VCXL 1.0 Candidate Release

By Amir Mimran

Last update: 31/08/2006

SAP NetWeaver Visual Composer

VCXL 1.0 Specification

Executable Specification of Graphic User Interfaces

History

Version	Date	Comment
1.0	2008-11-1	
	2009-12-10	Revised

Table of Contents

This document describes the *Visual Composer Extension Descriptor language* (VCXL).

VCXL is a declarative specification language designed to serve as the common ground between NetWeaver Visual Composer 7.0 Runtime and a 3rd party Flex component that is to be embedded inside a Visual composer application.

VCXL acts as the descriptor for the component, in terms of:

- *General properties* - application's filename, description, height/width dimensions etc...
- *Configurable properties* – default properties of the developed control
- *Data flow* – what data does this control receive, and from which type?

Document Structure

The documentation is organized according to the *VCXL* component structure. In particular:

Part I: The Root Component (the "Component" tag)

Part II: Configurable Properties (the "Properties" tag)

Part III: Data (the "Data" tag)

Attribute types

The attribute types used in a *VCXL* document are defined below:

Attribute type	Description
bool	A Boolean value, which can be either true or false.
enum	An enumerated value. That is, a value selected out of a predefined set of possible values..
float	A floating-point number
int	An integer number
name	An *XGL* element name. The *name* is guaranteed to be unique, but only under the immediate parent of the element. Therefore, the *name* can be used for referencing the element only in the context of its parent element.
str	A characters string of any length, which contains only the following characters: A-Z, a-Z, 0-9 and underscore.
variant	A literal value in any of the primitive *XGL* data types (§**Error! Reference source not found.**)

Component

Every *VCXL* component specification is written in a separate *VCXL* document. The *VCXL* document is self-contained and provides all the information necessary in order to embed and execute the component. The *VCXL* component has the following structure:

```
<?xml version="1.0"?>
<vcx:Component component-attrs>
   <vcx:Properties>
       Properties group-decl nested
           control-decl many
   </vcx:Properties>
   <vcx:Data>
       InputPort-decl
           Port-decl
               Field-decl many
       OutputPort-decl
           Port-decl
               Field-decl many
   </vcx:Data>
</vcx:Component>
```

The *Component* element has the following attributes:

Attribute	Type	Description
name	str	**Required**. The component display name in the Visual Composer design time
id	namespace	**Required**. the component's id, e.g. com.sap.extenral.gauge
type	str	**Required**. Type of component (e.g. "flex2" or "xcelsius")
file	str	**Required.** Filename of the component that this VCXL describes.
version	str	**Required.** The component version. (in a NN.NN format, e,g 1.01)
width	int	**Required.** Default width of component in pixels
height	int	**Required.** Default height of component in pixels
description	str	**Required.** Long description of the component
layoutImage	str	Optional. Image for the Visual Composer layout Pane
icon16	str	Optional. Icon Image for Design Board(16x16)
icon32	str	Optional. Icon Image for Design Board (32x32)
resizable	bool	Optional. Whether the component can be resized. Default is false.
withAspect	bool	Optional. Determines if the Height/Width ratio should be

		enforced in cases where the component is resizable. Default is true
creator	str	Optional. Name of this component creator

In addition the Component elements should contain the following namespace attribute:

```
xmlns:vcx="http://www.sap.com/visualcomposer/2008/vcxl"
```

Configurable Properties

This part of the VCXL defines which properties does the component accepts, and which UI control will be used in Visual Composer design time to change it.

```
..
<vcx:Properties>
   Prop-group-decl
       property-decl many
</vcx:Properties>
..
```

Properties Group

A property group defines a set of configurable properties that have a common use. For instance a property group with the label “Category Axis” for a chart component will contain properties such as: “Axis Label”, “Axis Field”, “Interval between ticks” etc…

The general structure of a properties group declaration is as follows:

```
Prop-group-decl:
<vcx:PropGroup label>
  property-decl many
```

Attribute	Type	Description
label	str	**Required**. The properties group’s display name in the Visual Composer design time

Property

Defines the configurable properties for the component along with the UI control that will be used to change them on design-time.

