

Generic UIML Vocabulary Specification

Abstract

It is the purpose of the Generic UIML vocabulary to be platform and language independent while providing to the UI designer a powerful set of parts that can be used to create user interfaces on virtually any platform with little need for platform or language specific optimization. While it is not possible to remove all need for target platform or language optimizations to control the details of target-specific rendering, it is the goal of this vocabulary to reduce that need as much as possible.

Creating a generic vocabulary to effectively describe user interfaces is a continuous endeavor, therefore changes and enhancements will be continually introduced over the lifetime of the project. Efforts will be made to preserve existing functionality, but when that is not possible lists of changed and deprecations will be published to inform designers of modifications to the vocabulary that may break existing designs.

Goals of the Generic Vocabulary:

1. Create a generic widget set representative of the most common types of user interface controls which can also be mapped to more complex control types
2. Create a generic set of properties to define all basic aspects of each generic widget
3. Create a generic set of basic data types that can be mapped to language specific types
4. Allow UI layout to be handled in a generic way by supporting various common layout techniques in the generic vocabulary

Generic UIML Vocabulary Specification Abstract	1
1 Table of Generic Widgets, Properties and Data Types	3
1.1 Generic Widgets	3
1.1.1 Generic Containers	3
1.1.2 Generic Controls	4
1.2 Generic Properties and Events	4
1.2.1 Common Generic Properties	4
1.2.2 Layout Properties	5
1.2.3 Generic Events	6
1.3 Generic Data Types	7
2 Specification	8
2.1 Widgets	8
2.1.1 Containers	8
2.1.2 Controls	19
2.1.3 Text	30
3 Layout	33
4 Borders	36
4.1 Supported Generic Borders	36
Appendix A – Constants	37
Appendix B – Known Issues and Future Enhancements	39

1 Table of Generic Widgets, Properties and Data Types

These tables describe each element of the generic vocabulary and what its purpose is as a widget, property or event.

Conventions used in this specification

Identifier Type	Convention	Example
Widget	<i>G:Identifier</i>	G:TopContainer
Property/Event	<i>g:identifier</i>	g:size
Data Type	<i>g:Identifier</i>	g:Integer
Property Value	<i>literal; x,y, predefined choices</i>	15; 300,200; true false
Default value	Choice Preceded by *	*true false – True is default
Accelerator	Java KeyStroke.getKeyStroke() string	"control alt a"; "typed b"
Mnemonic	<i>Character Literal</i>	a; Q; z

1.1 Generic Widgets

1.1.1 Generic Containers

<i>Element</i>	<i>Purpose</i>	<i>Page</i>
G:TopContainer	The generic container in which all parts are placed in the target	8
G:Area	Container describing a section of the surrounding container	10
G:ChildFrame	An child frame for an MDI application	11
G:MenuBar	Top-level container to hold G:Menu's	13
G:Menu	Container to describe any type of menu	13
G>List	Container to present a sequential ordering of elements	14
G:Table	Container providing a 2-dimensional, homogeneous, view of data or parts	15
G:HetroTable	Container providing a 2-dimensional view of heterogeneous parts. Each cell can contain different parts, regardless of column.	16

<i>Element</i>	<i>Purpose</i>	<i>Page</i>
G:TableRow	Single row for a G:HeteroTable	17
G:Tree	Container providing a hierarchical view of data	17
G:ToolBar	Container and layout for a tool bar	18

1.1.2 Generic Controls

<i>Element</i>	<i>Purpose</i>	<i>Page</i>
G:Button	Generic control who only has a single interaction mode	19
G:ButtonGroup	Generic logical grouping for buttons for multiple exclusion	20
G:Image	Generic control representing visual (non-text) data	21
G:MultiMedia	Generic control representing any media object	21
G:Text	Generic control representing text and embedded data	30
G:TextBox	Generic Control representing a multi-line text area	22
G:TextField	Generic control representing a single line of text input	24
G:ColumnDef	Column definition for a homogeneous table	25
G:RowDef	Row definition for the homogeneous table	27
G:TableCell G:TreeItem	Generic control representing a single item in a generic container-control (HeteroTables, Trees)	27
G:MenuItem	Represents a single menu item in a G:Menu	27
G:RangeSelector	Generic control allowing the user to select between a range	28
G:PromptInput	Generic control prompting the user for some input	29

1.2 Generic Properties and Events

1.2.1 Common Generic Properties

Property	Description	Page
g:alternate-text	Specify the text rendered if the target cannot render the control's default form	

Property	Description	Page
g:bgcolor	Specify the background color of the widget	
g:border	Specify the border type for the container	
g:fgcolor	Specify the foreground color of the widget	
g:content	Specify the content of the container or control	
g:editable	Specify whether or not the control accepts input (affects rendering)	
g:enabled	Specify whether or not the widget is enabled for input	
g:font	Specify the font for the control	
g:image-src	Specify the image to be associated with the control	
g:location	Specify the location in absolute terms for the widget	
g:mimetype	Specify the MIME type for the widget	
g:padding	Specify the distance between this control and surrounding controls	
g:resizable	Specify whether or not the window is resizable	
g:size	Specify the physical size of the widget	
g:text	Specify text associated with the widget	
g:title	Specify the title of the widget	
g:visible	Specify whether or not the widget should be rendered currently	

1.2.2 Layout Properties

Property	Description	Page
g:layoutborder	Set the border alignment in a Border Layout	
g:layout	Specify the part layout for the container	
g:layoutwidth	Specify the number of columns to span in the layout	

Property	Description	Page
g:layoutheight	Specify the number of rows to span in the layout	
g:layoutmargin	Sets the padding between objects to be laid out	
g:layoutweightx	Sets the weight for the width of a cell in a Grid Layout	
g:layoutweighty	Sets the weight for the height of a cell in a Grid Layout	

