This guide will take you through the process of generating neat and professional looking documents pre-filled with data from the datasource of your choice using DOCX templates, inside your airSlate application.

Requirements:
An airSlate account and working knowledge of airSlate.
# Table of Contents

**How to set up DOCX templates** 3

**How to generate tags via the Tag Generator** 7

**Fillable fields in DOCX templates** 9
   - Add fillable fields using the Fillable Fields panel 9
   - Add fillable fields using text tags 14
   - Fillable fields inside dynamic DOCX tables 17

**Online Editor: text formatting** 20

**Set up Bots for documents generated from DOCX templates** 21

**Convert Document Generation templates to Contract Negotiation templates** 23

**DOCX template syntax** 25
   - What are tags used for? 25
   - How to add tags? 25
   - How to populate tags? 25
   - Types of tags 26
   - How to generate tags 27
   - How to use tags 28
   - Text formatting functions (date, number, currency) 28
      - How to insert today's date 28
      - Table with a dynamic number of rows 29
      - How to hide an empty table header or footer (using {{h}} tag) 29
      - How to calculate totals in a table column ($sumabove variable) 30
   - Conditional Logic 30
   - List 31
   - Variables 31
      - Math functions 32
      - Fillable fields 34

© 2022 airSlate Inc.  
All rights reserved.
How to set up DOCX templates

Easily create documents that are dynamically pre-filled using DOCX templates. Add field tags to your DOCX templates and set up Flows using any pre-fill bot. Documents will be generated using original DOCX formatting, and will be pre-filled with data from the selected datasource in seconds.

Refer to the short step-by-step instructions below to get started:

1. Open your airSlate application and select **Create New Flow**.

![Create New Flow](image)

2. The **Add documents and forms** page will open.

In the **Create** section, select **Document** from the list.
Upload a DOCX document (30 MB max) or create a new one using the online editor.

Note: Upload a DOCX document with previously added tags or add them later when editing the template.

Select **Edit template** to:

- edit existing tags in an uploaded DOCX template
- add new tags (see the How to generate tags via Tag Generator section)
- add fillable fields (see the Fillable fields in DOCX templates section)
A DOCX template will open in the online editor.

When finished, click **Complete**.

You can also:

- temporarily hide a template (until the visibility conditions set in the Conditional Document Bot are met)
- download the original version of a DOCX template (along with edits made in the online editor)
- create a template copy
- delete a template from the Flow
To do so, click the three dots and select the option you need.

**Note:** Once a revision for a DOCX document has been created, you won't be able to edit or delete the DOCX template (until all created revisions of that DOCX document have been deleted).

Set up Bots and/or roles (Steps) for your document, if needed (see the Assigning Roles and Permissions to Fields and Documents user guide).
How to generate tags via the Tag Generator

You can easily generate tags using the Tag Generator in the online editor:

1. To do so, switch to the **Tags** panel in the upper right corner.

2. In the **Generator** tab, select the tag type:
   - Insert text
   - Insert table
   - List
   - Add conditional logic
   - Insert image

   Enter the field name. Then, select the tag formatting such as **text, date, number, or currency**.

3. Once finished, a tag will be automatically generated according to your settings. Select **Copy tag** above the tag box to copy it. Paste a tag wherever you need in your DOCX template by pressing **Ctrl+V** on your keyboard.
Note: For more detailed information on tags, go to the Documentation tab of the Tags panel.
Fillable fields in DOCX templates

Add fillable fields to your DOCX templates and convert them to PDF so that your clients and colleagues can add their fillable data. Add fillable fields using text tags or move them from the Fillable Fields panel using the online editor.

Add fillable fields using the Fillable Fields panel

Add fillable fields to your DOCX templates by moving them from the Fields panel in the online editor.

1. The Fields panel (with the list of fillable fields) will open automatically when editing a DOCX template.

There will be two sections with fillable fields:

- **Standard fields** (general field types)
- **Template fields** (specific field types)

♫ Note: When users start filling your documents, the data entered in the template fields will be validated according to the specific field type. For example, when filling the US phone number field, users won’t be able to enter their phone number in any other formats.

