence we must restrict the names of any variables which might be sent to Maxima as follows:

- To avoid problems in Maxima, Maxima reserved words are not permitted as variable names; these will be found in the Maxima manual packaged with the CAS software or at http://maxima.sourceforge.net/docs/manual/en/maxima.html
- Variable names must contain only alphanumeric characters, i.e. of the form:
    - As a regular expression: [a-zA-Z][a-zA-Z0-9]*
    - In BNF notation from the W3C specifications:

      MathVarIdentifier ::== MathVarIdentifierStartChar (MathVarIdentifierChar*)

      MathVarIdentifierStartChar ::== [a-z] | [A-Z]

      MathVarIdentifierChar ::== MathVaridentifierStartChar | [0-9]

So a variable name may contain any (English) alphabetic character followed by zero or more alphanumeric characters, but no punctuation or other special characters are permitted.

Some examples are given in the following table:

| **Allowed** | **Not allowed** | |
|---|---|---|
| **Name** | **Name** | **Reason** |
| x | 3x | Begins with a number |
| var0 | _var0 | Begins with non-alphanumeric character |
| endVal | end-val | Contains non-alphanumeric character, and might be interpreted as end minus val |
| cosX | cos | Maxima function name |
| s3 | simp | Maxima reserved word |

## ScriptRule

### *Attributes*

- syntax string (required)
    - Enumeration

      text/x-maxima

### *Authoring*

- The customOperator can be used anywhere where a QTI expression is permitted.
- The itemVariable used in <setTemplateValue/> (in the example this has identifier="dummy") must be declared as single cardinality and boolean.
- If a ScriptRule customOperator contains code with call(s) to the Maxima random() function, a variable with identifier="iRandomState" should be declared, with baseType integer and ordered cardinality, in order to preserve the random state for item review.
- The attribute syntax="text/x-maxima" will remain unchanged (unless, in a future implementation, a different CAS is to be used).
- The MathQurate authoring tool inserts the CDATA markup by default; authors coding by hand should insert CDATA markup.
- When variables containing values of units or operators are generated in Maxima, these should be created as Maths Content variables. In the Maxima code these variables should be assigned values using function notation:
    - for units, e.g. mUnitLength:maUnits("km");
    - for operators, e.g. mSignB:maOperator("+");
- Where the name of a variable within the question is randomised, the itemVariable containing the variable name is a mathsContent variable of record cardinality; a means of coding this, using arrays in Maxima, is suggested in the ScriptRule example below.
- Errors in the code will result in messages to the author.

### *Example*

```
<setTemplateValue identifier="dummy">
        <customOperator class="org.qtitools.mathassess.ScriptRule" ma:syntax="text/x-maxima">
                <baseValue baseType="string">
                        <![CDATA[
                        iA: ev((random(9)+1)*(random(2)*2-1),simp);
                        iB: ev((random(9)+1)*(random(2)*2-1),simp);
                        iAbsA: abs(iA);
```

```
                    iAbsB: abs(iB);
                    if iB>0 then mSignB: maOperator("+") else msSignB: maOperator("-");
                    iAB: iA*iB;
                    iAbsAB: abs(iAB);
                    array(aarr,19);
                    fillarray(aarr,[a,b,c,d,g,h,k,m,n,p,q,r,s,t,u,v,w,x,y,z]);
                    mX:aarr[random(19)];
                    ]]>
                </baseValue>
        </customOperator>
</setTemplateValue>
```

