Generic Import - Architect - Instantiation

Description: This document explains how to use Generic Import to process Architect instantiation.

Keywords:

Last Revision Date: 07/11/13

www.pcvuesolutions.com

FRANCE - Paris
ARC Informatique
Head Office

GERMANY - Munich
PcVue GmbH

ITALY - Milan
PcVue Srl

UK - London
Control Technology International

USA - Boston
PcVue Inc

SINGAPORE - Singapore
PcVue Sea

MALAYSIA – Kuala Lumpur
PcVue Sdn Bhd

CHINA - Shangai
PcVue China

JAPAN - Nagoya
PcVue Japan

ARC Informatique
ISO 9001 : 2008
ISO 14001 : 2004
certified

Authorization

	Name	Stamp	Date
Written by	Pascale Gallardo		06/11/2013
Checked by			
Authorized by			

Revision history

Revision	Author	Action	Editing	Date	Distribution
1.0a	PG	Creation		06/11/13	
1.0b					
1.0c					
1.0d					

SUMMARY

1 INTRODUCTION 6

1.1 Object 6
1.2 Document description 6

2 CREATING AND EDITING TOPOLOGY ELEMENTS 7

2.1 Interfaces 7
2.1.1 TopologyElements (Collection) 7
2.1.2 TopologyElement (Item) 7

2.2 XML sample file an result 8
2.2.1 XML File 8
2.2.2 Result 8

3 CREATING AND EDITING TEMPLATE INSTANCES 9

3.1 Interfaces 9
3.1.1 Instances (Collection) 9
3.1.2 Instance (Item) 9

3.2 XML sample file an result 10
3.2.1 XML File 10
3.2.2 Result 11

4 SETTING INPUT VALUES AND INPUT PARAMETER VALUES 12

4.1 Interfaces 12
4.1.1 InstanceInputValues (Collection) 12
4.1.2 InstanceInputValue (Item) 12
4.1.3 InstanceParameterInputValues (Collection) 13
4.1.4 InstanceParameterInputValue (Item) 13

4.2 XML sample file an result 14
4.2.1 XML File 14
4.2.2 Result 14

5 SELECTING OR UNSELECTING ITEMS IN A TEMPLATE INSTANCE 15

5.1 Interfaces 15
5.1.1 TemplateObjectInstances (Collection) 15
5.1.2 TemplateObjectInstance (Item) 15

5.2 XML sample file an result 16
5.2.1 XML File 16
5.2.1 Result 16

6 GENERATING ARCHITECT APPLICATION AND SAVING CONFIGURATION 17

6.1 XML File 17
6.2 Using XmlImporter 17

1 Introduction

1.1 Object

This document describes how the XML generic import format will be used to import the instances of templates in Architect and the resolution of parameterized properties.

In this document only the instantiation of the Architect will be discussed.

1.2 Document description

Currently the functions that are possible with the Architect are:

- Creating and editing topology elements
- Creating and editing template instances
- Setting input values and input parameter values
- Selecting or unselecting items in a template instance
- Generating Architect application and saving configuration

For each of these functions, this document will detail the use of the interfaces available in the generic import.
This document provides also, for each function, a sample XML file and the result obtained if importing this sample file.

Note on the topological tree:
The topological tree is always created by the Architect with the name of the project. It is not currently possible to change it by the generic import.
In the XML file, it can be referenced by the keyword '(default)' lowercase.
In the configuration HMI of the Architect, it appears as the project name.

2 Creating and editing topology elements

2.1 Interfaces

2.1.1 TopologyElements (Collection)

Syntax	<Collection type="TopologyElements " </Collection>		
Attributes	type	TopologyElements	*Collection of TopologyElement*
	id	#InstId1.#InstId2 ….#InstIdN-1	*Reference to a full parent instance path (tree, template instance or topology level). '(default)' id refers to the root (tree)*
Parents	Group		
Children	Item	TopologyElement	
Prerequisites	None		

2.1.2 TopologyElement (Item)

Syntax	<Item id="#InstId"> </Item>		
Attributes	type		*Omitted*
	Id	# InstId	*Name of topology level*
Parents	Group		
	Collection	TopologyElements	
Children	Property		
	Command		
Prerequisites	None		

