

CALM XML Document Specification

Overview

CALM uses an XML document format to persist its internal representation of the layout of a clinical assessment, including the scores, questions of the assessment, the layout of the form elements to input assessment data and the associations of the form fields and logical field groups with the corresponding clinical assessment definition in the clinical database. The general structure of an CALM XML document is as follows

```
<caslayout type="document">
<document name="sars2.xml">
<description>A Clinical assessment layout description file</description>
<page number="1">
...
</page>
...
<page number="n">
</page>
</document>
<assessment name="Simpson Angus Rating Scale">
<scores>
....
</scores>
<mandatory-fields>
...
</mandatory-fields>
<items>
....
</items>
</assessment>
<bindings>
<assessment-binding id=" " name="assoc-assessment">
...
</assessment-binding>
<score-binding id=" " name="Overall">
...
</score-binding>
....
<mandatory-binding id=" " name="Date">
...
</mandatory-binding>
...
</bindings>
<logical-groups>
<group id='1'>
...
</group>
...
</logical-groups>
</caslayout>
```


In the following sections, each tag and children will be explained.

1. Document Declaration (<document>)

A document consists of multiple pages of forms for user input. Each form (<page>) usually corresponds to a single page in the paper form.

Attributes

- name - the name of the document (usually the filename the document is saved in if the CALM document is created by the CALM).
- version - the version number of the CALM clinical assessment (optional)

1.1 Page Declaration

Each page contains a single <container> object holding the form elements and other containers for complex layouts.

Attributes

- number - the page number (starts from 1).

1.2 Display Component Declaration

A display component corresponds to form input element or static text in a form. Each display component occupies a rectangular region of the size minus margins of the container cell it is in. Thus there can be only one display component per container cell. (A container is divided by an potentially irregular grid into cells for the display components). Like a container , a display component has margins of 3 pixels in each side, which means that the actual width and height of a display component is 6 pixels shorter than its cell's width and height, respectively. The x and y coordinates of a display component are absolute, not relative to the container the component is in. The common properties of a display component are

- x - the upper left hand corner x coordinate of the container (in pixels)
- y - the upper left hand corner x coordinate of the container (in pixels)
- height - the height of the container in pixels
- width - the width of the container in pixels
- id - a unique *component id* of the form letter 'C' followed by digits.
- name - currently not used but needs to be present and can be set to an empty string (e.g. <property name="" value="" />)

- parent - an integer unique identifier of the parent component (container) of this display component

In addition each type of display component have properties unique to them (Unless explicitly stated every property is required).

1.2.1 Text Display Component

A text display component corresponds to static text (with possible styles like bold, italic, underline via embedded html tags and/or CSS style).

- text - the text to be rendered. Can contain HTML tags , <u> or <i>. If any of these tags are used, they should be encoded for XML (e.g. < becomes < > becomes >)
- justification - the alignment of the text. Possible values are Left (default), Center, Right.

An example text display component in XML is as follows

```
<object class="caslayout.ui.TextDisplayComponent">
<property name="x" value="361" />
<property name="y" value="223" />
<property name="height" value="56" />
<property name="width" value="289" />
<property name="id" value="C136" />
<property name="name" value=" " />
<property name="parent" ref-id="2608483" />
<property name="text" value="&lt;b&gt;Source&lt;/b&gt;" />
<property name="justification" value="Left" />
</object>
```

1.2.2 Checkbox Display component

A checkbox display component corresponds to a checkbox form element.

- text - the label to the right of the checkbox. Can be empty.
- justification - the alignment of the checkbox and its label (if any) Possible values are Left (default), Center, Right.

An example checkbox display component is as follows

```
<object class="caslayout.ui.CheckBoxDisplayComponent">
<property name="x" value="650" />
<property name="y" value="223" />
<property name="height" value="56" />
<property name="width" value="526" />
<property name="id" value="C131" />
<property name="name" value=" " />
<property name="parent" ref-id="2608483" />
<property name="text" value="Proband" />
<property name="justification" value="Left" />
</object>
```

1.2.3 Radio Button Display component

A radio button display component corresponds to a radio button form element.