### *Implementation*

- Authoring systems may insert CDATA sections by default.
- CDATA sections are optional, and are used so that symbols do not need to be replaced with entities, e.g. a > b can be typed normally rather than a &lt; b. This facilitates testing of code in the CAS.
- The code should be extracted from any CDATA section before sending to Maxima,
- Before the CAS code is sent to Maxima, the Maxima session is informed of the values of all the itemVariables by creating local Maxima variables with the same name as their QTI counterpart and setting the appropriate value; in the case of Maths Content variables, only the value of the Maxima field is sent to Maxima.
- If the CAS code does not run in Maxima, a message is displayed.
- Afterwards, each templateVariable or responseVariable is given the value of its counterpart in Maxima.
- The operation of the random state in Maxima can be modified from within a test, see "Integration with Tests" below
- The customOperator returns a boolean value; if the value is false, a message is displayed to the user.
- When setting maths content variables, the customOperator will
    - Obtain output in Maxima's form of Presentation MathML,
    - Convert the Maxima MathML to sanitised Presentation MathML and thence to Content MathML,
    - Insert the forms of the expression into the appropriate fields, thus keeping the fields synchronised.

## CasProcess

### *Attributes*

- syntax string (required)
  - Enumeration

    text/x-maxima
- returnType string (required)
  - Enumeration

    mathsContent

    integer

    integerMultiple

    integerOrdered

    float

    floatMultiple

    floatOrdered

    boolean

    booleanMultiple

    booleanOrdered

    string

    stringMultiple

    stringOrdered

    point

    pointMultiple

    pointOrdered

### *Authoring*

- The customOperator can be used anywhere where a QTI expression is permitted.
- The itemVariable to be computed must be declared.
- The attribute syntax=”text/x-maxima” will remain unchanged (unless, in a future implementation, a different CAS is to be used).
- The attribute returnType is used to pass the expected type of return value to CasProcess; authors should ensure that this matches the expected baseType and cardinality.
- The code must consist of a single CAS command, though this may be a block command.

- The MathQurate authoring tool inserts the CDATA markup by default; authors coding by hand should insert CDATA markup.
- Use of the random() function should be avoided in CasProcess. If the random() function is used in CasProcess, the results are not repeatable because the random state is not preserved. In addition, this may affect the random state of Maxima, and hence this may not match the random state already saved by an earlier call to ScriptRule.
- Errors in the code will result in messages to the author.

## *Example*

```
<setTemplateValue identifier="mAns">
        <customOperator class="org.qtitools.mathassess.CasProcess" ma:syntax="text/x-maxima"
                                                                        ma:returnType="mathscontent">
                <baseValue baseType="string">
                        <![CDATA[
                        ev((random(5)+1)*y,simp);
                        ]]>
                </baseValue>
        </customOperator>
</setTemplateValue>
```

## *Implementation*

- The baseType and cardinality of the variable to be computed (mAns in the example) is obtained from the returnType attribute.
- Authoring systems may insert CDATA sections by default.
- CDATA sections are optional, and are used so that symbols do not need to be replaced with entities, e.g. a > b can be typed normally rather than a &lt; b. This facilitates testing of code in the CAS.
- The code should be extracted from any CDATA section before sending to Maxima,
- Before the CAS code is sent to Maxima, the Maxima session is informed of the values of all the itemVariables by creating local Maxima variables with the same name as their QTI counterpart and setting the appropriate value; in the case of Maths Content variables, only the value of the Maxima field is sent to Maxima.

  e.g. If y is an itemVariable, its value will be used in Maxima, otherwise, y is treated as an unknown.
- The values of itemVariables other than the variable to be computed must not be changed by CasProcess.

- When setting maths content variables, the customOperator will
    - Obtain output in Maxima's form of Presentation MathML,
    - Convert the Maxima MathML to sanitised Presentation MathML and thence to Content MathML,
    - Insert the forms of the expression into the appropriate fields, thus keeping the fields synchronised.

## CasCompare

### *Attributes*

- syntax string (required)
    - Enumeration

      text/x-maxima
- action string (required)
    - Enumeration

      equal

      syntequal

      code

      ...
- code string
    - CAS code (Maxima for MathAssess)
- simplify string (false)
    - Enumeration

      true

      false

### *Authoring*

- The value returned is a single cardinality boolean.
- If the attribute action="code", the attribute code contains the Maxima version of the condition.
- The attribute simplify="false" prevents Maxima from simplifying the CAS code before comparing.
- The two child elements are either itemVariables of cardinality record (i.e. mathContent) or else of baseType boolean, integer, float, string or point, or baseValue elements of baseType boolean, integer, float, string or point.