Select the icon of the fillable field you’d like to add by clicking it. Then move the field onto the DOCX template.

The fillable field will be placed in a DOCX template where your cursor is located.
2. Customize your fillable field's settings:

- Enter some text to help your clients complete the fields correctly.
- Specify if your fillable field is required to be filled in.
- Enter a name for your fillable fields (it will be used by Bots).
- Set the font properties of your fillable fields (font type, size, text alignment etc.).
For more settings, select Advanced.

Fillable fields are placed in a DOCX template with the Inline wrapping style by default. This means the fillable fields will be placed in the DOCX template where your cursor is located.
Selecting a wrapping style other than **Inline** allows you to move the field around by dragging, or resizing it using the dots around the edges.

To change your fillable field's wrapping style, right click it and select **Wrapping style** from the menu.

Next, select a wrapping style from the list.

When finished, click **Complete**.

3. Once a PDF has been generated out of a DOCX template, the fillable fields added to your document will show up as follows and will be ready for filling by your clients and colleagues.
PURCHASE AND SALE CONTRACT

This Purchase and Sale Contract (the "Contract") is made in two original copies, effective between:

BETWEEN:

the "Vendor", a company organized and existing under the laws of , with its head office at ,

AND:

the "Purchaser", an individual existing under the laws of the

Please enter text.
Add fillable fields using text tags

Text tags in Document Creator are specifically formatted combinations of symbols that can be placed anywhere within your documents in order to preset the location, size and type of various fields such as Signature, Text, Initial fields, etc.

You can add text tags before uploading your document or when a document is uploaded (by editing it in the editor).

For example, the text tags being used for the ‘Text’ line are the following:

```
{{t (for Type of field): t (for Text): r:n; l (field name): "TextField1"; }}
```

The text tags first being used on the ‘Sign Here’ line are the following:

```
{{t (for Type of field): s (for Signature); r:n; l:"SignatureField1"}}
```
The table below shows the most widely used text tags in Document Generation along with the types of fields for which these text tags can be applied.

<table>
<thead>
<tr>
<th>KEY</th>
<th>What does it mean?</th>
<th>Accepted format &amp; values</th>
<th>What fields need this KEY?</th>
</tr>
</thead>
<tbody>
<tr>
<td>t</td>
<td>TYPE of the field</td>
<td>s (for signature)</td>
<td>For every field</td>
</tr>
<tr>
<td></td>
<td></td>
<td>i (for initials)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>t (for text)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>d (for dropdown)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>c (for checkboxes)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>dt or date (for dates)</td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>REQUIRED</td>
<td>y (for required)</td>
<td>For every field</td>
</tr>
<tr>
<td></td>
<td></td>
<td>n (for optional)</td>
<td></td>
</tr>
<tr>
<td>l</td>
<td>LABEL</td>
<td>“Field name”</td>
<td>For every field</td>
</tr>
<tr>
<td>dd</td>
<td>DROPDOWN</td>
<td>&quot;option1, option2,</td>
<td>Only for Dropdown fields</td>
</tr>
<tr>
<td></td>
<td></td>
<td>option3, ....&quot;</td>
<td></td>
</tr>
<tr>
<td>w</td>
<td>WIDTH</td>
<td>w (width)</td>
<td>For every field</td>
</tr>
<tr>
<td>h</td>
<td>HEIGHT</td>
<td>h (height)</td>
<td>For every field</td>
</tr>
<tr>
<td>p</td>
<td>DATE FORMAT</td>
<td>MM/DD/YYYY</td>
<td>Only for Date fields</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DD/MM/YYYY</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>HH:mm</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>MMM DD, YYYY</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>MM/DD/YY</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>YYYY-MM-DD</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>YYYY/MM/DD</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>YYYY-M-D</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>YYYY/M/D</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D.M.YYYY</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D-M-YYYY</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D/M/YYYY</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DD.MM.YYYY</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DD-MM-YYYY</td>
<td></td>
</tr>
</tbody>
</table>
Once a document has been generated out of a DOCX template, text tags added to your document will show up automatically being converted into fillable fields.