- text - the label to the right of the radio button. Can be empty.
- value - the value that is submitted when the radio button is selected. Can be different than the label (text property).
- justification - the alignment of the radio button and its label (if any) Possible values are Left (default), Center, Right.

An example radio button display component is as follows

```
<object class="caslayout.ui.RadioButtonDisplayComponent">
<property name="x" value="1045" />
<property name="y" value="52" />
<property name="height" value="23" />
<property name="width" value="23" />
<property name="id" value="C21" />
<property name="name" value="" />
<property name="parent" ref-id="8820986" />
<property name="text" value="0" />
<property name="value" value="Normal" />
<property name="justification" value="Center" />
</object>
```

1.2.4 Text Field Display Component

A text field display component corresponds to a text input field in a form.

- fieldLength - the length in characters of the text field.
- maxFieldLength - the maximum length in characters of the text field.
- justification - the alignment of the text field (if any) Possible values are Left (default), Center, Right.

An example text field display component in XML is as follows

```
<object class="caslayout.ui.TextFieldDisplayComponent">
<property name="x" value="652" />
<property name="y" value="508" />
<property name="height" value="107" />
<property name="width" value="528" />
<property name="id" value="C137" />
<property name="name" value="" />
<property name="parent" ref-id="32032133" />
<property name="fieldLength" value="10" />
<property name="maxFieldLength" value="10" />
<property name="justification" value="Left" />
</object>
```

1.2.5 Text Area Display Component

A text area display component corresponds to a textarea input field in a web form.

- numCols - the width of the text area field in number of characters.
- numRows - the height of the text area field in number of lines.
- justification - the alignment of the text field (if any) Possible values are Left (default), Center, Right.

An example text area display component in XML is as follows

```
<object class="caslayout.ui.TextAreaDisplayComponent">
<property name="x" value="888" />
<property name="y" value="6" />
<property name="height" value="224" />
<property name="width" value="294" />
<property name="id" value="C42" />
<property name="name" value="" />
<property name="parent" ref-id="30776636" />
<property name="numCols" value="20" />
<property name="numRows" value="3" />
<property name="justification" value="Left" />
</object>
```

1.2.6 Dropdown Display Component

A dropdown display component corresponds to a textarea input field in a web form.

- justification - the alignment of the text field (if any) Possible values are Left (default), Center, Right.

1.2.6.1 Options section

Each dropdown component needs to have a <options> section containing zero or more <option> tags. Each option has two attributes.

- label - the label that will be displayed to the user in the dropdown component
- value - the value string that will be submitted to the web server with the form.

An example dropdown display component in XML is as follows

```
<object class="caslayout.ui.DropdownDisplayComponent">
<property name="x" value="593" />
<property name="y" value="248" />
<property name="height" value="12" />
<property name="width" value="586" />
<property name="id" value="C2193" />
<property name="name" value="" />
<property name="parent" ref-id="14891765" />
<options>
<option label="unknown relationship" value="unknown relationship" />
<option label="self" value="self" />
<option label="mother" value="mother" />
<option label="father" value="father" />
<option label="mother-in-law" value="mother-in-law" />
<option label="father-in-law" value="father-in-law" />
</options>
```



```

<option label="sister" value="sister" />
<option label="brother" value="brother" />
<option label="sister-in-law" value="sister-in-law" />
<option label="brother-in-law" value="brother-in-law" />
<option label="daughter" value="daughter" />
<option label="son" value="son" />
<option label="daughter-in-law" value="daughter-in-law" />
<option label="son-in-law" value="son-in-law" />
<option label="aunt" value="aunt" />
<option label="uncle" value="uncle" />
<option label="niece" value="niece" />
<option label="nephew" value="nephew" />
<option label="colleague" value="colleague" />
<option label="roommate" value="roommate" />
</options>
<property name="justification" value="Center" />
</object>

```

1.2.7 Button Display Component

A button display component corresponds to a submit button in a HTML form. Only 'Next', 'Previous' and 'Submit' buttons can be defined.