1.2.3 Generic Events

Event	Description	Page
g:actionperformed	When the user performs the default action on the control	
g:activated	When the window is switching from a nonactive state to active	
g:closed	When the window has been destroyed	
g:closing	When the user has tried to close the window	
g:deactivated	When the window is switched from active state to nonactive state	
g:focuslost	When the widget loses the focus of the user	
g:keypressed	When a key is pressed	
g:keyreleased	When a key is released	
g:keytyped	When the user has pressed a key on a text input device	
g:locatordbltapped	When the user's locator device has tapped twice	
g:locatordown	When the locator device has been pressed down	
g:locatorentered	When the user's locator device enters the confines of the widget	
g:locatorexited	When the user's locator device has exited the confines of the widget	
g:locatormoved	When the user's locator device has moved	
g:locatortapped	When the user's locator device has done some input	

Event	Description	Page
g:locatorup	When the locator device has been released	
g:onfocus	When the widget gains the focus of the user	
g:opened	When the window is first displayed to the user	
g:selectionchanged	When the user changes the selection in a list or table	

1.3 Generic Data Types

Type	Description	Page
g:String	Character string	
g:Character	Single character	
g:Integer	Single integer	
g:Float	Single floating point number	
g:Double	Single double precision floating point number	
g:Boolean	Single true/false value	
g:Object	Single Object in languages that support compound types	
g:Color	32-bit Color specified in the RGBA form as a comma-separated quad (R,G,B,a) where each member is an unsigned 8-bit integer (ex. 23, 234, 128, 0) or as a 32bit Hex number in the form RrBbGgAa where each of Rr, Bb, Gg, and Aa are 8 bit hex values.	
g:Dimension	Dimension as a comma-separated double (ex. 800,600)	
g:Font	Font specified as family-style-size. See Text section for full explanation.	
g:Point	A single point on the display	

2 Specification

Widget properties and events control how they will be rendered on the target platform and language and how they will interact with each other. It is possible to use the generic set of properties and events to closely control the type of control the target generates, how it will look and how it will interact with the user and other widgets without sacrificing the generality of the design. This section will define what properties and events are supported on each widget and how they combine to define the role of the widget and how they allow the UI designer to implement a powerful user interface using a simple generic toolkit.

2.1 Widgets

This section defines the properties and events on all widgets in the generic vocabulary.

The properties enumerated are not exhaustive, the designer is free to provide as many or as few properties as are necessary to fully describe the widget.

Events enumerated under each widget are exhaustive. If other events are required that are not enumerated, the designer will have to provide language-specific event handlers for each language that is to be supported by the user interface.

2.1.1 Containers

2.1.1.1 G:TopContainer

The generic top container will render to the root container of the target platform and language. All user interface parts will be children of this container.

Recommended Properties:

Property	Valid value	Description
g:title	<i>Text</i>	Sets title of the main container
g:topcontainertype	*sdi	Creates a single document interface
	mdi	Creates a multiple document interface
	applet	Creates a web-enabled application, if supported
	dialog	Creates a dialog container if supported
g:layout		See Layout chapter 3
g:size	<i>x,y</i>	
g:image-src	<i>Resource</i>	Resource ID to icon for top container

Property	Valid value	Description
g:resizable	*true false	Whether or not the top container is resizable
g:modal	true *false	Whether or not the top container is modal (applies to dialogs only)

Optional Properties

Property	Valid Value	Description
g:bgcolor	<i>Color</i>	Background Color
g:fgcolor	<i>Color</i>	Foreground Color

Events Supported

Event	Description
g:locatormoved	Called when the mouse moves across the container
g:locatortapped	Called when the locator device is pressed and released
g:locatorentered	Called when the locator device enters the container
g:locatorexited	Called when the locator device leaves the container
g:keytyped	Called when a keyboard key is typed
g:opened	Called when the container is first opened
g:closing	Called after the user has requested the container be closed
g:closed	Called after the container has been destroyed
g:activated	Called when the container is selected
g:deactivated	Called when the container is deselected

Example:

```
<part id="top_sdi" class="G:TopContainer">
  <style>
```

```

<property name="g:topcontainertype">sdi</property>
<property name="g:title">SDI Application</property>
<property name="g:size">640,480</property>
<property name="g:resizable">>false</property>
<property name="g:image-src">icon_1.gif</property>
</style>
</part>

```

2.1.1.2 G:Area

A G:Area describes an abstract area in the parent container. These areas are useful for containing child parts that logically should be grouped together. The rendering of a G:Area is greatly affected by the properties set on it. By default, a G:Area will render down into the most abstract generalization of a container in the target language (i.e. JPanel, or <DIV>).

Recommended Properties:

Property	Valid value	Description
g:title	<i>Text</i>	Title for the area
g:scrollable	true *false	Whether or not the area is scrollable
g:border		See Borders chapter 4
g:size	<i>Dimension</i>	Preferred size of the area

Optional Properties

Property	Valid value	Description
g:areatype	tabbed	Creates a tabbed pane
	split	Creates a resizable split viewing area
	parent	Creates a parent window for G:ChildFrame's
	*div	Creates a simple container to hold other widgets
g:layout	See Ch. 3	Specifies the layout manager to be used for the area
g:location	<i>Point</i>	Location of the upper-left corner of the area for absolute layout
g:bgcolor	<i>Color</i>	Background Color
g:fgcolor	<i>Color</i>	Foreground Color

*Please note, use of optional properties may cause some targets not to render correctly or to be rendered in an alternate way not intended by the designer, such options will be documented with the target platform.