Such fillable fields can then be mapped as usual, while configuring your bots.
Fillable fields inside dynamic DOCX tables

Add fillable fields to dynamic DOCX tables using fillable field tags. Assign steps (roles) to the tables and allow users to easily add and/or edit the added text in generated PDF documents.

Let's consider an example where we need to pre-fill one column of a DOCX table with the names of employees from another document (web form) using the Pre-fill from Doc to Doc Bot. Users will enter their date of employment and email into the fillable fields of the DOCX table next to their names.

Requirements:

- a document for pre-filling a DOCX table
- add a DOCX template with a dynamic table to the Flow
- add fillable fields to the dynamic table (see the Add fillable fields using text tags section of this user guide)
- configure the Pre-fill from Doc to Doc Bot for the Flow

Note: You can only assign a step (role) to a single column of a dynamic table. It is not possible to assign a step (role) to each fillable field separately.

The DOCX table with fillable field tags will appear as follows in the online editor:
Once the requirements have been met, let’s proceed to creating Slates. The document that DOCX tables will be pre-filled from will appear as follows:

Then, users are expected to proceed to the second document that will be pre-filled with the data entered in the first document. Users will need to fill in the fillable fields.
### Employee Data Table

<table>
<thead>
<tr>
<th>Name</th>
<th>Date</th>
<th>Email</th>
</tr>
</thead>
<tbody>
<tr>
<td>James Smith</td>
<td>14/09/2021</td>
<td><a href="mailto:james.smith@gmail.com">james.smith@gmail.com</a></td>
</tr>
<tr>
<td>Tom Dopp</td>
<td>10/09/2021</td>
<td><a href="mailto:tom.dopp@gmail.com">tom.dopp@gmail.com</a></td>
</tr>
<tr>
<td>Emily Hudson</td>
<td>6/10/2022</td>
<td><a href="mailto:emily.hudson@gmail.com">emily.hudson@gmail.com</a></td>
</tr>
</tbody>
</table>
Online Editor: text formatting

If you have any questions regarding DOCX template formatting, proceed to **File**. Then, switch to the **Help** tab where you will find detailed information that explains how to format the text.
Set up Bots for documents generated from DOCX templates

Follow the instructions below to set up Bots for a document generated from a DOCX template:

1. Navigate to Bots in the Flow settings panel.

Then, in the Bots settings, select Add Bot.

2. Select the Bot you intend to use from the list and click Install (the Pre-fill from Salesforce Record Bot in the example).

Find detailed instructions on how to configure Bots with regards to Salesforce settings in the General Information section.

Note: If the Pre-fill from Salesforce Record Bot is added for a Flow with DOCX documents containing tables, it is possible to select the type of data to be mapped - single lines (set by default) or tables.
- If **Single line** is selected, the field tags should look as follows:
  \{Account\_name\}, allows using numbers and underscores.

- If **Table** is selected, the field tags from document, like
  \{for(Product)\}{Name}\{endfor\} should look as follows:
  \{Product[\]Name\}, i.e. without any dots or gaps.

Once a Bot has been successfully configured, click **Apply setup**.
Convert Document Generation templates to Contract Negotiation templates

Easily create modifiable DOCX templates with merge and fillable fields from Document Generation DOCX templates. Users can then collaborate on the resulting documents, allowing them to make changes and add suggestions as well as accept or decline them. The DOCX contracts can then be finalized as PDFs.

To do so, enable redlining by switching on the toggle.

In the Convert to Contract Negotiation pop-up, select Convert anyway to proceed.

>Note: If there are any Steps (roles) assigned in the Document Generation DOCX template, they will be discarded after converting to Contract Negotiation. The Bot settings will remain unchanged.
That's it! The Document Generation DOCX template will be instantly converted to a Contract Negotiation DOCX template.

Once a revision has been created, the Contract Negotiation functionality will be available for the document. You can also switch back to the Document Generation version of the DOCX template anytime you need.
DOCX template syntax

Use the following DOCX syntax to add field tags to your DOCX documents and markup your documents to work with a selected datasource.