Every multi-screen (multi-form) assessment requires navigation buttons ('Previous' and/or 'Next'). Currently only a single 'Submit' button is supported. It can appear in multiple form screens to submit what is populated so far.

- label - the label of the button. Possible values are 'Next', 'Previous' and 'Submit'
- action - the name of the action to be performed when this button is pressed on an online form. Possible values are 'Next', 'Previous' and 'Submit'
- justification - the alignment of the text field (if any) Possible values are Left (default), Center, Right.

An example text field display component in XML is as follows

```

<object class="caslayout.ui.ButtonDisplayComponent">
<property name="x" value="1004" />
<property name="y" value="829" />
<property name="height" value="71" />
<property name="width" value="176" />
<property name="id" value="C150" />
<property name="name" value="" />
<property name="parent" ref-id="32032133" />
<property name="label" value="Next" />
<property name="action" value="Next" />
<property name="justification" value="Center" />
</object>

```


1.3 Container Declaration

A container is a rectangular area which contains zero or more display components and/or other containers. Each container is uniquely identified by an **id**, each container has a **layout object** describing how the contained objects should be laid out on the CALM screen. Each container has a list of components for the display components and/or other containers contained by it. Each container has margins (as defined by a `java.awt.Insets` object)

Attributes

- **class** - the java class corresponding the container (`caslayout.ui.CAPanel`)
- **isGroup** - not used but must be present. Setting it to 'false' is the best bet.
- **id** - the unique identifier (an integer) used to determine parent-child object relationships. The only prerequisite is it must be unique.

Each container (and each display component object) has the following properties defined

Properties

- **x** - the upper left hand corner x coordinate of the container (in pixels)
- **y** - the upper left hand corner y coordinate of the container (in pixels)
- **height** - the height of the container in pixels
- **width** - the width of the container in pixels
- **id** - a unique id of the form letter 'C' followed by digits. This id property is different than the id attribute necessary for the XML specification of a container. The id attribute in the XML serialized version of a container is a unique number (a hashcode) used internally to determine parent-child relationships.
- **name** - currently not used but needs to be present and can be set to an empty string (e.g. `<property name="name" value="" />`)

Below, is a simplified declaration of a container in XML

```
<container class="caslayout.ui.CAPanel" isGroup="false" id="17023643">
<property name="x" value="0" />
...
<layout-man id="15607307" class="caslayout.ui.CAGridLayout" name="layoutMan">
...
</layout-man>
<insets class="java.awt.Insets">
...
</insets>
<collection name="components" type="list" class="java.util.ArrayList" size="1">
...
</collection>
</container>
```


1.3.1 Layout Manager Declaration

A layout manager object is responsible for the laying out of the components contained by a container. It defines the cells where the child components reside. There are two mechanisms available to create tailored layouts, which are a column span, row span based grouping of cells into a one cell and percentage based resizing of the columns and rows. You should use percentage based method, since it is more intuitive and robust one. Each layout manager specifies the number of columns and rows of the initial grid. Because of the column/row merges each row may have different number of apparent cells.

Attributes

- id -the unique identifier (an integer) used to determine parent-child object relationships. The only prerequisite is it must be unique.
- class - the Java class for the layout manager (`caslayout.ui.CAGridLayout`)

Properties

- cols - number of columns in the initial grid
- rows - number of rows in the initial grid
- hGap - horizontal gap in pixels (can be 0)
- vGap - vertical gap in pixels (can be 0)

1.3.1.1 GridCellInfo Declaration (<grid-cell-info>)

A GridCellInfo is helper holding the state for the constraints of each grid cell in a layout manager. A GridCellInfo always has a layout manager as parent and always contained within a layout manager object. Each GridCellInfo contains a collection (generic data structure to hold multiple objects) for the visible rows of the layout. Because of the possible merging of the columns, each row may have apparently different number of columns. In percentage based layouts (the recommended layout model), the rows and columns cannot merged. However, columns can have 0% widths, resulting in apparent decrease in the number of columns in an individual row.

Attributes

- class - the Java class for the GridCellInfo object (always `caslayout.ui.GridCellInfo`).