Events Supported

Event	Description
g:locatormoved	Called when the mouse moves across the container
g:locatortapped	Called when the locator device is pressed and released
g:locatorentered	Called when the locator device enters the container
g:locatorexited	Called when the locator device leaves the container
g:keytyped	Called when a keyboard key is typed

Example:

```
<part id="scrollable" class="G:Area">
  <style>
    <property name="g:scrollable">true</property>
    <property name="g:title">A Scrollable Tab Pane</property>
  </style>
  <part id="tabbed" class="G:Area">
    <style>
      <property name="g:areatype">tabbed</property>
    </style>
  </part>
</part>
```

2.1.1.3 G:ChildFrame

A G:ChildFrame is a child of an MDI G:TopContainer. It acts just like a G:TopContainer in every aspect, it can take the same children and it can have the same properties set on it, except for g:type.

Recommended Properties:

Property	Valid value	Description
g:title	<i>Text</i>	Sets title of the child container
g:layout		See Layout chapter 3
g:size	<i>x,y</i>	
g:selected	true false	If this child frame has the focus
g:resizable	true false	Whether or not this frame can be resized

Optional Properties

Property	Valid Value	Description
g:bgcolor	<i>Color</i>	Background Color
g:fgcolor	<i>Color</i>	Foreground Color

Events Supported

Event	Description
g:activated	Called when the container is selected
g:closed	Called after the container has been destroyed
g:closing	Called after the user has requested the container be closed
g:deactivated	Called when the container is deselected
g:keytyped	Called when a keyboard key is typed
g:locatorentered	Called when the locator device enters the container
g:locatorexited	Called when the locator device leaves the container
g:locatormoved	Called when the mouse moves across the container
g:locatortapped	Called when the locator device is pressed and released
g:opened	Called when the container is first opened
g:resized	Called when the component is resized

Example:

```
<part id="mdi" class="G:TopContainer">
  <style>
    <property name="g:topcontainertype">mdi</property>
    <property name="g:size">200,200</property>
    <property name="g:title">Parent Frame</property>
    <property name="g:visible">>true</property>
  </style>
  <part id="child" class="G:ChildFrame">
    <style>
      <property name="g:title">Child Frame</property>
      <property name="g:size">300,100</property>
    </style>
  </part>
</part>
```

```

    <property name="g:resizable">false</property>
  </style>
</part>
</part>

```

2.1.1.4 G:MenuBar

A G:MenuBar is simply a container that contains G:Menu's. No other children are allowed under this part. For targets that support native menu bar's, this will simply be integrated, for those that do not, a reasonable replica will be created from available parts.

2.1.1.5 G:Menu

A G:Menu is a collection of G:Button's that can be displayed as part of a G:MenuBar or displayed as a popup menu. G:Menu's may contain other G:Menu's to create nested menus.

Required Properties:

Property	Valid value	Description
g:text	<i>Text</i>	Sets title displayed for this menu

Recommended Properties:

Property	Valid value	Description
g:enabled	*true false	Whether the menu is allowing input
g:alternate-text	<i>Text</i>	Alternate text to display for the menu

Optional Properties

Property	Valid value	Description
g:image-src	<i>Resource ID</i>	Sets an icon associated with this menu
g:accelerator	<i>Accelerator</i>	Sets the accelerator key for this menu
g:mnemonic	<i>Mnemonic</i>	Sets the mnemonic for the menu
g:fgcolor	<i>Color</i>	Sets the foreground color for the menu
g:bgcolor	<i>Color</i>	Sets the background color for the menu

Example:

```
<part id="menubar" class="G:MenuBar">
  <part id="menu1" class="G:Menu">
    <!--Menu Item Children -->
  </part>
</part>
```

2.1.1.6 G:List

A G:List provides the user with a sequential view of data that may be merely structured output or may allow the user to choose an item or items from the list. The actual target control used may vary from platform to platform, but it will be able to provide the use intended by the designer in most cases.

Required Properties

None.

Recommended Properties

Property	Valid value	Description
g:editable	*true false custom	True if the list can take input (selection), false if not, custom if the list allows custom input.
g:listtype	*list	A single input list box
	range	A single range input list box
	multiple	A multiple input list box
	combo	A pull-down combo box

Optional Properties

Property	Valid value	Description
g:content	<i>Constant Model</i>	Set the content on the list by using a <constant> model (See Appendix A)

Events Supported

Event	Description
g:selectionChanged	Called when the selection changes
g:locatordblapped	Called when the locator is double tapped

Event	Description
g:actionperformed	Called when custom input is made to a list

Example:

```
<part id="top" class="G:TopContainer">
  <style>
    <property name="g:size">400,300</property>
    <property name="g:layout">border</property>
  </style>
  <!-- Normal -->
  <part id="list" class="G:List">
    <style>
      <property name="g:content">
        <constant model="list">
          <constant value="hello"/>
          <constant value="hola"/>
          <constant value="hallo"/>
        </constant>
      </property>
    </style>
  </part>
</part>
```

2.1.1.7 G:Table

A G:Table is a two-dimensional homogeneous representation of data, much like a spreadsheet. The data may be uneditable or editable, this property affects what type of control is actually rendered in the target language. The table may be selected as an entire row or edited by individual cell.

The G:Table contains either no children, in which case the column definitions are set from a constant model, or it contains children of type G:ColumnDef used to layout the table and determine the types of data held in each column and what type of control is used to render and edit that cell.