Properties

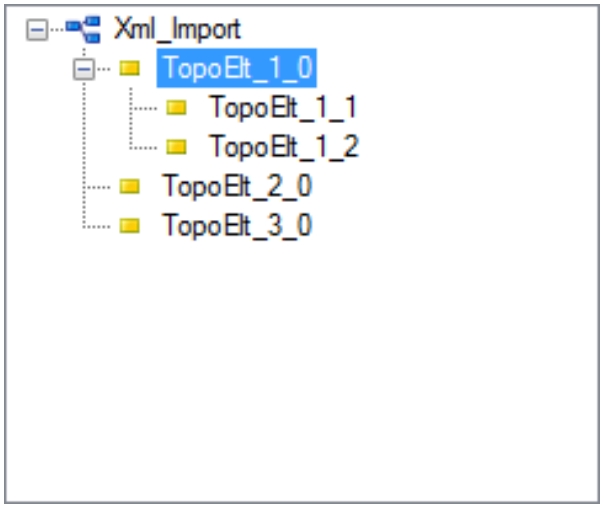
Branch	*The branch that will be used during application generation for all configuration elements that the instance contains.*
<Property id="Branch"> #Branch </Property>	
#Branch	String of characters : 0-255 characters
Description	*A description of the instance. Only used within the Application Architect and has no effect on the application itself.*
<Property id="Description"> #Description </Property>	
#Description	String of characters : 0-255 characters

2.2 XML sample file an result

2.2.1 XML File

```
Import>
  <Group customRef="Topology">
    <!-- Adding 3 Topology Element on tree (id of tree is 'Default') -->
    <Collection type="TopologyElements" id="(default)">
      <!-- Adding Topology Element TopoElt_1_0 direct on tree -->
      <Item id="TopoElt_1_0">
        <Property id="Description">Topology Element TopoElt_1_0</Property>
        <Property id="Branch">TE_1_0</Property>
      </Item>
      <!-- Adding Topology Element TopoElt_2_0 direct on tree -->
      <Item id="TopoElt_2_0">
        <Property id="Description">Topology Element TopoElt_2_0</Property>
      </Item>
      <!-- Adding Topology Element TopoElt_3_0 direct on tree -->
      <Item id="TopoElt_3_0">
        <Property id="Description">Topology Element TopoElt_3_0</Property>
      </Item>
    </Collection>
    <!-- Adding 2 Topology Element under Topology Element TopoElt_1_0 -->
    <Collection type="TopologyElements" id="(default).TopoElt_1_0">
      <!-- Adding Topology Element TopoElt_1_1 under Topology Element TopoElt_1_0 -->
      <Item id="TopoElt_1_1">
        <Property id="Description">Topology Element TopoElt_1_1</Property>
        <Property id="Branch">TE_1_1</Property>
      </Item>
      <!-- Adding Topology Element TopoElt_1_2 under Topology Element TopoElt_1_0 -->
      <Item id="TopoElt_1_2">
        <Property id="Description">Topology Element TopoElt_1_2</Property>
      </Item>
    </Collection>
  </Group>
</Import>
```

2.2.2 Result



3 Creating and editing template instances

3.1 Interfaces

3.1.1 Instances (Collection)

Syntax	<Collection type="Instances " </Collection>		
Attributes	type	Instances	*Collection of Instances*
	id	#InstId1.#InstId2 ….#InstIdN-1	*Reference to a full parent instance path (tree, template instance or topology level).* *'(default)' id refers to the root (tree)*
Parents	Group		
Children	Item	Instance	
Prerequisites	None		

3.1.2 Instance (Item)

Syntax	<Item id="# InstId "> </Item>		
Attributes	type		*Omitted*
	Id	# InstId	*Name of template instance*
Parents	Group		
	Collection	Instances	
Children	Property		
	Command		
Prerequisites	None		

Properties

Branch	*The branch that will be used during application generation for all configuration elements that the instance contains.*
<Property id="Branch"> #Branch </Property>	
#Branch	String of characters : 0-255 characters
Description	*A description of the instance. Only used within the Application Architect and has no effect on the application itself.*
<Property id="Description"> #Description </Property>	
#Description	String of characters : 0-255 characters
OriginTemplateName	*Full name (with library) of the origin template*
<Property id="OriginTemplateName"> #OriginTemplateName </Property>	

