MS Visio to QPR Importer

Version 1.4.0 (29.2.2016)

Contents

1 Overview 1

2 Compatibility 2

3 Installation and usage 2

3.1 Installation 2
3.2 Uninstallation 2
3.3 Using the tool 2

4 Configuration file format 3

5 Expression language 5

6 Example XML configuration 6

QPR Software Plc
Huopalahdentie 24
FI-00350 Helsinki, Finland

Tel. +358 290 001 150
Fax +358 290 001 151
www.qpr.com

Customer Care +358 290 001 155
customercare@qpr.com

Domicile Helsinki
Business ID 0832693-7
VAT number FI08326937

1 Overview

MS Visio to QPR Importer is a tool to convert Microsoft Visio files to QPR ProcessDesigner or EnterpriseArchitect models. Visio files and QPR models are conceptually very different stores of information, and thus several details need to be considered to get the import functioning as desired. The logic of the import is about mapping Visio **shapes** to QPR model **elements**, and the mappings are defined in an xml configuration file.

Visio to QPR Importer is compatible with Visio **vsdx** file format. Other Visio file formats need to be converted to vsdx files before importing. All versions of Microsoft Visio 2013 support the vsdx format, and the support is also available in Microsoft Visio 2010 in SP2 update (more information https://support.microsoft.com/en-us/kb/2844302).

Visio to QPR Importer is not able to make any changes to QPR metamodel, such as element types, attribute types, or element type graphical properties. These settings need to be configured in place using QPR Modelling Client before importing. Of course some settings may be changed afterwards, such as graphical settings.

If multiple Visio files are imported in the same run, the files are processed in an alphabetical order. If the order of processing is relevant, files may be renamed to produce the desired order. A single Visio file may contain multiple **Visio pages**, which are imported from left to right. If the order of page processing is relevant, the order of pages need so be changed by opening the file in Visio. When a single Visio page (i.e. one diagram) is processed, QPR elements created in the following order:

- organization items (swimlanes) from top to bottom
- other element types except connectors
- connectors

Organization items are imported first, because elements are connected to them when elements are positioned on top of organization items. Connectors are imported last, so that they can be connected to elements, which need to exist in the model before connectors are imported.

Processing order may be relevant e.g. in a case where one Visio file contains a shape representing a sub process, and another Visio page describes the contents of that sub process in detail.

The xml configuration file contains a list of **mapping**, which are processed for every Visio shape starting from the first mapping. When a matching mapping is found, information of that mapping is used to create a QPR element. The information includes e.g. which type of element is created and which attributes are imported. If no matching mappings are found, that shape is not imported to QPR.

Visio to QPR Importer can be set to create **instances** of existing elements instead of creating new elements. Existing elements are identified by an **identifier** value, which may simply be a name of the element or a combination of multiple fields. If no existing element is found (i.e. no other element with the same identifier value), a new element is created. Any elements that exists in the QPR model before the import are omitted, i.e. only elements that are created in the same import run are taken into account. Note that connectors cannot be instantiated.

By default, Visio page is imported as a new diagram in the main diagram level of the QPR model. If the import has already created a diagram which identifier value matches the name of the Visio page, contents of the Visio page is imported to in the diagram. To enable this behavior, make sure the identifier matches the page name.

When importing connectors, note that the **Arrow style** in the connector's **Graphical properties** matches the connector type in Visio. If the Visio file contains **Curve** like connectors, and matching QPR connector type is **Elbow**, an error may be caused, because consecutive point in the connector path must be in the same horizontal or vertical plane in Elbow connectors (there are no restrictions in Curve connectors).

2 Compatibility

Compatible with QPR 2015.1 and QPR 2014.1.

3 Installation and usage

Before using Visio to QPR Importer, the tool needs to be **installed**. If updating a new version of the tool, the old version needs to be **uninstalled** before that.

3.1 Installation

1. Unzip the installation package **VisioToQPRImporter.zip** to freely selectable location.
2. Double click **setup.exe**.
3. Click **Install** button. The tool is installed and started.
4. Make sure QPR ProcessDesigner or EnterpriseArchitect Client is installed in the workstation and the COM object is registered to the client (more information https://community.qpr.com/node/1670) – especially if you encounter any error messages when running the tool.

3.2 Uninstallation

1. Open **Programs and Features** in Windows Control Panel.
2. Search **Visio to QPR Importer**, click it, and then click **Uninstall/Change**.
3. Click **OK** button.