Required Properties

Property	Valid value	Description
g:editable	true false	Whether or not the table can be changed

Optional Properties

Property	Valid value	Description
g:content	<i>Constant model</i>	Set the table contents from a constant model consistent with the column definitions

Supported Events

Event	Description
g:locatortapped	Called when the locator is tapped over the table
g:selectionchanged	Called when the selected row changes
g:actionperformed	Called when the user double taps a table row
g:keytyped	Called when a key was typed when the table has focus

Example:

```

<part id="table" class="G:Table">
  <style>
    <property name="g:header">
      <constant model="list">
        <constant value="Header 1"/>
        <constant value="Header 2"/>
      </constant>
    </property>
    <property name="g:content">
      <constant model="table.rowMajor">
        <constant>
          <constant value="Row 1, Item 1"/>
          <constant value="Row 1, Item 2"/>
        </constant>
        <constant>
          <constant value="Row 2, Item 1"/>
          <constant value="Row 2, Item 2"/>
        </constant>
      </constant>
    </property>
  </style>
</part>

```

2.1.1.8 G:HeteroTable

A G:HeteroTable represents a heterogeneous table, that is a table where each individual cell can have a different data type or control.

A G:HeteroTable contains G:TableRow's, which in turn contain G:TableCell's to define the complete layout of the table. This sets the table definition and content at the same time. If scrolling of the table is needed, place the table into a G:Area whose g:scrollable attribute is set to true.

Example:

```

<part id="HeteroTableTest" class="G:TopContainer">
  <style>
    <property name="g:title">Hetero Table Test!</property>
    <property name="g:size">640,480</property>
  </style>
</part>

```

```

</style>
<part id="hTable" class="G:HeteroTable">
  <part class="G:TableRow">
    <part class="G:TableCell">
      <part id="r1c1" class="G:Text"/>
    </part>
    <part class="G:TableCell">
      <part id="r1c2" class="G:Text"/>
    </part>
  </part>
  <part class="G:TableRow">
    <part class="G:TableCell">
      <part id="r2c1" class="G:List"/>
    </part>
    <part class="G:TableCell">
      <part id="r2c2" class="G:Image"/>
    </part>
  </part>
</part>
</part>

```

2.1.1.9 G:TableRow

Wrapper container that is used to contain G:TableCells’s in the order they will be placed into the table. Be aware that target platforms may have restrictions on what type of controls may be placed into tables.

2.1.1.10 G:Tree

A G:Tree creates a hierarchical representation of data, while a G:Table creates a flat two dimensional view of data. G:Tree’s are built by nesting G:TreeItem’s to create the desired affect. The rendered representation of the tree will differ across platforms, but the concept of a hierarchical view will remain intact.

Optional Properties:

Property	Valid value	Description
g:content	<i>Constant model</i>	Build this tree from a constant model

Supported Events

Event	Description
g:actionperformed	Called when the user double clicks on a tree item
g:selectionchanged	Called when a selection is made in the tree
g:locatortapped	Called when the locator is tapped on the tree

Event	Description
g:treeexpanded	Called when a tree branch expanded
g:treecollapsed	Called when a tree branch collapsed

Example:

```
<part id="GTree" class="G:Tree">
  <style>
    <property name="g:editable">>false</property>
    <property name="g:content">
      <constant model="tree">
        <constant value="root">
          <constant value="rootleaf"/>
          <constant value="rootlimb">
            <constant value="rootlimbleaf"/>
            <constant value="rootlimbleaf2"/>
          </constant>
          <constant value="rootlimb2">
            <constant value="moo"/>
          </constant>
        </constant>
      </constant>
    </property>
  </style>
</part>
```

2.1.1.11 G:ToolBar

A G:ToolBar is simply a container that contains G:Button's and is placed below the menu bar. If this control is natively supported, it will be integrated into the UI, if it is not, a simulation will be rendered that closely matches the functionality of a native control. On platforms that have no clear analogue, this container may be ignored. It is good UI design to only use toolbars as a short cut and the functionality should be available from a different part of the UI to ensure that the omission of this part does not break the UI functionality.

Example:

```
<part id="ToolBar" class="G:TopContainer">
  <style>
    <property name="g:title">Tool Bars</property>
    <property name="g:size">640,480</property>
  </style>
  <part id="toolbar" class="G:ToolBar">
    <part id="tb1" class="G:Button">
      <style>
        <property name="g:text">ToolBar Button!</property>
      </style>
    </part>
    <part id="tb2" class="G:Button">
      <style>
        <property name="g:text">Better Button</property>
      </style>
    </part>
  </part>
```

```
</part>
</part>
```

2.1.2 Controls

2.1.2.1 G:Button

The G:Button is the heart of the set of input controls for the generic vocabulary. A G:Button is an abstract button that must take on properties to be rendered as the desired type of control. Any interface control that can be abstracted to accepting a single type of input (such as a mouse click) can be represented as a G:Button.

Required Properties

Property	Valid value	Description
g:text	<i>Text</i>	Text label for the button
g:buttontype	toggle	Creates a button that toggles between two states
	*push	Creates a simple push button
	submit	Creates a button to submit or confirm
	reset	Creates a button to reset a form
	radio	Creates a radio button
	checkbox	Creates a checkbox button

Recommended Properties:

Property	Valid value	Description
g:alternate-text	<i>Text</i>	Display alternate text for the button, usually a tool tip description
g:image-src	<i>Image Resource</i>	An Image to display with or as the button
g:selected	true false	If the button is currently selected (for toggle buttons)
g:enabled	true false	If the button is enabled or not

Optional Properties

Property	Valid value	Description
g:size	<i>x,y</i>	Size of the button
g:location	<i>x,y</i>	Location of upper-left corner of button
g:mnemonic	<i>Mnemonic</i>	Sets the mnemonic for the button

Supported Events

Event	Description
g:actionperformed	Called when the push button is pressed and released
g:selectionchanged	The selection changed event handler for two-state buttons

2.1.2.2 G:ButtonGroup

A Generic ButtonGroup is a logical grouping of buttons to form a mutual exclusion group. These are helpful for radio buttons and radio button menu items as they automate the code that ensures that only a single item in a group is selected at any time.