What are tags used for?

airSlate document generation tags are the tools for creating document content automatically. Use tags to:

- Create merge fields
- Insert images
- Add tables with a dynamic number of rows
- Show or hide text paragraphs based on conditions
- Insert a list of items
- Insert HTML content

How to add tags?

Tags are included in figure brackets - {{TagName}}

Use the Tag Constructor panel to generate tag syntax automatically. Then, just copy & paste the tag syntax into your document.

👉 Tip: You can name your tags anyway you like. There is no required naming convention. Just make sure you give your tags descriptive names so that you can easily identify them when mapping fields with Bots.

How to populate tags?

Use Prefill Bots to populate your tags with content from external sources such as Salesforce, MS Dynamics 365, NetSuite, Google Sheets, MS Excel 365, Airtable, Smartsheet, MySQL, and many others. With airSlate, you can use multiple data sources to generate documents.
For example, you can grab a section of data from your Salesforce records, another section of data from NetSuite records, and generate documents that contain both sections of data.

## Types of tags

There are different types of tags that can be used for specific purposes. The following are the tag types along with examples of their usage:

<table>
<thead>
<tr>
<th>Type</th>
<th>Example</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text tag</td>
<td>{{TagName}}</td>
<td>Use this tag to insert a merge field. Text tag names don’t need to be unique. If you add multiple merge fields with the same name, they will all be populated with the same data.</td>
</tr>
<tr>
<td>For tag</td>
<td>{{for(ListName)}}&lt;br&gt;List Element Field1:&lt;br&gt;{{for(ListName)}}&lt;br&gt;List Element Field2:&lt;br&gt;{{for(ListName)}}&lt;br&gt;{{endfor}}</td>
<td>Use this tag to insert a list of items. This tag’s name should be unique per document.</td>
</tr>
<tr>
<td>Table tag</td>
<td>{{table}}&lt;br&gt;{{ColumnField1}}&lt;br&gt;{{ColumnField2}}&lt;br&gt;{{endtable}}</td>
<td>Use this tag to insert a table with a dynamic number of rows. This tag’s name should be unique per document.</td>
</tr>
<tr>
<td>If tag: Condition depending on variable or field value</td>
<td>{{if(ConditionFieldName='Value1')}}&lt;br&gt;Output 1&lt;br&gt;{{else((ConditionFieldName='Value2'))}&lt;br&gt;Output 2&lt;br&gt;{{endif}}</td>
<td>Use this tag to add conditional logic in your documents. With this tag, you can show or hide paragraphs of text depending on conditions. This tag’s name should be unique per document.</td>
</tr>
<tr>
<td>imageFile tag</td>
<td>{{imageFile(TagName)}}&lt;br&gt;{{imageFile(tagName).height(100).width(200)}}</td>
<td>This tag is used for inserting PDF, DOCX, JPEG, and PNG files as images. This tag’s name should be unique per document - unless you want to insert the same image multiple times.</td>
</tr>
</tbody>
</table>
By default, the image is embedded in its original size.

You can specify both the width and height. When one of the parameters is specified, the image will be resized and scaled proportionally.

If the image size is wider than the printable document area, this image will be scaled down to fit the visible page area.

<table>
<thead>
<tr>
<th>HTML tag</th>
<th>{{html:FieldName}}</th>
<th>Use this tag to insert HTML-formatted content (for example, from Salesforce Rich Text Field).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fillable field tag</td>
<td>Refer to the Fillable fields section</td>
<td>If you don’t want to drag and drop fillable fields into your document, you can use text tags syntax to create fillable fields.</td>
</tr>
</tbody>
</table>

**How to generate tags**

Add the tags you need to your DOCX templates when editing them in the online editor.

You can easily generate tags using the tag generator:

1. To do so, switch to the **Tags** panel in the upper right corner.
2. In the **Generator** tab, select the tag type. Then, enter the tag name.
3. Proceed with additional settings for the selected tag type.
4. Once finished, a tag will be automatically generated according to your settings.
5. Select **Copy tag** above the tag box to copy it. Paste a tag wherever you need in your DocX template.
How to use tags

Here you will find step-by-step instructions and examples of how to use various tag types to complete specific tasks in your DOCX documents. For example, calculating amounts using math functions, inserting files in your DOCX documents, etc.