3.3 Using the tool

1. First define an xml configuration file and a QPR model where data is imported.
2. Open **Visio to QPR Importer** from Windows start menu (or by double clicking the same **setup.exe** used in installing the tool).
3. Click **1. Select xml configuration** and select the xml configuration file.
4. Click **2. Select Visio files** and select Visio files (one or many).
5. Click **3. Start import** to start the import.
6. A message a shown when the import is ready (or an error has occurred).

4 Configuration file format

The XML configuration uses namespace http://www.qpr.com/QPREnterpriseArchitect/VisioImport. It has the following hierarchical XML tag structure:

visioimport (1)
  mapping (1..n)
    attribute (1..n)

The following tables list all available attributes of the xml tags. Tags having a default value are optional (i.e. they may be left out).

Tag: **visioimport**	
Tag attribute	**Description**
logfile (string) (default: [empty])	Folder path and file name of the log file. If empty, logging is disabled. Logged data is appended to the end of an existing file.
servername (string)	QPR server name
serverport (integer) (default: 4251)	QPR server port
username (string) (default: [empty])	QPR username
password (string) (default: [empty])	QPR password
domain (string) (default: [empty])	QPR user domain
showclient (boolean) (default: true)	**true** or **false**. When true, the QPR modelling client is visible during the import.
modelname (string)	Name of the QPR EA/PD model, where the data is imported to. The model name always starts with \\. Folder names can be defined with \. Format is e.g. - \\modelName - \\folder1\folder2\modelName
pixelsperinch (decimal number)	Shapes in Visio file are defined as length units (such as inches) and elements in QPR model are defined as pixels. pixelsperinch define in what size elements are drawn in QPR model. The larger the number, the larger the elements. 100 might be approximately a good value.
diagramelementtype (string)	Visio pages are imported as separate diagrams in the QPR model in the main diagram level. The diagramelementtype defines the used diagram/subprocess element type.
closeclient (boolean) (default: true)	**true** or **false**. When false, the modelling client is left open after the import and the model is not saved. The user has then a possibility to check the import result and close the model with or without saving.

loglevel (string) (default: information)	Determines the amount of information that is logged during the import. One of the following: - **error**: Only errors are logged. - **information**: In addition to error level, main phases of the import and all imported elements are logged. - **verbose**: In addition to information level, very detailed logging of the import process is performed.

Tag: **mapping**	
Tag attribute	**Description**
matchingshape (expression)	Expression which result determines whether this mapping is used. The expression must be evaluated to a Boolean value. For more information of the expression language, see 5. Expression language.
elementtype (string)	QPR element type name.
useexisting (default false)	**true** or **false**. Determines whether existing QPR elements may be reused, i.e. they are instantiated from existing elements instead of creating a new element. If no existing element is found, a new element is created in any case. An existing element is searched using the defined identifier (see the next attribute). useexisting can also be defined as an expression. For more information of the expression language, see 5. Expression language.
identifier (expression) (default: element name)	Expression which result is the identifying information ("key") of an element when searching for a corresponding existing element. If identifier is not defined, the defined element name attribute is used as an identifier (if name attribute is defined in the configuration). For more information of the expression language, see 5. Expression language.
type (default: other)	Determines the main behavior category of the element. One of the following: - **diagram**: Shape is imported as an element having diagram/subprocess behavior. - **role**: Shape is imported as an Organization item element type (i.e. swimlane). - **connector**: shape is imported as a connector type of element. - **other**: all other types of elements than the previous ones, e.g. activities. Note that the defined QPR element type must match this setting.

Tag: **attribute**	
Tag attribute	**Description**
name	QPR element's attribute name. Available attribute names are mentioned in the product documentation: http://kb.qpr.com/qpr2015-1/index.html?setproperty3.htm

value	Value of the attribute defined as an expression. For more information of the expression language, see chapter 5. Expression language.

5 Expression language

MS Visio to QPR Importer has an embedded expression language, which can be used to define values in the configuration file dynamically. For more information about the expression language, see the documentation in https://community.qpr.com/node/1713.

Following additional function for reading Visio file data are available for the expressions. The functions have as a first parameter the shape object that is being processed. All parameters of these functions exept shapes are strings.

Function	Parameters	Description
Attribute (string)	- [shape] - attribute name - default value (optional)	Shape element attribute. If Shape element doesn't have the attribute, then the attribute is searched from corresponding master shape, and then from corresponding master element (in the masters part of the vsdx file). Shape properties can be seen in Visio by clicking a shape and then **Shape Name** in the **Developer** tab.
MasterShapeAttribute (string)	- [shape] - attribute name - default value (optional)	Attribute of the corresponding master shape. Master shape attributes can be seen in Visio in the **Drawing Explorer** (click **Drawing Explorer** tickbox in the **Developer** tab).
MasterAttribute (string)	- [shape] - attribute name - default value (optional)	Attribute of the corresponding master element (in the masters part).
Property (string)	- [shape] - section name - row name - property name - default value (optional)	Property of the shape in the defined section and row. If no property is found, the defined default value is returned.
ContainsProperty (boolean)	- [shape] - section name - row name - property name	Returns **true** or **false** depending whether the shape contains the property in the defined section and row.
ShapeData (string)	- [shape] - property name - default value	Returns value of a shape data property. Available shape data properties can be seen in Visio when selecting a shape and then in the popup menu **Data** > **Define Shape Data...**.