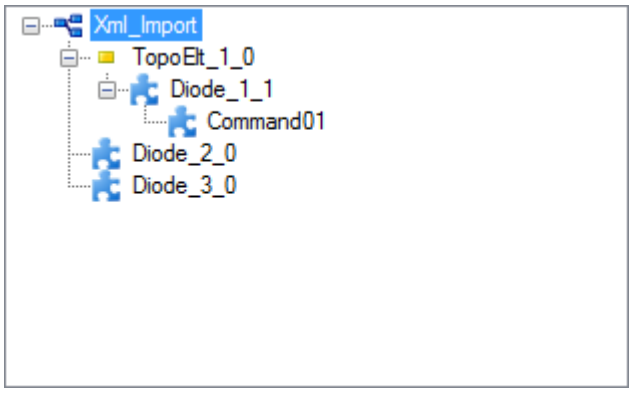
#OriginTemplateName	String of characters : 0-255 characters

3.2 XML sample file an result

3.2.1 XML File

```
<Import>
  <Group customRef="Topology">
    <!-- Adding 1 Topology level and 2 Template instances on tree (id of tree is 'Default')
-->
    <Collection type="TopologyElements" id="(default)">
      <!-- Adding Topology Element TopoElt_1_0 direct on tree -->
      <Item id="TopoElt_1_0">
        <Property id="Description">Topology Element TopoElt_1_0</Property>
        <Property id="Branch">TE_1_0</Property>
      </Item>
    </Collection>
    <Collection type="Instances" id="(default)">
      <!-- Adding Template instance Diode_2_0 direct on tree -->
      <Item id="Diode_2_0">
        <Property id="Description"> Instance of Diode Diode_2_0</Property>
        <Property id="Branch">DIODE_2_0</Property>
        <Property id="OriginTemplateName">/SH_ELEC/Electric_Diode</Property>
      </Item>
      <!-- Adding Template instance Diode_3_0 direct on tree -->
      <Item id="Diode_3_0">
        <Property id="Description"> Instance of Diode Diode_3_0</Property>
        <Property id="Branch">DIODE_3_0</Property>
        <Property id="OriginTemplateName">/SH_ELEC/Electric_Diode</Property>
      </Item>
    </Collection>
    <!-- Adding 1 Template instance under Topology Element TopoElt_1_0 -->
    <Collection type="Instances" id="(default).TopoElt_1_0">
      <!-- Adding Template instance Diode_1_1 under Topology Element TopoElt_1_0 -->
      <Item id="Diode_1_1">
        <Property id="Description"> Instance of Diode Diode_1_1</Property>
        <Property id="Branch">DIODE_1_1</Property>
        <Property id="OriginTemplateName">/SH_ELEC/Electric_Diode</Property>
      </Item>
    </Collection>
    <!-- Adding 1 Template instance under Template instance Diode_1_1 -->
    <Collection type="Instances" id="(default).TopoElt_1_0.Diode_1_1">
      <!-- Adding Template instance Command01 under Template instance Diode_1_1 -->
      <Item id="Command01">
        <Property id="Description">Instance of Command Switch Command01</Property>
        <Property id="Branch">CMD01</Property>
        <Property id="OriginTemplateName">/SH_DISPLAYS/Command_Switch_LIGHT</Property>
      </Item>
    </Collection>
  </Group>
</Import>
```

3.2.2 Result



4 Setting input values and input parameter values

4.1 Interfaces

4.1.1 InstanceInputValues (Collection)

Syntax	<Collection type="InstanceInputValues " </Collection>		
Attributes	type	InstancerInputValues	*Collection of Input values for element's properties*
	id	#InstId1.#InstId2 ….#InstIdN	*Reference to a full instance path (tree, template instance, topology level or element). '(default)' id refers to the root (tree)*
Parents	Group		
Children	Item	InstanceInputValue	
Prerequisites	None		

4.1.2 InstanceInputValue (Item)

Syntax	<Item id="#InstId"> </Item>		
Attributes	type		*Omitted*
	Id	# InstId	*Name of the element's property*
Parents	Group		
	Collection	InstanceInputValues	
Children	Property		
	Command		
Prerequisites	None		

Properties

Value	*The value of the parameter*
<Property id="Value">#Value</Property>	
#Value	