Text formatting functions (date, number, currency)

You can format Text tags to show date, number, or currency values.

📍 Note: Be sure to only use ‘these’ quotation marks for strings:

- `dateFormat(DateFieldName, current_date_format, required_date_format)`

  ```
  {{dateFormat('Date1', 'yyyy-MM-dd', 'yyyy-dd-MM')}}
  “2019-16-10”

  {{dateFormat('Date2', 'yyyy-MM-dd', 'EEE, d MMM yyyy')}}
  Wed, 16 Oct 2019
  ```

- `numFormat(FieldName, string_format)` - accepts a numeric value and a string with a pattern for formatting.

  - `numFormat(FieldName, string_format, groupseparator_symbol, decimalseparator_symbol)` is also acceptable.

  ```
  {{numFormat(Field1, '###,###.00')}}
  16,326.20
  “###,###.00’ adds zeros if an expression has less symbols after a comma than is required.”

  {{numFormat(Field2, '###,###.##')}}
  16,326.23
  ‘###,###.##’ rounds expression to two symbols after a comma if the number has more.

  {{numFormat(Field3, '$###,###.00')}}
  $16,326.20
  Adds currency symbol before the number.
  ```

Use the Tag Constructor panel to see other available date and number formats.

How to insert today’s date

Use the `{{$today}}` function to insert non-editable today's date.
Table with a dynamic number of rows

The following is an example of a simple table with headers:

<table>
<thead>
<tr>
<th>Name</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>{{table(Products)}}</td>
<td></td>
</tr>
<tr>
<td>{{Name}}</td>
<td>{{Quantity}}</td>
</tr>
<tr>
<td>{{endtable}}</td>
<td></td>
</tr>
</tbody>
</table>

How to hide an empty table header or footer (using {{h}} tag)

If tables contain any tags below headers / above footers, that should be automatically hidden (in case tables have no records), add an {{h}} tag before each row with headers / footers.

To automatically hide empty tables that don’t contain at least one row of data (pre-filled from tags), refer to the syntax below:

| {{table(Products)}} |          |
| {{h}}Product List   |          |
| {{h}}Name           | Quantity |
| {{Name}}            | {{Quantity}} |
| {{endtable}}        |          |
How to calculate totals in a table column ($sumabove variable)

{$sumabove} displays the sum of numbers in cells above this tag.

```
{|table(Products)|
    |{h}Name| Quantity |
    |{Name}| {{Quantity}} |
    |{h}Total Quantity| {$sumabove} |
{|endtable|
```

It's possible to use the result from a {$sumabove} tag as a parameter in functions like

{{numFormat($sumabove, '$###,###.00')}}

### Conditional Logic

You can show or hide paragraphs of text depending on conditions. To create a condition, you can use a Field or Variable (for more complex cases) with operators such as =, !=, >, <.

For condition values, you can use set values, field values, or variables.