Required Properties

Property	Valid value	Description
g:group	<i>String</i>	A name for this group

Example of Buttons and ButtonGroups:

```
<part id="ButtonGroup" class="G:TopContainer">
  <part id="group1" class="G:ButtonGroup">
    <style>
      <property name="g:group">bestgroup</property>
    </style>
    <part id="b1" class="G:Button">
      <style>
        <property name="g:text">Checkbox1</property>
        <property name="g:buttontype">checkbox</property>
      </style>
    </part>
    <part id="b2" class="G:Button">
      <style>
        <property name="g:text">Checkbox2</property>
        <property name="g:buttontype">checkbox</property>
      </style>
    </part>
  </part>
  <part id="b3" class="G:Button">
```

```

<style>
  <property name="g:text">Checkbox3</property>
  <property name="g:buttontype">checkbox</property>
</style>
</part>
</part>

```

2.1.2.3 G:Image

A G:Image is a simple placeholder that references an outside image resource to be displayed to the user.

Required Properties

Property	Valid value	Description
g:image-src	<i>Resource ID</i>	Resource ID to find the requested image

Events Supported

Event	Description
g:locatormoved	Called when the mouse moves across the container
g:locatortapped	Called when the locator device is pressed and released
g:locatorentered	Called when the locator device enters the container
g:locatorexited	Called when the locator device leaves the container

Example:

```

<part id="monkey" class="G:TopContainer">
  <style>
    <property name="g:size">300,300</property>
  </style>
  <part id="mongbat" class="G:Image">
    <style>
      <property name="g:image-src">smiley17f.gif</property>
    </style>
  </part>
</part>

```

2.1.2.4 G:MultiMedia

A G:MultiMedia object represents any type of media that has a MIME type associated with it. This object can be used to represent text, images, sound, or video.

Required Properties

Property	Valid value	Description
g:content	<i>Resource ID</i>	Resource ID to find the requested object
g:mimetype	<i>MIME type</i>	MIME type for the associated media object

Currently no examples exist. This Part is not supported under any LiquidUI renderers for this release, support will be added next release.

2.1.2.5 G:TextBox

The G:TextBox generic control is the central means of displaying textual data to the user. These controls may be simple single line labels, or complex fully functional editors, depending on the target platform. Setting the properties on a G:TextBox part allows the mapping to choose the correct target control that most closely resembles the intentions of the UI designer.

Required Properties

Property	Valid value	Description
g:textboxtype	*textarea	Creates a simple multi-line text input box
	styled	Creates a text editor capable of handling styled text
g:editable	true *false	Whether or not the text area can be edited
g:mimetype	MIME type	Creates a styled editor capable of handling a specific MIME type

Recommended Properties

Property	Valid value	Description
g:text	<i>Text</i>	Simple text to be displayed to the user
g:content	<i>Resource ID</i>	How a styled text box is to be initialized. Styled boxes may be initialized from an outside resource.
g:size	<i>Dimension</i>	Size of the text box in pixels
g:rows	<i>Integer</i>	Number of rows of text
g:columns	<i>Integer</i>	Number of columns of text

Property	Valid value	Description
g:align	<i>Alignment</i>	Alignment of the text

Optional Properties

Property	Valid value	Description
g:font	<i>Font</i>	Set the font of the text
g:location	<i>Point</i>	Location of the upper-left corner of the text box
g:columns	<i>Integer</i>	Number of columns in the text control

Supported Events

Event	Description
g:onfocus	Called when the text element gains focus
g:focuslost	Called when the text element loses focus
g:locatorentered	Called when the locator enters the text area
g:locatorexited	Called when the locator exits the text area
g:keytyped	Called when a key is typed in text area
g:actionperformed	Called when the return key is pressed in the text area

Examples:

```
<part id="textarea" class="G:TextBox">
  <style>
    <property name="g:textboxtype">textarea</property>
    <property name="g:text">This is an uneditable TextArea</property>
    <property name="g:editable">>false</property>
  </style>
</part>
<part id="textpane" class="G:TextBox">
  <style>
    <property name="g:textboxtype">styled</property>
    <property name="g:mimetype">text/plain</property>
  </style>
</part>
```

```

    <property name="g:text">This is a TextPane displaying plain
text</property>
  </style>
</part>
<!--TextBox showing the content of a web page -->
<part id="edpane" class="G:TextBox">
  <style>
    <property name="g:textboxtype">styled</property>
    <property name="g:mimetype">text/html</property>
    <property name="g:content">http://www.vt.edu</property>
  </style>
</part>

```

2.1.2.6 G:TextField

The G:TextField provides the user with a single line simple text field to enter textual data.

Recommended Properties

Property	Valid value	Description
g:textfieldtype	*text	Specifies default text inputtr
	password	Specifies the output should be suppressed

Optional Properties

Property	Valid value	Description
g:columns	<i>Integer</i>	Number of visible columns of text
g:text	<i>Text</i>	Initial text displayed
g:enabled	*true false	Whether the user can input data or not
g:location	<i>x,y</i>	Location of the upper-left corner of the field
g:align	<i>Alignment</i>	Alignment of the text in the box

Supported Events

Event	Description
g:onfocus	Called when the text element gains focus
g:focuslost	Called when the text element loses focus
g:locatorentered	Called when the locator enters the text area

Event	Description
g:locatorexited	Called when the locator exits the text area
g:keytyped	Called when a key is typed in text area
g:actionperformed	Called when the return key is pressed in the text area

Example:

```

<part id="textfield" class="G:TextField">
  <style>
    <property name="g:text">This is a TextField</property>
  </style>
</part>
<part id="passfield" class="G:TextField">
  <style>
    <property name="g:textfieldtype">password</property>
    <property name="g:text">This is a TextField</property>
  </style>
</part>

```

2.1.2.7 G:ColumnDef

A G:ColumnDef is a column definition for a G:Table. It specifies such things as the type of data, control used to display and edit the data, column width, default cell height, font, color, etc. The table is then filled using a constant model keeping to the restrictions placed by the column definitions.