4.1.3 InstanceParameterInputValues (Collection)

Syntax	<Collection type="InstanceParameterInputValues " </Collection>		
Attributes	type	InstanceParameterInput Values	*Collection of Parameter's values*
	id	#InstId1.#InstId2 ….#InstIdN	*Reference to a full instance path (tree, template instance or topology level). '(default)' id refers to the root (tree)*
Parents	Group		
Children	Item	InstanceParameterInputValue	
Prerequisites	None		

4.1.4 InstanceParameterInputValue (Item)

Syntax	<Item id="#InstId"> </Item>		
Attributes	type		*Omitted*
	Id	# InstId	*Full name (with library) of the parameter*
Parents	Group		
	Collection	InstanceParameterInputValues	
Children	Property		
	Command		
Prerequisites	None		

Properties

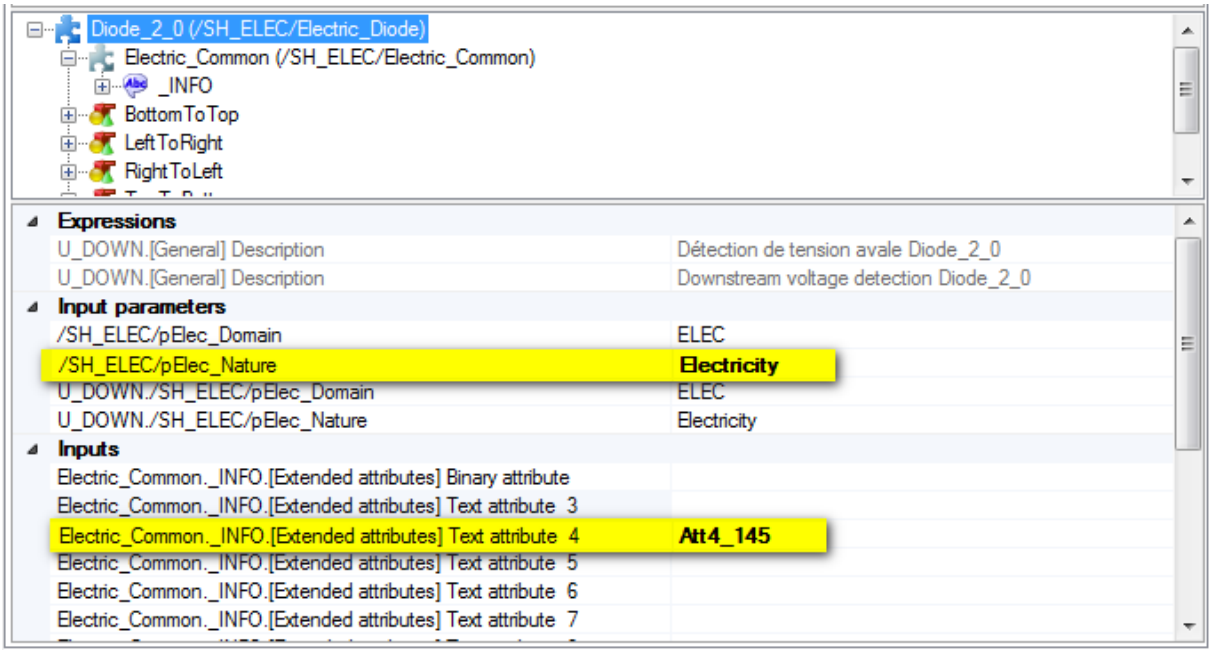
Value	*The value of the parameter*
<Property id="Value">#Value</Property>	
#Value	

4.2 XML sample file an result

4.2.1 XML File

```
<!-- Adding Parameter Value on Template instance Diode_2_0 -->
<Collection type="InstanceParameterInputValues" id="(default).Diode_2_0">
  <Item id="/SH_ELEC/pElec_Nature">
    <Property id="Value">Electricity</Property>
  </Item>
</Collection>
<!-- Adding Input Value on property "ExtText4" of object Electric_Common._INFO in Template
instance Diode_2_0 -->
<Collection type="InstanceInputValues" id="(default).Diode_2_0.Electric_Common._INFO">
  <Item id="ExtText4">
    <Property id="Value">Att4_145</Property>
  </Item>
</Collection>
```