**Examples:**

```
{{if(AccountCity='Boston')}}Hello!
{{else(AccountCity='London')}} God save the Queen!
{{else}} Season's greetings! {{endif}}
```
Here is a summary report for your account:

```html
{{total=0}}

<table>
<thead>
<tr>
<th>Opportunity Name</th>
<th>Opportunity Status</th>
<th>Opportunity Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{{table(Opportunity)}}</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

{{OpportunityName}}  {{OpportunityStatus}}  {{OpportunityAmount}}

{{endtable}}

{{sumabove}}

{{total=$sumabove}}

{{if($total<50000)}}
  Happy new year!
{{else($total<100000)}}
  Happy new year! You are eligible for a massive discount.
{{endif}}

List

Use the For tag syntax to display a list of items. For example, a list of patient’s visits to the doctor.

Examples:

Doctor’s name: {{DoctorFullName}}
{{for(Visits)}}
Patient’s name: {{PatientFullName}}
Date of visit: {{DateofVisit}}
{{endfor}}

If you need to list all the doctors, and then add a list of visits to each of them, use nested For tags.

Examples:

{{for(Doctors)}}
  Doctor’s name: {{DoctorFullName}}
  {{for(Visits)}}
  Patient’s name: {{PatientFullName}}
  Date of visit: {{DateofVisit}}
  {{endfor}}
{{endfor}}

Variables

{{$variableName = VariableValue}} creates a variable, named variableName in the template environment and assigns it a value such as VariableValue.
If a variable named `variableName` already exists, its value is rewritten and is assigned the `VariableValue` name. You won’t be able to reach the previous value of `$variableName` if it’s overwritten.

- `{{$var = 5.48}}` Assign variable $var value 5
- `{{$var = 'Harry Potter'}}` Assign variable $var the string value 'Harry Potter'
- `{{$var = OpportunityName}}` Assign variable $var the value from data source
- `{{$var = 5 + 47 / 2 - 34 * CompanyPrice}}` Assign variable $var the calculated value
- `{{$var = true}}` Assign variable $var and assign it boolean value true
- `{{$var}}` Display value of $var
- `{{$var > 47}}` `{{endsWith(OpportunityName, $var)}}` Example of using variable in calculations and functions

**Math functions**

Here you’ll find a table with tags for calculating mathematical functions along with examples of how they work:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Accepts numbers</th>
<th>Example 1</th>
<th>Example 2</th>
<th>Example 3</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>*</code></td>
<td><code>Accepts numbers:</code></td>
<td><code>{{4 * 5}}</code></td>
<td><code>{{sqrt(9) * sqrt(16)}}</code></td>
<td><code>{{4 * (5 + 3)}}</code></td>
</tr>
<tr>
<td><code>/</code></td>
<td><code>Accepts numbers:</code></td>
<td><code>{{42 / 3}}</code></td>
<td><code>{{(5 + 3) / (1 + 1)}}</code></td>
<td><code>{{sum(3, 4, 5) / 4}}</code></td>
</tr>
<tr>
<td><code>%</code></td>
<td><code>Accepts numbers. Divides first parameter by second and returns the remainder:</code></td>
<td><code>{{42%2}}</code></td>
<td><code>{{33 % 4}}</code></td>
<td><code>{{34 % 4}}</code></td>
</tr>
<tr>
<td>Operator</td>
<td>Description</td>
<td>Examples</td>
<td>Results</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>--------------------------------------------------</td>
<td>-----------------------------------------------</td>
<td>------------------</td>
<td></td>
</tr>
<tr>
<td>+</td>
<td>Accepts two numbers or strings:</td>
<td>{2 + 6 * 7}</td>
<td>44</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>{4 + abs(-2)}</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>{sqrt(17 + 8)}</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>{‘Pineapple avenue,’ + 56}</td>
<td>Pineapple avenue, 56</td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>Accepts numbers:</td>
<td>{22 - 4}</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>{17 * 3 - 8}</td>
<td>43</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>{pow(6-3, 2)}</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>&gt;</td>
<td>Accepts numbers, return booleans:</td>
<td>{5 &gt; 7}</td>
<td>false</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>{5 &gt; 3}</td>
<td>true</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>{16 &gt; pow(3, 2)}</td>
<td>true</td>
<td></td>
</tr>
<tr>
<td>&gt;=</td>
<td></td>
<td>{6 &gt;= 4 + 2}</td>
<td>true</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>{8 &gt;= 4 +2}</td>
<td>true</td>
<td></td>
</tr>
<tr>
<td>&lt;</td>
<td></td>
<td>{5 &lt; 7}</td>
<td>true</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>{5 &lt; 4}</td>