Required Properties

Property	Valid value	Description
g:columntype	text	The column will be used to display text
	number	The column will be used to display numerical data
	combo	The column will be edited by a combo box
	checkbox	The column will be rendered by a checkbox
	button	The column will be used to contain a button
	image	The column will be rendered with image data
	custom	The renderer and editor for the column will be provided, only stub code is generated.

Recommended Properties

Property	Valid value	Description
g:editable	true false	Whether or not the column's data can be edited. Default inherited from table.
g:header	<i>String</i>	Header for the column
g:resizable	true false	Whether the column can be resized

Optional Properties

Property	Valid value	Description
g:bgcolor	<i>Color</i>	Default background color for the column
g:fgcolor	<i>Color</i>	Default foreground color for the column
g:align	<i>Alignment</i>	Default alignment for items in the column
g:width	<i>Pixels</i>	Width of the column in pixels

Example:

```

<part id="table" class="G:Table">
  <part class="G:ColumnDef">
    <style>
      <property name="g:align">center</property>
      <property name="g:bgcolor">yellow</property>
      <property name="g:header">Header 1</property>
    </style>
  </part>
  <part class="G:ColumnDef">
    <style>
      <property name="g:align">center</property>
      <property name="g:bgcolor">yellow</property>
      <property name="g:header">Header 2</property>
    </style>
  </part>
  <part class="G:ColumnDef">
    <style>
      <property name="g:align">center</property>
      <property name="g:bgcolor">yellow</property>
      <property name="g:header">Header 3</property>
    </style>
  </part>
</part>

```

2.1.2.8 G:RowDef

A G:RowDef defines a set of attributes common to a row. Multiple G:RowDef's may be present in a G:Table definition, in that case as rows are added, the first row definition is applied to the first row, the second definition to the third row, and so on, looping back to the first row definition after the available row definitions have been exhausted. This is useful for such things as alternating cell backgrounds, fonts, colors, etc.

This part is not implemented currently in this release of Liquid UI.

2.1.2.9 G:TableCell

Simply a container for table data in a G:HeteroTable. Place desired parts into this wrapper and they will be formatted into the table.

2.1.2.10 G:MenuItem

A G:MenuItem is the only child member allowed for G:Menu's. They can represent all types of menu items such as check boxes, radio buttons, text with icons, etc.

Required Properties

Property	Valid value	Description
g:text	<i>Text</i>	Text displayed to the user for this menu item
g:menutype	*normal	Normal menu item
	checkbox	Checkbox menu item
	radio	Radio button menu item
g:selected	true *false	If the radio/check box menu item is selected
g:enabled	*true false	Whether or not the menu item is enabled

Recommended Properties

Property	Valid value	Description
g:image-src	<i>Resource ID</i>	Resource ID to an associated image
g:accelerator	<i>Accelerator</i>	Specify the accelerator for the menu item
g:mnemonic	<i>Mnemonic</i>	Sets the mnemonic for the menu

Supported Events

Event	Description
g:actionperformed	Called when the menu item is selected
g:selectionchanged	Called when the selected two-state menu item has changed

Example:

```
<part id="menu1" class="G:Menu">
  <style>
    <property name="g:text">The First Menu!</property>
    <property name="g:enabled">>true</property>
  </style>
  <part id="mi1" class="G:MenuItem">
    <style>
      <property name="g:text">First Menu Item.</property>
      <property name="g:enabled">>false</property>
    </style>
  </part>
  <part id="mi4" class="G:MenuItem">
    <style>
      <property name="g:menuitemtype">radio</property>
      <property name="g:selected">>true</property>
    </style>
  </part>
</part>
```

2.1.2.11 G:RangeSelector

A G:RangeSelector is a type of control that allows the user to select a value, usually numerical, between a specified range.

Required Properties

Property	Valid value	Description
g:minimum	<i>Integer</i>	Minimum value for the range
g:maximum	<i>Integer</i>	Maximum value for the range
g:extent	<i>Integer</i>	Set the extent (stepping) on the range

Recommended Properties

Property	Valid value	Description
g:rangetype	*slider	A slider control
	spinner	A spinner control (two buttons and a text box)

Property	Valid value	Description
g:size	<i>Dimension</i>	Size of the range selector
g:location	<i>Point</i>	Location of the range selector
g:content	<i>Integer</i>	Current value for the range selector

Supported Events

Event	Description
g:selectionchanged	Called when the selected range has changed

Example:

```
<part id="rangetop" class="G:TopContainer">
  <style>
    <property name="g:title">Range Selector Test</property>
    <property name="g:size">640,480</property>
  </style>
  <part id="slider" class="G:RangeSelector">
    <style>
      <property name="g:rangetype">slider</property>
      <property name="g:minimum">0</property>
      <property name="g:maximum">1000</property>
      <property name="g:extent">100</property>
      <property name="g:size">300,80</property>
    </style>
  </part>
  <part id="spinner" class="G:RangeSelector">
    <style>
      <property name="g:rangetype">spinner</property>
      <property name="g:maximum">100</property>
      <property name="g:minimum">0</property>
      <property name="g:extent">1</property>
    </style>
  </part>
</part>
```

2.1.2.12 G:PromptInput

A G:PromptInput control is a pre-defined control, usually a modal dialog box, that wants some sort of immediate input from the user. Examples include the file selection dialog and confirmation box.