		Note that the property name is the "Name" of the property, not "Label".
Text (string)	- [shape]	Text value of the shape (in vsdx format the Text tag of the Shape).
ParentShape (shape)	- [shape]	Returns shape's parent shape. Can be used if the identifying property is in the parent shape.
ChildShapes (array of shapes)	- [shape]	Returns shape's child shapes.
XPath (shape)	- [shape] - xpath expression	Returns a result of an xpath expression targeted to the shape. Returns only a single xml tag.

Example expressions for matching expression:

- ShapeData([shape], 'BpmnElementType') = 'Event'
- text([shape]) = "Activity 1"
- ShapeData(ParentShape([shape]), 'BpmnElementType') = 'Event'
- IndexOf(MasterAttribute([shape], 'Name'), 'Dynamic connector') = 0
- ContainsProperty([shape], 'Property', 'Function', 'Value')
- true (This is always a match. May be useful as a last mapping.)

6 Example XML configuration

```
<?xml version="1.0" encoding="utf-8"?>
<visioimport xmlns="http://www.qpr.com/QPREnterpriseArchitect/VisioImport" logfile="C:\Projektit\Visio Import\VisioImportLog.txt"
servername="localhost" serverport="4251" username="qpr" password="demo" domain="" showclient="1" modelname="\\Iycon"
pixelsperinch="100" diagramelementtype="Sub-Process" closeclient="true" loglevel="verbose">
  <mapping matchingshape="ShapeData([shape], 'BpmnElementType') = 'Event' and ShapeData([shape], 'BpmnEventType') =
'Start'" elementtype="Start Event">
    <attribute name="name" value="ShapeData([shape], 'BpmnName')" />
    <attribute name="trigger" value="ShapeData([shape], 'BpmnTriggerOrResult')"/>
  </mapping>
  <mapping matchingshape="ShapeData([shape], 'BpmnElementType') = 'Event' and ShapeData([shape], 'BpmnEventType') = 'End'"
elementtype="End Event">
    <attribute name="name" value="ShapeData([shape], 'BpmnName')" />
    <attribute name="result" value="ShapeData([shape], 'BpmnTriggerOrResult')"/>
  </mapping>
  <mapping matchingshape="ShapeData([shape], 'BpmnElementType') = 'Event' and (ShapeData([shape], 'BpmnEventType') =
'Intermediate' or ShapeData([shape], 'BpmnEventType') = 'Intermediate (Non-Interrupting)' or ShapeData([shape],
'BpmnEventType') = 'Intermediate (Throwing)')" elementtype="Intermediate Event">
    <attribute name="name" value="ShapeData([shape], 'BpmnName')" />
    <attribute name="trigger" value="ShapeData([shape], 'BpmnTriggerOrResult')"/>
  </mapping>
  <mapping matchingshape="ShapeData([shape], 'BpmnElementType') = 'Sub-Process' and ShapeData([shape], 'BpmnActivityType')
= 'Sub-Process'" elementtype="Sub-Process" useexisting="true">
    <attribute name="name" value="ShapeData([shape], 'BpmnName')" />
  </mapping>
  <mapping matchingshape="ShapeData([shape], 'BpmnElementType') = 'Sub-Process' and ShapeData([shape], 'BpmnActivityType')
= 'Task'" elementtype="Task">
    <attribute name="name" value="ShapeData([shape], 'BpmnName')" />
  </mapping>
  <mapping matchingshape="ShapeData([shape], 'BpmnElementType') = 'Gateway'" elementtype="Gateway">
    <attribute name="name" value="ShapeData([shape], 'BpmnName')" />
    <attribute name="GatewayType" value="ShapeData([shape], 'BpmnGatewayType')"/> <!-- custom attribute in QPR -->
```

```
  </mapping>
  <mapping matchingshape="ShapeData([shape], 'BpmnElementType') = 'Connecting Object' and ShapeData([shape],
'BpmnConnectingObjectType') = 'Sequence Flow'" elementtype="SequenceFlow" type="connector">
    <attribute name="name" value="ShapeData([shape], 'BpmnName')" />
  </mapping>
  <mapping matchingshape="ShapeData([shape], 'BpmnElementType') = 'Connecting Object' and ShapeData([shape],
'BpmnConnectingObjectType') = 'Message Flow'" elementtype="MessageFlow" type="connector">
