Advanced Document Templating

The plugin provides a Smart Service to create docx documents defining the dynamic structure within a docx template. The plugin is based on the library "XDocReport" https://code.google.com/p/xdocreport/ .

How to Configure the 'DOCX From Dynamic Template' Smart service

Data Input Tab

Input	Data Type	Required	Multiple	Description
Docx Template	Document	Yes	No	Docx Document that is defininig the structure of the document to be generated.
Xml Data Model	Any Type	Yes	N/A	Xml String containing all the values that will be placed into the docx documen. Please follow 'How to use it' section to define it correctly.
Images	Any Type	No	N/A	Dynamic CDT containing images that will be placed into the generated docx document. Please follow 'How to use it' section to define it correctly.
Create New Document	Boolean	Yes	No	Whether to create a new document, or update an existing one.
New Document Name	Text	No	No	If creating a new document, use this as the name.
New Document Desc	Text	No	No	If creating a new document, use this as the description.
Save In Folder	Folder	No	No	If creating a new document, save into this folder.
Existing Document	Document	No	No	If not creating a new document, overwrite this one.

Data Output Tab

Output	Data Type	Multiple	Description
Success	Boolean	No	Returns false if an error has occurred. Save this value to a process variable to enable exception processing on the subsequent activity in the process flow.
Error Message	Text	No	Lists the text of the error message if one occurred.
New Document Created	Document	No	A created DOCX Document

How To use the 'DOCX From Dynamic Template' Smart service

Defining a docx Template

How to design a report: https://code.google.com/p/xdocreport/wiki/DocxDesignReport#Create_with_MS_Word .

Defining 'XML Data Model' data input

The expected Xml String passed into DynamicCdtDataModel Data Input must follow these principles:

1. The XML String has only one main element (in the example is "project") which is the root of all the mapped values
2. Within the root element there can be simple elements or multiple elements:
    - Simple elements (in the example is "name") are simple placeholders in the DOCX template. Their name is the same as the placeholder name in the DOCX template
    - Multiple elements (in the example is "developer") are placeholders that are mapped in the DOCX template with dynamic structures (such as grids). Their name is the same as the placeholder name in the DOCX template and its field attributes have the same name as the sub placeholders in the dynamic structure (for instances, if the multiple elements is mapping a grid, the fields of the multiple elements will have the same name of the column placeholders of the grid). Multiple elements must be of type type!CDT from the data management tab. For instance, the value of a multiple element can be the output of a query rule/query entity.

To test the plugin with the docx template provided as example you can use the following example of 'Xml Data Model' definition:

```
="<project>
    <name>This is a simple placeholder</name>
    <developer name='John' lastEmail='Smith' mail='smith@email' day='10-10-2014' />
    <developer name='Frank' lastEmail='Brown' mail='frank@email' day='10-10-2014' />
</project>"
```

Defining 'Images' data input

This custom Smart Services allows you to place images (and dynamic arrays of them) into the docx document. This input parameters expects a dynamic CDT as input where all the attributes are named as the name of the placeholders and contain an Appian uploaded image document (or an array of them) as value. To test the plugin with the docx template provided as example you can set the input of this parameter as follow:

```
={logo:cons!ADT_IMAGE}
```

The costant "ADT_IMAGE" should be of type document (multiple) and contain more than one image.

Troubleshooting

Here below a list of possible errors during the execution of the plugin which can be challenging to troubleshoot:

"The entity name must immediately follow the '&' in the entity reference."

The data input "Xml Data Model" of the Smart Service is expected to be an XML string of all the data that will be placed into the generated DOCX document. Since it's an XML string, it's not expecting to find the char "&" (or other XML protected chars like “<” or “>”) within the value of an xml tag. In this case the solution should be using the fn!toHtml every time a value from a pv! (or ri!) is placed in the text string that is the XML. This will substitute the special chars with the HTML code of them (avoiding them to break the XML syntax otherwise) but will still show correctly in the generated DOCX. For instance, if the content of a pv! (or ri!) is “&<>” this will happen:

```
fn!toHtml("&<>")      →      &amp;&lt;&gt;
```