4.2.2 Result



5 Selecting or unselecting items in a template instance

5.1 Interfaces

5.1.1 TemplateObjectInstances (Collection)

Syntax	<Collection type="TemplateObjectInstances" </Collection>		
Attributes	type	TemplateObjectInstances	*Collection of elements in an instance (variables, trends, included template…)*
	id	#InstId1.#InstId2 ….#InstIdN-1	*Reference to a full parent instance path (tree, template instance or topology level). '(default)' id refers to the root (tree)*
Parents	Group		
Children	Item	TemplateObjectInstance	
Prerequisites	None		

5.1.2 TemplateObjectInstance (Item)

Syntax	<Item id="#InstId"> </Item>		
Attributes	type		*Omitted*
	Id	# InstId	*Name of the element*
Parents	Group		
	Collection	TemplateObjectInstance	
Children	Property		
	Command		
Prerequisites	None		

Properties

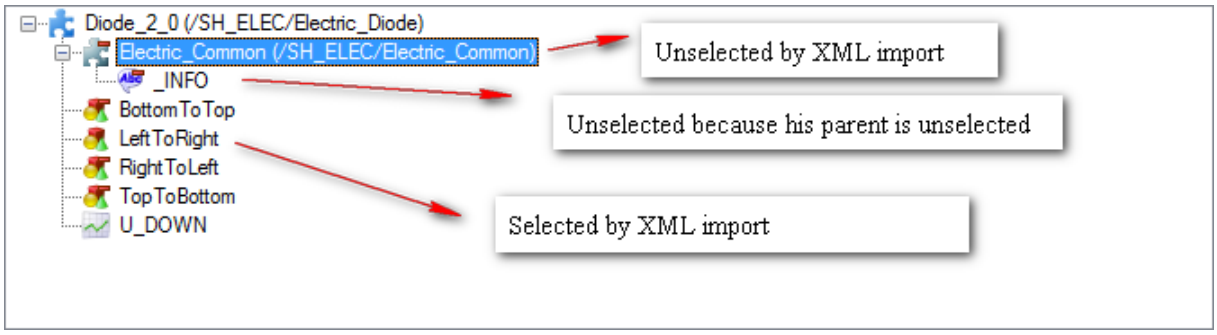
Selected	*Indicates if the element is or not generated. If Selected property is "true" (or not set) the element will be generated. If Selected property is "false" the element will not be generated.*
<Property id="Selected"> #Selected </Property>	
# Selected	Selected = true or -1 Unselected = false or 0

5.2 XML sample file an result

5.2.1 XML File

```
<Collection type="TemplateObjectInstances" id="(default).Diode_2_0">
  <!-- Unselect included template in instance Diode_2_0 -->
  <Item id="Electric_Common">
    <Property id="Selected">false</Property>
  </Item>
  <!-- Select symbol representation in instance Diode_2_0 -->
  <Item id="LeftToRight">
    <Property id="Selected">true</Property>
  </Item>
</Collection>
```

5.2.1 Result



6 Generating Architect application and saving configuration

To save Architect files that have been modified you must use the command "Save" on the item "TemplatesConfiguration".
To generate Architect files you must use the command "FullGeneration" on the item "TemplatesConfiguration".

Using this command will build the application at the Architect level.
This means that the Architect will produce the import file "Tpl_InstancesImport.xml "used to generate real objects for PcVue.

The file Tpl_InstancesImport.xml is located in the C directory of the projet.
To generate real objects in Pcvue you must import the file Tpl_InstancesImport.xml using the XmlImporter tool. This tool is located in BIN directory of PcVue and should not be moved.

6.1 XML File

```
-<Import>
  <!–Generate Tpl_InstancesImport.xml in C directory -->
  -<Group customRef="TemplateInstances">
    -<Item type="TemplatesConfiguration">
      <Command id="Save"/>
      <Command id="FullGeneration"/>
    </Item>
  </Group>
</Import>
```

6.2 Using XmlImporter

The syntax is:

XmlImporter file1 [file2 [file3 [etc..]]

Where each Filex names a file to import or contains a list of files to import.
In the latter case, the list of files is an XML formatted file as follows:

```
<Imports>
<Import>Import1.xml</Import>
<Import>Import2.xml</Import>
…
</Imports>
```