Currently only File Choosers are supported in this release.

Required Properties

Property	Valid value	Description
----------	-------------	-------------

Property	Valid value	Description
g:prompttype	*file	Creates a file selection dialog
	confirm	Creates a confirmation box
	message	Creates a message box
	error	Creates an error box
	question	Creates a question box
g:title	<i>Text</i>	Sets title of the prompt
g:text	<i>Text</i>	Sets the text string on the prompt input

Supported Events

Event	Description
g:actionperformed	Called when the prompted input has been made

Example:

```
<part id="top" class="G:TopContainer">
  <style>
    <property name="g:size">500,500</property>
  </style>
  <part id="file" class="G:PromptInput">
    <style>
      <property name="g:text">This is the Open Button</property>
      <property name="g:prompttype">file</property>
    </style>
  </part>
</part>
```

2.1.3 Text

The properties of text data and the way it can be formatted and embedded in other objects warrant its own discussion in this section. The generic text object is special in that it can contain different formats as well as embedded non-textual data, such as images and sound. Text is also amorphous – it can surround and envelope other objects and can be represented in many different forms, such as HTML.

2.1.3.1 G:Text

The G:Text object represents a stream of text and other embedded data. It does not map to any specific control, merely to a format suitable for display in the target environment.

Required Properties

Property	Valid value	Description
g:text	<i>Text</i>	The text string segment
g:mimetype	<i>Mime Type</i>	The Mime type representing this object and all children of this object. Default: text/html

Optional Properties

Property	Valid value	Description
g:font	<i>Font</i>	Font style for the text segment
g:bold	true *false	Bold style
g:italic	true *false	Italic style
g:underline	true *false	Underline style
g:strike	true *false	Strikethrough style
g:break	line	Line break after segment
	paragraph	Paragraph break before segment
	page	Page break after segment

About Fonts:

Fonts in the generic vocabulary are handled like fonts in Java, as a hyphen-separated triple. The first value is the font family, which can be a specific font family, such as Times, Ariel, Helvetica, or a generic family such as courier, serif or sansserif. The second value is the font style, one of either BOLD, ITALIC or BOLDITALIC. The third value is the integer point size of the font. For example: Helvetica-bold-12 will produce a 12-point Helvetica Bold font.

Available Generic Font Families:

Generic Family	Specific Members of Family
serif	Times New Roman
sansserif	Arial, Helvetica
monospaced	Courier

Available Font Styles

- bold
- italic
- bolditalic

Examples of Text:

Label:

```
<part id="label" class="G:Text">
  <style>
    <property name="g:text">This is a label</property>
  </style>
</part>
```

Formatted Text:

```
<part id="P" class="G:Text">
  <style>
    <property name="g:break">paragraph</property>
    <property name="g:text">This is a Paragraph break.</property>
    <property name="g:bgcolor">47,255,255</property>
    <property name="g:fgcolor">red</property>
  </style>
  <part id="internalBlock" class="G:Text">
    <style>
      <property name="g:italic">>true</property>
      <property name="g:text"> This is an internal block of
text.</property>
    </style>
    <part class="G:Text">
      <style>
        <property name="g:text"> A Sub Part of the Internal
Block.</property>
        <property name="g:fgcolor">green</property>
      </style>
    </part>
  </part>
  <part id="lastBlock" class="G:Text">
    <style>
      <property name="g:text">. Outside the Internal Block</property>
      <property name="g:bold">>true</property>
    </style>
  </part>
</part>
```

Renders to:

This is a Paragraph break. This is an internal block of text. A Sub part of the Internal Block.
Outside the Internal Block.

Properties on embedded text cascade to children, which can override them with another property.

3 Layout

Layout of the UI can be difficult to handle in a generic way since different target platforms and languages handle it differently. HTML must be laid out by the designer inline with the document itself, while Java/Swing, MFC, Motif, and etc. have the layout handled by a layout manager attached to the containers themselves. Layout techniques are similar enough across platforms that allowing for the most commonly used layouts to be handled generically would not create great difficulty and in fact would greatly enhance the usability of the generic vocabulary by removing the need for language-specific code in the generic document.

Layout can be handled in the OO-based languages by setting a property on the container that will hold the parts to be laid out. Layout for markup languages, on the other hand, requires a secondary layout engine to add/remove/reorder parts and restructure the UIML to physically layout the parts in the language specific UIML. This can most likely be done in something as simple as XSL and the resultant UIML passed to the backend renderer to generate the target document, be it HTML/JSP, HTML/PHP, etc.

Layout is set in the generic vocabulary via the `g:layout` property on containers that support layout (currently the `G:TopContainer` and `G:Area` containers).

Layouts Supported:

Layout Name	Layout Property Value	Layout Description
Flow Layout	flow	Standard layouts for HTML and Java sub-elements. Components are laid out according to the <code>g:layoutalign</code> property as either Left To Right, Centered, or Right to Left
Border Layout	border	Components are anchored to an area of the area they are laid out in according to the <code>g:layoutborder</code> property
Grid Layout	grid	A layout scheme that lays out components into a grid of equally proportioned grid cells unless weight values are set on specific rows or columns to alter the proportion. This layout scheme is the most powerful and complex

Layout Name	Layout Property Value	Layout Description
Absolute Layout	absolute	Not actually a layout, each component must have a g:size and g:location set to position each component absolutely into the parent container

Flow Layout Properties

Property Name	Valid Values	Description
g:layoutalign	left, right, center	Sets the direction the components are laid out from

Border Layout Properties

Property Name	Valid Values	Description
g:layoutborder	north, south, east, west, center, northeast, northwest, southeast, southwest	The part of the parent container to anchor to

Grid Layout Properties

Property Name	Valid Values	Description
g:layoutwidth	<i>Integer</i> , relative, remainder	Number of columns in the grid to span. Relative means this part will span all columns until the last one. Remainder means this part will span all columns to and including the last column in the grid.
g:layoutheight	<i>Integer</i> , relative, remainder	Number of rows in the grid to span. Relative means this part will span all rows until the last one. Remainder means this part will span all rows to and including the last column in the grid. Only parts in the left-most column may span down rows.
g:layoutweightx	<i>Double</i>	Relative weight for this column. Only the highest weightx in the column is used.
g:layoutweighty	<i>Double</i>	Relative weight for this row. Only the highest weighty for the row is used.