- Use of the random() function should be avoided in CasCompare. If the random() function is used in CasCompare, the results are not repeatable because the random state is not preserved. In addition, this may affect the random state of Maxima, and hence this may not match the random state already saved by an earlier call to ScriptRule.
- The table shows possible values for the action attribute; there will be others, but those described are definitely needed

| *action* | *CAS Code* | *Effect* |
|---|---|---|
| equal | is(equal($1,$2)); | algebraic equivalence (e.g. factorised and expanded forms are equivalent) |
| syntequal | is($1=$2); | syntactical equality (e.g. factorised and expanded forms are not syntactically equal) |
| code | *content of code attribute* | depends on code provided by the author |

### *Example*

```
<customOperator class="org.qtitools.mathassess.CasCompare" ma:syntax="text/x-maxima" ma:action = "equal" ma:simplify = "false">
        <variable identifier = "RESPONSE"/>
        <correct identifier = "RESPONSE"/>
</customOperator>
```

### *Implementation*

- If either of the two child elements is an itemVariable with cardinality="record", the Maxima field is selected for use in the comparison; if the Maxima field does not exist it returns NULL; if the Maxima field is NULL it returns NULL.
- For child elements whose cardinality is not record, the value of the whole variable is used.
- CasCompare must not change the values of any other variables.

## CasCondition

### *Attributes*

- syntax string (required)
  - o Enumeration

    text/x-maxima
- code string (required)
  - o CAS code (Maxima for MathAssess)
- simplify string (false)

- o Enumeration

  true

  false

### *Authoring*

- The CAS expression, which evaluates to true or false, is in the code attribute.
- Use of the random() function should be avoided in CasCondition. If the random() function is used in CasCondition, the results are not repeatable because the random state is not preserved. In addition, this may affect the random state of Maxima, and hence this may not match the random state already saved by an earlier call to ScriptRule.
- The customOperator has 0 or more child elements which are QTI expressions of the same number and in the same order as the $n parameters in the CAS code.
- The child elements are either itemVariables of cardinality record (i.e. mathContent) or else of baseType boolean, integer, float, string or point, or baseValue elements of baseType boolean, integer, float, string or point.
- The value of the content of each child element is substituted for the corresponding $n parameter in the code.
- If a $ symbol is needed, $$ is used.

### *Example*

In this example, the variable mAns is differentiated with respect to x and the result compared with the student input to determine whether the expressions are equivalent; the code is set to simplify the expressions before comparing.

```
<customOperator class=“org.qtitools.mathassess.CasCondition” ma:syntax="text/x-maxima"
                                        ma:code=“is(equal(ev(diff($1,x),simp),ev($2,simp)));”>
        <variable identifier = “mAns"/>
        <variable identifier = "RESPONSE"/>
</customOperator>
```

### *Implementation*

- If any of the child elements is a variable of record cardinality, the Maxima field is selected for use in the expression; otherwise, the value of the whole variable is used.
- The value of the CAS expression is returned as a single cardinality boolean.
- CasCondition must not change the values of any other variables.

## MathAssess customInteraction for Mathematical Input

### MathEntryInteraction

#### *Attributes*

|  |  |
|---|---|
| • syntax | string (required) |
|     o Enumeration | |
|     text/x-maxima | |
| • expectedLength | integer (20) |
| • printIdentifier | QTI identifier Type |

#### *Authoring*

MathEntryInteraction is a customInteraction. Handling of invalid input is the author's responsibility; suitable feedback should be provided for NULL or unexpected values.

