**Translating Long Form Dates into other languages**

Suppose we are designing a document generation solution where we will have a single NDA Document Generation Form, but need to output the contract document in multiple languages.

There are a few different ways to approach the overall design. The most important consideration is how the contract language would be determined. This could be set as a field in an upstream system, such as Salesforce, or a human decision made by the submitter/contract drafter. Keep in mind that CLM does not read into the contents of the template, but accesses content via the attributes of the document.

In the scenario, we make the assumption that the submitter must make a human decision on which language in which to generate the contract, so we now need to plan out the Document Generation design. The customer has requirements for the contract to be generated in six different languages, and they currently manages these in six separate word document templates

| **Configuration Component** | **Quantity Needed** | **Rationale** |
| --- | --- | --- |
| Word templates | 6 | NDA template in 6 different languages |
| Doc Gen Form | 1 | Required inputs for each template are identical. Assumption that users will be comfortable working in a single language even if generating contracts in other languages. |
| Doc Gen Configuration | 1 | The single Doc Gen Configuration will present all 6 template from a single button where the user selects which language to generate the contract |

The Doc Gen Form has been created, and includes fields with a date. The contracts require a date to be input in the format of dd MMMM yyyy (e.g. “23 February 2021”). However, as of this writing, the DocuSign CLM Doc Gen Form will not output a written date in a language other than English. The desired output would be a date in a non-English language. For example, the Spanish date should be “23 Febrero 2021.”

**Approach**

1. Parse the date from a single XML node into separate merge tags for the day, month and year.
2. Configure the conditional merge tag to translate for a month.

**Input XML**

Here is the sample Document Generation XML used in this exercise.

<?xml version="1.0" encoding="utf-16" standalone="yes"?>

<TemplateFieldData displayName="" displayValue="">

<Counterparty\_Name>Burlington Textiles Corp of America</Counterparty\_Name>

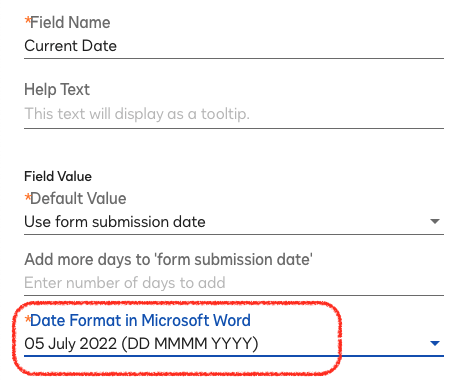
<Dates displayName="Dates" displayValue="Dates">

<SLA\_Expiration>30 March 2020</SLA\_Expiration> <SLA\_Expiration\_unformatted>2020-03-30T00:00:00.0000000</SLA\_Expiration\_unformatted>

</Dates>

</TemplateFieldData>

The SLA\_Expiration node was configured in CLM Document Generation as dd MMMM yyyy. The date format is an option when we configure the date doc gen field.



**How to Parse the Date into Unique Fields**

Instead of the standard single merge tag for the date, we will parse the date field into three separate components (day, month and year) using xpath.

Standard merge tag to insert the date:

* <# <Content Select="//SLA\_Expiration" Optional="true" /> #>

The contents between the quotation marks is an xpath expression which is evaluating the input XML (from our document generation form). Xpath expressions can be tested out using an xPath tester tool such as [Code Beautify](https://codebeautify.org/Xpath-Tester).

To parse the string from the SLA\_Expiration node into smaller segments, use the Substring xpath function.

Substring xpath function: substring(string, offset, length)

To configure Day & Year:

| Unit | xPath Expression | Merge Tag | Expected Output |
| --- | --- | --- | --- |
| Day | substring(//SLA\_Expiration/text(),0,3) | <# <Content Select="substring(//SLA\_Expiration/text(),0,3)" Optional="true" /> #> | 30 |
| Year | substring(//SLA\_Expiration/text(), string-length(//SLA\_Expiration) - 4 + 1, 4) | <# <Content Select="substring(//SLA\_Expiration/text(),0,3)" Optional="true" /> #> | 2020 |

In the year example, we are still using the Substring xpath function.

* Text: //SLA\_Expiration/text()
* Offset: string-length(//SLA\_Expiration) - 4 + 1
* Length: 4

**Conditional to translate March to Marzo**

For the month, we could use the same expression to parse out and get the month:

* substring(//SLA\_Expiration/text(),4,(string-length(//SLA\_Expiration)-8))

However, we do not simply want to output “March”, we want to have the translated version (“Marzo”) displayed if “March” was in the date field. This will require a conditional merge tag.