    <attribute name="name" value="ShapeData([shape], 'BpmnName')" />
  </mapping>
  <mapping matchingshape="ShapeData([shape], 'BpmnElementType') = 'Connecting Object' and ShapeData([shape],
'BpmnConnectingObjectType') = 'Association'" elementtype="Association" type="connector">
    <attribute name="name" value="ShapeData([shape], 'BpmnName')" />
  </mapping>
  <mapping matchingshape="ShapeData([shape], 'BpmnElementType') = 'Lane'" elementtype="Organization item type" type="role">
    <attribute name="name" value="ShapeData([shape], 'BpmnName')" />
  </mapping>
  <mapping matchingshape="IndexOf(MasterAttribute([shape], 'Name'), 'Swimlane.') = 0 or MasterAttribute([shape], 'Name') =
'Swimlane'" elementtype="Organization item type" type="role"> <!-- this mapping is for non-BPMN shape -->
    <attribute name="name" value="property([shape], 'User', 'visHeadingText', 'Value')" />
  </mapping>
  <mapping matchingshape="ShapeData([shape], 'BpmnElementType') = 'Artifact' and ShapeData([shape], 'BpmnArtifactType') =
'Data Object'" elementtype="DataObject">
    <attribute name="name" value="ShapeData([shape], 'BpmnName')" />
  </mapping>
  <mapping matchingshape="ShapeData([shape], 'BpmnElementType') = 'Artifact' and ShapeData([shape], 'BpmnArtifactType') =
'Group'" elementtype="Group type">
    <attribute name="name" value="ShapeData([shape], 'BpmnName')" />
  </mapping>
  <mapping matchingshape="ShapeData([shape], 'BpmnElementType') = 'Artifact' and ShapeData([shape], 'BpmnArtifactType') =
'Annotation'" elementtype="Note type">
    <attribute name="name" value="ShapeData([shape], 'BpmnName')" />
  </mapping>
  <mapping matchingshape="ShapeData([shape], 'BpmnElementType') = 'Artifact' and ContainsProperty([shape], 'Property',
'BpmnInitiating', 'Value')" elementtype="Message">
    <attribute name="name" value="ShapeData([shape], 'BpmnName')" />
  </mapping>
  <mapping matchingshape="ShapeData([shape], 'BpmnElementType') = 'Artifact'" elementtype="DataStore">  <!-- this mapping is
for shapes with BpmnElementType=Artifact where BpmnArtifactType is not any of the ones above -->
    <attribute name="name" value="ShapeData([shape], 'BpmnName')" />
  </mapping>
  <mapping matchingshape="ContainsProperty([shape], 'Property', 'Function', 'Value') and ToLower(Property([shape], '', '',
'FillForegnd')) = '#ff0000'" elementtype="Red element">
    <attribute name="name" value="text([shape])" />
  </mapping>
  <mapping matchingshape="ContainsProperty([shape], 'Property', 'Function', 'Value') and ToLower(Property([shape], '', '',
'FillForegnd')) = '#ffc000'" elementtype="Yellow element">
    <attribute name="name" value="text([shape])" />
  </mapping>
  <mapping matchingshape="ContainsProperty([shape], 'Property', 'Function', 'Value') and ToLower(Property([shape], '', '',
'FillForegnd')) = '#bfbfbf'" elementtype="Grey element">
    <attribute name="name" value="text([shape])" />
  </mapping>
  <mapping matchingshape="ContainsProperty([shape], 'Property', 'Function', 'Value')" elementtype="Task">
    <attribute name="name" value="text([shape])" />
  </mapping>
  <mapping matchingshape="IndexOf(MasterAttribute([shape], 'Name'), 'Dynamic connector') = 0" elementtype="SequenceFlow"
type="connector">
    <attribute name="name" value="Text([shape])" />
  </mapping>
  <mapping matchingshape="IndexOf(MasterAttribute(ParentShape([shape]), 'Name'), 'CFF Container') = 0 and Text([shape]) != ""
elementtype="Task">
    <attribute name="name" value="Text([shape])" />
  </mapping>
```

```
  <mapping matchingshape="IndexOf(MasterAttribute(ParentShape([shape]), 'Name'), 'Separator') = 0 and Text([shape]) != ""
elementtype="Checkpoint type">
    <attribute name="name" value="Text([shape])" />
  </mapping>
</visioimport>
```