Properties

- maxRows - the maximum number of rows in the initial grid. Because of possible row merging, the actual number of rows may be less than this.
- maxCols - the maximum number of columns any row can have.
- parent -an integer unique identifier of the parent layout manager of this GridCellInfo object (e.g. `<property name="parent" ref-id="15607307" />`)

1.3.1.1.1 Declaration of the grid cell constraints (<collection>)

The cell constraints per row are contained in a *Collection* object. Each row <entry> contains an <array> data structure holding CellConstraint objects. Since percentage based cell constraining is the recommended way, the <array> will contain cell constraints of type `casayout.ui.PercentCellConstraint` for non-zero percent width columns. Hence each row entry can have an array which is not the same size as the original (`maxCols`) number of columns.

1.3.1.1.1.1 PercentCellConstraint Object Declaration (<object>)

A `PercentCellConstraint` Object is a simple object (has attributes and properties and no complex objects). It defines the grid location and size of a cell in percentages.

Attributes

- class - the Java class corresponding to the `PercentCellConstraint` Object (`casayout.ui.PercentCellConstraint`)

Properties

- `rowIdx` - 0 based row index of the cell
- `colIdx` - 0 based column index of the cell
- `rowPercent` - the height of the cell in percentage of the total height of the container
- `colPercent` - the width of the cell in percentage of the total width of the container

1.3.2 Margin (Insets) Declaration (<insets>)

Each container has an margin object.

Attributes

- class - the Java class corresponding to the margin object (`java.awt.Insets`)

Properties

- left - left side margin in pixels (always 3)
- top - top margin in pixels (always 3)
- right - right side margin in pixels (always 3)
- bottom - bottom margin of the container in pixels (always 3)

1.3.3 Components collection for the container object (<collection>)

The components contained in a container are represented via a *Collection* having the name 'components'. Each <entry> in the components collection corresponds either to a single display component object or another container. Any zero width column is represented by a null in the CALM. The <entry> index specifies where the entry is in the collection and indirectly where it is in the visualized container. The components collection can be seen as a one dimensional representation of the layout grid, first the columns of the first row, then the second and so on.

2. Assessment Declaration (<assessment>)

The assessment declaration corresponds to the assessment metadata read from the BIRN HID database particularly from NC_ASSESSMENT, NC_ASSESSMENTSCORE, NC_ASSESSMENTSCORECODE and NC_ASSESSMENTITEM tables for the particular assessment the form fields are associated.

Attributes

- name - the name of the clinical assessment (as stored or will be stored in NC_ASSESSMENT.NAME column).

The <assessment> tag has a <scores> tag encapsulating 0 or more <score> tags.

2.1 Score declaration (<score>)

A <score> tag represents a clinical assessment score complete with its score codes (enumerations).

Attributes

- assessment - The name of the assessment this score belongs (as stored or will be stored in NC_ASSESSMENT.NAME column).
- name - the score name (as stored or will be stored in NC_ASSESSMENTSCORE.SCORENAME).
- sequence - an integer representing the place of this score in the scores sequence (as stored or will be stored in NC_ASSESSMENTSCORE.SCORESEQUENCE).
- type - the data type of this score (Possible values are 'integer', 'float', 'varchar', 'boolean')
- level - an integer representing the tree level of this score in the score hierarchy (as stored or will be stored in NC_ASSESSMENTSCORE.SCORELEVEL).
- sec-class - A string representing the security class for this assessment score

A score can have a limited number of possible values, which are represented by a series of <scorecode> tags encapsulated in a <scorecodes> tag.