A property structure is as follows:

```
property-decl: one of
 <vcx:Inputfield prop-attrs control-attrs/>
 <vcx:Checkbox prop-attrs control-attrs/>
 <vcx:Spinner prop-attrs control-attrs/>
 <vcx:ColorPicker prop-attrs control-attrs/>
 enum-decl

enum-decl: one of
 <vcx:ComboBox prop-attrs type>
   <vcx:Choice choice-attrs/>many
 <vcx:ComboBox>

 <vcx:DropDown prop-attrs type>
   <vcx:Choice choice-attrs/>many
 </vcx:DropDown>
```

Each property can contain the following attributes:

Attribute	Type	Description
name	str	**Required.** The *name* is the application-specific field identifier.
label	str	**Required.** The property label in design time.
init	variant	Optional. Default value of the property,

Control Types

InputField Control

Purpose

InputField is a single-line input control supporting all data types.

Classification

Data type: ☑ String ☑ Number ☑ Boolean ☑ Date ☑ Time

Attributes

Attribute	Type	Description
type	str	**Required**. Type of field, where type could be one of the following: ***bool, float, int, date, time*** and ***str*** (see §1.2)

Checkbox Control

Purpose

A *CheckBox* is a dual-state selection control used for setting or clearing *Boolean* values.

Classification

Data type: ☐ String ☐ Number ☑ Boolean ☐ Date ☐ Time

Spinner control

Purpose

A *Spinner* is a *numeric* range control with up/down buttons for stepping through the values in the range.

Classification

Data type: ☐ String ☑ Number ☐ Boolean ☐ Date ☐ Time

Attributes

Attribute	Type	Description
min	float	**Required**. Minimal number in range
max	float	**Required**. Maximal number in range
step	float	Optional. Interval between values

ColorPicker control

Purpose

A *ColorPicker* is a input field control with a palette icon for choosing color values in hexadecimal representation, for example: #FFFFFF for white.

Classification

Data type: ☑ String ☐ Number ☐ Boolean ☐ Date ☐ Time

Dropdown control

DropDown is a selection control consisting of a dropdown list and a selection field. The list presents the possible values that a user can select and the selection field displays the currently selected value.

The DropDown's value is restricted and cannot be equal to values that are not defined in the enumeration

Classification

Data type: ☑ String ☑ Number ☑ Boolean ☑ Date ☑ Time

Attributes

Attribute	Type	Description
type	str	**Required**. Type of the drop-down entries' value, where type could be one of the following: ***bool, , float, int, date, time*** and ***str*** (see §1.2)

The structure of a *Choice* control is as follows:

```
combobox-decl:
<vcx:DropDown prop-attrs type>
  <vcx:Choice choice-attrs/>many
```

Note that the DropDown tag contains the property attributes (§2.2).

The required attributes for the *Choice's* tags themselves are:

Attribute	Type	Description
text	str	**Required**. The Display Text for the drop-down entry
value	variant	**Required**. The value for the drop-down entry

Combo Box Control

Purpose

ComboBox is a combination of *DropDown* and *InputField* controls. Like a *DropDown* control it consists of a dropdown list and a selection field. The list presents the possible values that a user can select and the selection field displays the current value, but - Unlike a *DropDown* control, the selection field is an **editable** *InputField* that can be used to enter values not available in the list.

Classification

Data type: ☑ String ☑ Number ☑ Boolean ☑ Date ☑ Time

Attributes

Attribute	Type	Description
type	str	**Required**. Type of the drop-down entries' value, where type could be one of the following: ***bool, , float, int, date, time*** and ***str*** (see §1.2)

The structure of a *Choice* control is as follows:

```
combobox-decl:
<vcx:ComboBox prop-attrs type>
  <vcx:Choice choice-attrs/>many
```

Note that the *ComboBox* tag contains the property attributes (§2.2).

The required attribute for the *Choice* tags is:

Attribute	Type	Description
value	variant	**Required**. The value for the drop-down entry

Note:

The ComboBox dropdown list does not use an enumeration of [Display Text : Value] pairs, but a list of possible values.

Data

This part of the VCXL defines which data does the component expect to receive and which data it dispatches out to the embedding application.

```
<vcx:Data>
   inports-decl
        Port-decl many
            Field-decl many
   outports-decl
        Port-decl many
            Field-decl many
</vcx:Data>
```

The “Data” section is divided to two main sections: “Inports” and “Outports”,

Input and output ports

Ports are named connection points through which parameters flow between the custom component and the embedding application during execution. Each port is associated with a set of *Field*s, which defines the parameters that are sent or received through the port. Ports are uni-directional and are therefore divided into:

