



# Table of Contents

Document Structure .....	3
Attribute types .....	3
Component .....	4
Configurable Properties.....	5
Properties Group .....	5
Property .....	6
Control Types .....	6
InputField Control .....	6
Checkbox Control .....	7
Spinner control .....	7
ColorPicker control .....	7
Dropdown control .....	7
Combo Box Control .....	8
Data.....	9
Input and output ports.....	9
Ports types .....	10
Singleton Port.....	10
Array Port .....	10
Field declaration .....	11
Example of a VCXL Document.....	12
Copyright .....	13

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 3<sup>rd</sup> 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:

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

## 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-declnested
    control-declmany
  </vcx:Properties>
  <vcx:Data>
    InputPort-decl
    Port-decl
    Field-declmany
    OutputPort-decl
    Port-decl
    Field-declmany
  </vcx:Data>
</vcx:Component>
```

The *Component* element has the following attributes:

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

Attribute	Type	Description
label	str	<b>Required.</b> 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
<b>name</b>	str	<b>Required.</b> The <i>name</i> is the application-specific field identifier.
<b>label</b>	str	<b>Required.</b> The property label in design time.
<b>init</b>	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
<b>type</b>	str	<b>Required.</b> Type of field, where type could be one of the following: <i>bool, float, int, date, time</i> and <i>str</i> (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
<b>min</b>	float	<b>Required.</b> Minimal number in range
<b>max</b>	float	<b>Required.</b> Maximal number in range
<b>step</b>	float	Optional. Interval between values

## ColorPicker control

### Purpose

A *ColorPicker* is an 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
<b>type</b>	str	<b>Required.</b> Type of the drop-down entries' value, where type could be one of the following: <i>bool, , float, int, date, time</i> and <i>str</i> (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
<b>text</b>	str	<b>Required.</b> The Display Text for the drop-down entry
<b>value</b>	variant	<b>Required.</b> 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
<b>type</b>	str	<b>Required.</b> Type of the drop-down entries' value, where type could be one of the following: <i>bool, , float, int, date, time</i> and <i>str</i> (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	<b>Required.</b> 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 *Fields*, which defines the parameters that are sent or received through the port. Ports are uni-directional and are therefore divided into:

- *Inports* (input ports), used for sending input information from the main application to the component.
- *Outports* (output ports), used for sending output information from the component to the main application.

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

```

    port-declmany
      field-declmany
</vcx:Inports>

outports-decl:
<vcx:Outports>
  port-declmany
  field-declmany
</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
<b>name</b>	str	<b>Required.</b> The <i>name</i> 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-declmany
</vcx:SingletonPort>

```

## Array Port

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

```

array-infoset:
<vcx:ArrayPort name>
  field-declmany
</vcx:ArrayPort>

```

## Field declaration

*Fields* 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**

<b>Attribute</b>	<b>Type</b>	<b>Description</b>
<b>name</b>	name	<b>Required.</b> The <i>name</i> attribute uniquely identifies the field within the scope of its containing port declaration.
<b>init</b>	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

© Copyright 2009 SAP AG. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.

Microsoft, Windows, Excel, Outlook, and PowerPoint are registered trademarks of Microsoft Corporation.

IBM, DB2, DB2 Universal Database, System i, System i5, System p, System p5, System x, System z, System z10, System z9, z10, z9, iSeries, pSeries, xSeries, zSeries, eServer, z/VM, z/OS, i5/OS, S/390, OS/390, OS/400, AS/400, S/390 Parallel Enterprise Server, PowerVM, Power Architecture, POWER6+, POWER6, POWER5+, POWER5, POWER, OpenPower, PowerPC, BatchPipes, BladeCenter, System Storage, GPFS, HACMP, RETAIN, DB2 Connect, RACF, Redbooks, OS/2, Parallel Sysplex, MVS/ESA, AIX, Intelligent Miner, WebSphere, Netfinity, Tivoli and Informix are trademarks or registered trademarks of IBM Corporation.

Linux is the registered trademark of Linus Torvalds in the U.S. and other countries.

Adobe, the Adobe logo, Acrobat, PostScript, and Reader are either trademarks or registered trademarks of Adobe Systems Incorporated in the United States and/or other countries.

Oracle is a registered trademark of Oracle Corporation.

UNIX, X/Open, OSF/1, and Motif are registered trademarks of the Open Group.

Citrix, ICA, Program Neighborhood, MetaFrame, WinFrame, VideoFrame, and MultiWin are trademarks or registered trademarks of Citrix Systems, Inc.

HTML, XML, XHTML and W3C are trademarks or registered trademarks of W3C®, World Wide Web Consortium, Massachusetts Institute of Technology.

Java is a registered trademark of Sun Microsystems, Inc.

JavaScript is a registered trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape.

SAP, R/3, SAP NetWeaver, Duet, PartnerEdge, ByDesign, SAP Business ByDesign, and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and other countries.

Business Objects and the Business Objects logo, BusinessObjects, Crystal Reports, Crystal Decisions, Web Intelligence, Xcelsius, and other Business Objects products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of Business Objects S.A. in the United States and in other countries. Business Objects is an SAP company.

All other product and service names mentioned are the trademarks of their respective companies. Data contained in this document serves informational purposes only. National product specifications may vary.

These materials are subject to change without notice. These materials are provided by SAP AG and its affiliated companies ("SAP Group") for informational purposes only, without representation or warranty of any kind, and SAP Group shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP Group products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.