2.1.1 Score Parent declaration (<parent>)

Since scores can form hierarchies, some scores will have a parent as indicated by the <parent> tag

Attributes

- assessment - the name of the assessment the parent score belongs
- name - the score name of the parent score

2.1.2 Scorecode declaration (<scorecode>)

Attributes

- score-name -
- code -
- code-value
- type -

2.2 Mandatory Fields declaration (<mandatory-fields>)

TBD

2.2.1 Field declaration (<field>)

Attributes

- name - name of the mandatory field (possible values are 'date' and 'time')
- type - the data type of the mandatory field (can only be 'string')

2.3 Assessment Items (Questions) declaration (<items>)

This tag encapsulates zero or more <item> tags. **2.3.1 Assessment Item (Question) declaration (<item>)**

TBD

3. Bindings (Associations) Declaration (<bindings>)

3.1 Assessment binding (<assessment-binding>)

3.1.1 Left side of the assessment - document association (<left>)

Attributes

- assessment-name - the clinical assessment name the form(s) in this CALM will be associated (as stored or will be stored in NC_ASSESSMENT.NAME column).

3.1.2 Right side of the assessment - document association (<right>)

Attributes

- doc-name - The name of the CALM document

3.2 Score - form element(s) association (<score-binding>)

Attributes

- id - The id of the score association. Currently not used, however must be set to empty string.
- name - the name of the score association (binding). Must be the score name.

3.2.1 Left side of the score - form element(s) association (<left>)

Attributes

- score-name -
- assessment -

3.2.2 Right side of the score - form element(s) association (<right>)

Attributes

- id -

3.2.3 Score Value Binding (<score-value-binding>)

Attributes

- lid -

3.3 Score Code - form element Associations (<score-code-bindings>) (Optional)

Allows determination of the form element score code bindings (which score code is mapped to which display component). **3.3.1 Score code - form element association (<sc-binding>)** Attributes

- score-code - The value of the scorecode column of the corresponding score code for the score. Score codes are enumerated values for a score.
- id - the ID of the form element (a groupable component like radio button or checkbox) which is associated with the score code.

3.4 Mandatory field - form element Association (<mandatory-binding>)

Attributes

- id - The id of the mandatory field association. Currently not used, however must be set to empty string.
- name - the name of the mandatory field association (binding). Must be same as the corresponding mandatory field.

3.4.1 Left side of the mandatory field - form element association (<left>)

Attributes

- property - The name of the mandatory field

3.4.2 Right side of the mandatory field - form element association (<right>)

Attributes

- id - the component id of the form element (See display component id attribute).

3.4.3 Value Binding (<value-binding>) Attributes

- id -

TBD

4. Logical Groups Declaration (<logical-groups>)

Holds a list of grouped display components (like check boxes or radio buttons) mapping to a single score. Each logical group is represented by a <group> XML tag.

4.1 Logical Group Declaration (<group>)

Attributes

- id - an integer uniquely identifying the group (starts from 1)

Contains one or more <display-comp> XMLtags

4.1.1 Display component reference Declaration (<display-comp>)

Attributes

- id - a unique id of the form letter 'C' followed by digits for the display component belonging this logical group.

5. Data Structure Representations in XML

5.1 Array An array is a collection of data items , all of the same type, in which each item's position is uniquely designated by an integer. Each array XML definition contains a definite number (as defined by the length attribute) of <object> definitions of the same type

Attributes

- class - the Java class of the type of objects contained (e.g. caslayout.ui.PercentCellConstraint for percentage constraint applied to a cell)
- length - the number of data items in the array

5.2 Collection A collection is a data structure groups together several pontentially different things. XML definition of a collection consists of a number of <entry> objects corresponding to the non-null objects in the collection. A container can have null elements, the size attribute contains also the null objects.

Attributes

- name - the name of the property of the class having this collection as a property (instance variable). For example each containidner class like caslayout.ui.CAPanel in the CALM, has an instance variable (property) named components which is a collection. Hence the name attribute for the container caslayout.ui.CAPanel is 'components'.
- type - the type of the collection (Always 'list').
- class - the Java class corresponding to the collection implementation (always java.util.ArrayList).
- size - the number of objects (including null objects) in the collection.

5.2.1 Entry An entry corresponds to a thing inside a collection. An entry encapsulates any type of object, including other data structures.

Attributes

- index - a 0 based index of the object encapsulated by this entry in the collection

\$Id: calm_document_spec.tex,v 1.2 2007/10/08 17:52:09 bozyurt Exp \$