This layout has the same functionality as the Java [GridBagLayout](#) without the use of the anchor, gridx, gridy, padding or fill properties, although these may be added in the future. The grid always fills in both directions to take up all available space in the parent container. **Please note that this functionality is experimental for HTML.**

Java Specific Properties for Grid Layout

Property Name	Valid Values	Description
g:layoutcols	<i>Integer</i>	Number of columns in a GridLayout
g:layoutrows	<i>Integer</i>	Number of rows in a GridLayout
j:gridx	<i>Integer</i>	Absolute grid column position
j:gridy	<i>Integer</i>	Absolute grid row position
j:anchor	<i>Text</i>	See: GridBagLayout
j:insets	<i>Java Insets</i>	See: GridBagLayout
j:ipady	<i>Integer</i>	See: GridBagLayout
j:ipadx	<i>Integer</i>	See: GridBagLayout
j:fill	<i>Text</i>	See: GridBagLayout

Absolute Positioning Properties

Property Name	Valid Values	Description
g:size	<i>Dimension</i>	Size of the component
g:location	<i>Point</i>	Location of the upper-left corner of the component

4 Borders

As with layout, borders pose their own unique set of issues when attempting to describe them in a generic fashion. It is important to include as comprehensive of an implementation of borders in the generic vocabulary as possible. Not all border techniques are supported in all platforms, but many are related. It is possible to create a hierarchy of compatibility for borders. When the UI designer chooses a border technique not supported by a specific target platform, the renderer attempts to choose the closest match based on those that are supported. Since borders mostly serve as a means to separate content and are supported in all targets that render visually, although not all styles, it would not be unreasonable for the generic vocabulary to define a wide-range of styles while notifying the user that not all are supported on all targets and that a reasonable alternative will be chosen for those targets that do not support certain styles.

4.1 *Supported Generic Borders*

- Title Borders
- Recessed Borders
- Beveled Borders
- Simple Line Borders

Caveats

- Currently titled borders are not supported compounded with another type of border

Appendix A – Constants

Color

black	red	green	blue	magenta
cyan	yellow	white	gray	

Custom RGBA value examples:

90, 145, 67	20, 45, 201	190, 80, 10
-------------	-------------	-------------

Alignment

left	center	right
------	--------	-------

Constant Models

Model	Example
list	<pre><constant model="list"> <constant value="one"/> <constant value="two"/> <constant value="three"/> </constant></pre>
table.rowMajor	<pre><constant model="table.rowMajor"> <constant> <constant value="Row 1, Item1"/> <constant value="Row 1, Item 2"/> </constant> <constant> <constant value="Row 2, Item 1"/> <constant value="Row 2, Item 2"/> </constant> <constant> <constant value="Row 3, Item 1"/> <constant value="Row 3, Item 2"/> </constant> </constant></pre>

Model	Example
tree.preorder	<pre><constant model="tree"> <constant value="root"> <constant value="rootleaf"/> <constant value="rootlimb"> <constant value="rootlimbleaf"/> <constant value="rootlimbleaf2"/> </constant> <constant value="rootlimb2"> <constant value="moo"/> </constant> </constant> </constant></pre>

Resource ID's

In this release of LiquidUI, Resource ID's simply refer to a location to get the data needed by the part. The most common locations would be URI's and Files. In future revisions of the generic vocabulary the concept of these resources will be abstracted into resource identifiers that can be linked to specific (even multiple) sources by the designer. For this release the designer will refer to the resource by file name or URI, whichever is applicable for the specific widget.

Appendix B – Known Issues and Future Enhancements

These are known issues with the current version of the Generic Vocabulary and the LiquidUI 3.0 release.

Known Issues:

- It is not possible to set title borders with other border types
- It is not possible to set g:scrollable with any g:areatype except for the default
- G:Tree's are not supported under HTML
- G:RangeSelector is not supported under HTML
- Grid layouts for HTML does not respond well to malformed layouts, please be certain your layout conforms to the restrictions outlined in this as well as the Java GridBagLayout documentation
- HTML G:MenuItem's do not support the checkbox or radio types
- Under Java, G:Text creates a label and does not support nested children. This is a priority fix and will be addressed as soon as possible.

Future Enhancements

- Generic Vocabulary and Functionality
 - Currently Overriding Generic properties with language-specific properties is not supported
 - Type g:hyperlink for G:Text
 - Support Column-Major Row Model for Tables
 - Move away from g:text property to use G:Text children for Text components
 - Create defined namespaces for generic and target-specific vocabularies to allow overriding of generic properties with target-specific properties
 - Generalize Fonts and Text Properties
- Java
 - JList support for custom data sources
 - Allow g:scrollable with specific types of G:Area's
 - Support g:header property for HeteroTable Rows
 - Rewrite G:Text component to recursively traverse child tree applying formatting to string subsets.
- HTML

- Allow Split, tabbed panes
- Labels for buttons and other controls
- G:Tree Support
- G:RangeSelector support
- Add GridLayout absolute grid positioning support to Layout manager
- Images in buttons via embedded IMG tag with alternate text