<td>false</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>{sqrt(25) &lt; sqrt(4 * 9)}</td>
<td>true</td>
<td></td>
</tr>
<tr>
<td>&lt;=</td>
<td></td>
<td>{3 &lt;= 1 + 2}</td>
<td>true</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>{3 &lt;= 4}</td>
<td>true</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>{8 &lt;= 7}</td>
<td>false</td>
<td></td>
</tr>
<tr>
<td>!=</td>
<td>Accepts any type of expression, returns false if expressions are equal</td>
<td>{45 != 5 * 9}</td>
<td>false</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>{‘Jack’ != ‘Jackson’}</td>
<td>true</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>{2 &lt; 5 != 16 &lt; sqrt (49)}</td>
<td>true</td>
<td></td>
</tr>
</tbody>
</table>
### =,==

Accepts any type of expression, returns true if expressions are equal

- `{45 = 5 * 9}`
- `{{'Jack' == 'Jackson'}}`
- `{2 < 5 == 16 < sqrt(49)}`
- `{{'Jack Jackson' = 'Jack ' + 'Jackson'}}`

<table>
<thead>
<tr>
<th>expression</th>
<th>result</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>{45 = 5 * 9}</code></td>
<td>true</td>
</tr>
<tr>
<td><code>{{'Jack' == 'Jackson'}}</code></td>
<td>false</td>
</tr>
<tr>
<td><code>{2 &lt; 5 == 16 &lt; sqrt(49)}</code></td>
<td>false</td>
</tr>
<tr>
<td><code>{{'Jack Jackson' = 'Jack ' + 'Jackson'}}</code></td>
<td>true</td>
</tr>
</tbody>
</table>

The only accepted unary operation is boolean NOT that consumes a boolean and reverts it. Use the `!` sign to perform this operation.

For example: `!(2 > 5)` will return true.

### Fillable fields

In most cases, it's recommended to use the **Fields** panel for dragging & dropping fillable fields into your document. However, fillable fields can also be added using the text tags syntax if needed.

**Examples:**

- **Text field:**
  ```
  {{t:t;r:n;l:"text_field_1";}}
  ```
- **Signature:**
  ```
  {{t:s;r:n;l:"signature_field_1";}}
  ```
- **Initials:**
  ```
  {{t:i;r:n;l:"initials_field_1";w:200;}}
  ```
- **Dropdown:**
  ```
  {{t:d;r:n;l:"Dd_field_1";dd:first option,second option,third option;w:200;}}
  ```
- **Checkbox:**
  ```
  {{t:c;r:n;l:"checkbox_field_1";}}
  ```
- **Date:**
  ```
  {{t:dt;r:n;l:"date_field_1";w:150;p:"DD/MM/YYYY";}}
  ```

The table below shows the most widely used text tags in Document Generation along with the types of fields for which these text tags can be applied.
<table>
<thead>
<tr>
<th>KEY</th>
<th>What does it mean?</th>
<th>Accepted format &amp; values</th>
</tr>
</thead>
<tbody>
<tr>
<td>t</td>
<td>TYPE of the field</td>
<td>s  (for signature)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>i  (for initials)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>t  (for text)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>d  (for dropdown)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>c  (for checkboxes)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>dt or date (for dates)</td>
</tr>
<tr>
<td>r</td>
<td>REQUIRED</td>
<td>y  (for required)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>n  (for optional)</td>
</tr>
<tr>
<td>l</td>
<td>LABEL</td>
<td>“Field name”</td>
</tr>
<tr>
<td>dd</td>
<td>DROPDOWN</td>
<td>&quot;option1, option2,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>option3, ....&quot;</td>
</tr>
<tr>
<td>w</td>
<td>WIDTH</td>
<td>w  (width)</td>
</tr>
<tr>
<td>h</td>
<td>HEIGHT</td>
<td>h  (height)</td>
</tr>
<tr>
<td>p</td>
<td>DATE FORMAT</td>
<td>MM/DD/YYYY</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DD/MM/YYYY</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HH:mm</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MMM DD, YYYY</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MM/DD/YY</td>
</tr>
<tr>
<td></td>
<td></td>
<td>YYYY-MM-DD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>YYYY/MM/DD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>YYYY-M-D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>YYYY/M/D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>D.M.YYYY</td>
</tr>
<tr>
<td></td>
<td></td>
<td>D-M-YYYY</td>
</tr>
<tr>
<td></td>
<td></td>
<td>D/M/YYYY</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DD.MM.YYYY</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DD-MM-YYYY</td>
</tr>
</tbody>
</table>

Once a document has been generated out of a DOCX template, text tags added to your document will show up automatically being converted into fillable fields.

Such fillable fields can then be mapped as usual, while configuring your bots.