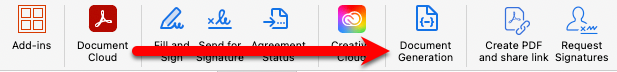
Tables, Tables, Tables

The Document Generation Word Add-in lets you quickly build templates that can turn your data into Word or PDF documents. You can easily insert tags and put together basic templates with just the Add-in. For a walkthrough on how to use the add-in, check out the [Add-in Demo](https://developer.adobe.com/document-services/docs/overview/document-generation-api/wordaddin/#add-in-demo).



But, instead of using the Add-in, I prefer to add tags manually. The Add-in is suitable for only elementary templates that don't use complicated logic. This is particularly true when creating tables. A table can be made from a JSON array within your data or from a JSONata expression that returns an array. Many tables require calculations within the rows or the ability to filter the data based on a particular piece of data. Manual tagging is necessary for these functions. The following examples show the various methods of creating tables.

## Examples:

See: tables-tables-tables.json for field reference.

### Referencing the Array:

When you add a table to a template using the Add-in, the tags can only reference an array in the JSON data. The "Price" column is blank because you can't perform calculations when a table is created this way.

|  | SKU | Product Name | Color | Unit Price | Price |
| --- | --- | --- | --- | --- | --- |
| {{products.quantity}} | {{products.sku}} | {{products.productName}} | {{products.description.color}} | {{products.price}} |  |

### Using Table Markers

You can also create tables using table markers; table-start and table-end. In the example below, we are seeding the table with the "products" array in the JSON data. Start by adding a 2-row table in Word. Be sure to set the top row as a header row and to repeat across pages. In the second row, add the table-start tag. It must be the first tag in the row that you want to be repeated, and the tag must reference an array in the JSON. As you will see in the following example, the array can be filtered. Then, the table-end tag must be the last one in the final column. This sets up the scope in which the JSONata functions can work. One row will be added for each item in the array.

As you can see in the last column, you can do table math when table markers are used. Also, the "Quantity" column is very narrow and hard to read, so I've put an example of the table-start tag just below. Because the tag is not within a table, it is ignored.

{% table-start products %}

|  | | SKU | Product Name | Color | Unit Price | Price |
| --- | --- | --- | --- | --- | --- | --- |
| {% table-start products %}  {{quantity}} | {{sku}} | | {{productName}} | {{description.color}} | {{price}} | {{quantity \* price}}  {% table-end %} |

### Using Table Markers with Array Filtering

To filter the data that will generate the table, use a JSONata expression. In the example below, we are filtering the products array based on the value of the color.

{% table-start products[description.color = 'Orange' or description.color = 'White'] %}

|  | | SKU | | Product Name | Color | Unit Price | Price |
| --- | --- | --- | --- | --- | --- | --- | --- |
| {% table-start products[description.color = 'Orange' or description.color = 'White'] %}  {{quantity}} | {{sku}} | | {{productName}} | | {{description.color}} | {{price}} | {{quantity \* price}}  {% table-end %} |

### Using Table Markers and Discarding Rows

Instead of filtering the data, a row can be discarded using the discard-row-if feature of Document Generation. When the condition in the discard-row-if(expr(*condition*)) returns true, the row is not added to the table. In the example below, we are displaying only the purple products. Notice that the discard-row-if directive can be in any column.

{% table-start products %}

|  | | SKU | | Product Name | Color | Unit Price | Price |
| --- | --- | --- | --- | --- | --- | --- | --- |
| {% table-start products %}  {{quantity}} | {{sku}} | | {{productName:discard-row-if(expr(description.color != 'Purple'))}} | | {{description.color}} | {{price}} | {{quantity \* price}}  {% table-end %} |