#### *Example*

<customInteraction class="org.qtitools.mathassess.MathEntryInteraction" responseIdentifier="RESPONSE3" ma:syntax="text/x-maxima" ma:expectedLength="20" ma:printIdentifier="pResponse3" />

The responseIdentifier must have cardinality="record".

#### *Implementation*

- MathEntryInteraction sets a responseVariable of record cardinality which has the fields specified above.
- MathEntryInteraction provides
  - a text box
  - a preview field (e.g. for ASCIIMathML)
- On submission, MathEntryInteraction
  1. Puts the content of the text box into the CandidateInput field,
  2. Takes the ASCIIMathML form of the input from the preview field, cleans up the Presentation MathML, converts it to Content MathML and thence to Maxima,
  3. Populates the record cardinality responseVariable with the forms obtained in 1 and 2,
  4. Copies the serialized Presentation MathML form of the expression into the variable specified in printIdentifer, which must have been declared as a string variable of single cardinality.

## Integration with Tests

In order to take advantage of the review facilities which reconstruct the question as the student saw it in the assessment, allowReview must be set to true in the test and the assessmentItem needs to have the following additional features:

Additional template variables:

- keepRandomSeed integer, single
- iRandomSeed integer, single

Additional setup calculations in the first ScriptRule customOperator:

- If the assessmentItemRef corresponding to the question within the test contains a templateDefault element setting the default value of the boolean single variable keepRandomState to true,
    - If the variable iRandomSeed is NULL, a value is calculated; this may itself be a random integer (each student gets different parameters), or may have a specific value (all students then have the same parameters),
    - If the value of iRandomSeed is not NULL, the existing value is unchanged so that the random state in Maxima is recreated,
    - When the question is reviewed, the value of iRandomSeed will thus be reinstated before the ScriptRule code is executed.
- The first time a student opens the question in a test, a new random state is created in the first ScriptRule (if such exists), using iRandomSeed as the seed.

Subsequent ScriptRules do not modify the random state, unless the author does this within the Maxima code, in which case review of the original question is unlikely to be successful.

## Notes for Hand-Coding and Non-Standard XML Construction

- The Maxima code inside the ScriptRule and CasProcess customOperators should be wrapped in a CDATA section so that symbols do not need to be replaced with entities, e.g. a > b can be typed normally rather than a &lt; b. This facilitates testing of code in the CAS. The use of CDATA sections is optional but preferred; the MathQurate authoring tool inserts the CDATA markup by default; authors coding by hand should insert CDATA markup.
- In the CAS expression inside the code attribute of the CasCompare and CasCondition customOperators, any symbols must be encoded using entity or character references, e.g. &lt; for <. Authoring systems, such as MathQurate, will encode symbols automatically.

## Recommendations

1. The QTI specification for printedVariable is too restrictive; authors need to be able to print variables with other than single cardinality
2. Maths needs a special variable type; MathAssess is producing a well-described set of features.

## Changes

Changes in the Document:

29/01/2009

- Moved information about escaping symbols and CDATA sections to “Notes for Hand-Coding...”, to try to prevent “double escaping” of code by zealous users of authoring systems,
- Changed name of input field to CandidateInput from ASCIIMathInput,
- Removed reference to keeping PMathML generated by ASCIIMathML; the CandidateInput field is only used if the variable is bound to a MathEntryInteraction,
- Restricted the types of values permitted in the child elements of the CasCompare and CasCondition customOperators.

20/02/2009

- Added \qv{} notation for itemVariables within LaTeX,
- Extended the baseTypes which can be used in Maxima to include string and point,
- Provided facilities for using strings for units and operators as mathsContent variables,
- Suggested means for randomising variable names in ScriptRule,
- Added restriction on CasProcess, CasCompare and CasCondition to prevent side effects where variables other than the one identified in the parent of CasProcess or the returned boolean of CasCompare and CasCondition are affected by these customOperators ,

26/03/2009

- Clarified the use of variables, particularly operators, in PMathML in the itemBody,
- Added details of integration between CAS randomisation and tests.