- I*nports* (input ports), used for sending input information from the main application to the component.
- O*utports* (output ports), used for sending output information from the component to the main application.

```
inports-decl:
<vcx:Inports>
```

```
        port-decl many
            field-decl many
</vcx:Inports>

outports-decl:
<vcx:Outports>
        port-decl many
            field-decl many
</vcx:Outports>
```

Ports types

There are two types of ports, *SingletonPort*, and *ArrayPort,* that can be used inside the *Inports* and *Outports* entities. The two must contain the name attribute:

Attribute	Type	Description
name	str	**Required**. The *name* attribute uniquely identifies the port within the scope of the component.

```
port-decl: one of
   singleton-port-decl
   array-port-decl
```

Singleton Port

A *singleton* port holds zero or one object.

```
singleton-port-decl:
<vcx:SingletonPort name>
   field-decl many
</vcx:SingletonPort>
```

Array Port

An *array* port holds an ordered list of zero or more objects.

```
array-infoset:
<vcx:ArrayPort name>
   field-decl many
</vcx:ArrayPort>
```

Field declaration

*Field*s are entities that define the parameters that are sent to and from the component through the Input and Output ports (§3.2)

A field is declared by using a tag name corresponding to its data type:

```
field-decl: one of
<vcx:str field-attrs/>    (string)
<vcx:int field-attrs/>    (integer number)
<vcx:float field-attrs/>  (floating point number)
<vcx:bool field-attrs/>   (Boolean)
<vcx:time field-attrs/>   (time)
<vcx:date field-attrs/>   (date)

field-attrs:
name, init
```

Fig. 1: Field declaration

Attribute	Type	Description
name	name	**Required**. The *name* attribute uniquely identifies the field within the scope of its containing port declaration.
init	variant	Optional. defines the initial (default) value of the field

Example of a VCXL Document

```
<vcx:Component id="com.sap.test" type="flex2" name="Test Grid" file="Test.swf" width="800"
     height="500" resizable="true" withAspect="true" description="Test control" creator="John Doe"
     version="1.0" layoutImage="preview.jpg" icon16="icon16.jpg" icon32="icon32.jpg"
     xmlns:vcx="http://www.sap.com/visualcomposer/2008/vcxl">
     <vcx:Properties>
        <vcx:PropGroup label="Properties">
           <vcx:ComboBox type="str" name="title" label="Title" init="Value1">
              <vcx:Choice text="Title 1" value="Title 1"></vcx:Choice>
              <vcx:Choice text="Title 2" value="Title 2"></vcx:Choice>
              <vcx:Choice text="Title 3" value="Title 3"></vcx:Choice>
           </vcx:ComboBox>
           <vcx:ColorPicker name="buttonColor" label="Button color" init="#222222" />
           <vcx:Spinner name="progress" label="Progress" min="0" max="100" init="0" step="1" />
        </vcx:PropGroup>
        <vcx:PropGroup label="Other Properties">
           <vcx:TextInput type="date" name="date" label="Date" />
           <vcx:TextInput type="time" name="time" label="Time" />
        </vcx:PropGroup>
     </vcx:Properties>
     <vcx:Data>
        <vcx:Inports>
           <vcx:ArrayPort name="fillData">
              <vcx:str name="myString" />
              <vcx:int name="myNumber" />
              <vcx:float name="myFloat" />
              <vcx:bool name="myBoolean" />
              <vcx:time name="myTime" />
              <vcx:date name="myDate" />
           </vcx:ArrayPort>
           <vcx:SingletonPort name="fillSingleton">
              <vcx:str name="myString" />
              <vcx:int name="myNumber" />
              <vcx:float name="myFloat" />
              <vcx:bool name="myBoolean" />
              <vcx:time name="myTime" />
              <vcx:date name="myDate" />
           </vcx:SingletonPort>
        </vcx:Inports>
        <vcx:Outports>
           <vcx:ArrayPort name="arrayChanged">
              <vcx:str name="myString" />
              <vcx:int name="myNumber" />
              <vcx:float name="myFloat" />
              <vcx:bool name="myBoolean" />
              <vcx:time name="myTime" />
              <vcx:date name="myDate" />
           </vcx:ArrayPort>
           <vcx:SingletonPort name="singletonChanged">
              <vcx:str name="myString" />
              <vcx:int name="myNumber" />
              <vcx:float name="myFloat" />
              <vcx:bool name="myBoolean" />
              <vcx:time name="myTime" />
              <vcx:date name="myDate" />
           </vcx:SingletonPort>
        </vcx:Outports>
     </vcx:Data>
</vcx:Component>
```

Copyright