The Conditional Merge Tag:

* <# <Conditional Test=“XPATH” /> #>
* We want to know if “March” is contained within the SLA\_Expiration node.

The proper xpath statement we should use is one which will resolve to true or match.

* //SLA\_Expiration[contains(text(),'March')]

The conditional merge tag to handle March would be:

* <# <Conditional Test=“//SLA\_Expiration[contains(text(),'March')]” /> #>Marzo<# <EndConditional> />

The conditional merge tags to translate all months to Spanish would be:

* <# <Conditional Test=“//SLA\_Expiration[contains(text(),'January')]” /> #>Enero<# <EndConditional> /><# <Conditional Test=“//SLA\_Expiration[contains(text(),'February')]” /> #>Febrero<# <EndConditional> /><# <Conditional Test=“//SLA\_Expiration[contains(text(),'March')]” /> #>Marzo<# <EndConditional> /><# <Conditional Test=“//SLA\_Expiration[contains(text(),'April')]” /> #>Abril<# <EndConditional> /><# <Conditional Test=“//SLA\_Expiration[contains(text(),'May')]” /> #>May<# <EndConditional> /><# <Conditional Test=“//SLA\_Expiration[contains(text(),'June')]” /> #>Junio<# <EndConditional> /><# <Conditional Test=“//SLA\_Expiration[contains(text(),'July')]” /> #>Julio<# <EndConditional> /><# <Conditional Test=“//SLA\_Expiration[contains(text(),'August')]” /> #>Augusto<# <EndConditional> /><# <Conditional Test=“//SLA\_Expiration[contains(text(),'September')]” /> #>Septiembre<# <EndConditional> /><# <Conditional Test=“//SLA\_Expiration[contains(text(),'October')]” /> #>Octubre<# <EndConditional> /><# <Conditional Test=“//SLA\_Expiration[contains(text(),'November')]” /> #>Noviembre<# <EndConditional> /><# <Conditional Test=“//SLA\_Expiration[contains(text(),'December')]” /> #>Diciembre<# <EndConditional> />

To put all the components together into a date statement, combine the day, month and year merge tag statements:

* <# <Content Select="substring(//SLA\_Expiration/text(),0,3)" Optional="true" /> <# <Conditional Test=“//SLA\_Expiration[contains(text(),'January')]” /> #>Enero<# <EndConditional> /><# <Conditional Test=“//SLA\_Expiration[contains(text(),'February')]” /> #>Febrero<# <EndConditional> /><# <Conditional Test=“//SLA\_Expiration[contains(text(),'March')]” /> #>Marzo<# <EndConditional> /><# <Conditional Test=“//SLA\_Expiration[contains(text(),'April')]” /> #>Abril<# <EndConditional> /><# <Conditional Test=“//SLA\_Expiration[contains(text(),'May')]” /> #>May<# <EndConditional> /><# <Conditional Test=“//SLA\_Expiration[contains(text(),'June')]” /> #>Junio<# <EndConditional> /><# <Conditional Test=“//SLA\_Expiration[contains(text(),'July')]” /> #>Julio<# <EndConditional> /><# <Conditional Test=“//SLA\_Expiration[contains(text(),'August')]” /> #>Augusto<# <EndConditional> /><# <Conditional Test=“//SLA\_Expiration[contains(text(),'September')]” /> #>Septiembre<# <EndConditional> /><# <Conditional Test=“//SLA\_Expiration[contains(text(),'October')]” /> #>Octubre<# <EndConditional> /><# <Conditional Test=“//SLA\_Expiration[contains(text(),'November')]” /> #>Noviembre<# <EndConditional> /><# <Conditional Test=“//SLA\_Expiration[contains(text(),'December')]” /> #>Diciembre<# <EndConditional> /> <# <Content Select="substring(//SLA\_Expiration/text(),0,3)" Optional="true" /> #>

The above merge tag design will translate a single Doc Gen field that outputs “30 March 2020” into “30 Marzo 2020”.

**Additional resources:**

* [Word Template Merge Tags](https://support.docusign.com/en/guides/SpringCM-Document-Generation-Word-Template-Merge-Tags)
* [XPath Expressions](https://support.docusign.com/en/guides/SpringCM-XPath-Expressions)
* [Partner Webinar: Streamline Doc Gen and Workflow with XPath Queries](https://dsupartners.docusign.com/path/docusign-si-partner-tech-deep-dive-webinar/how-to-streamline-doc-gen-and-workflow-with-xpath-queries)