Nutrient Salesforce Docs

User Manual

Table of Contents

Introduction

Nutrient Salesforce Docs is a powerful, fully customizable document generation and management solution built specifically for Salesforce. Generate, edit, and sign documents directly within Salesforce, with advanced features like document redaction, bundling, and flexible workflows. Unlike other document generation tools, Nutrient Salesforce Docs adapts seamlessly to your existing Salesforce setup, providing extensive customization and an intuitive, low-code environment.

Feature Highlights

Some of the key features of Nutrient Salesforce Docs include, but are not limited to:

Template-Driven Document Generation

Effortlessly create documents from customizable Word templates directly within Salesforce. Nutrient Salesforce Docs enables you to pull data from any Salesforce object, allowing you to generate tailored, data-rich documents such as contracts, invoices, and reports. Streamline document creation and ensure accuracy with dynamic placeholders that auto-populate fields from your Salesforce records.

In-Salesforce Document Editing

Make quick adjustments to generated documents without leaving Salesforce. Our in-house bult in *Document Authoring* allows you to review and modify Word documents in the browser without needing to download them. This is especially useful for making last-minute changes to documents generated from Office templates.

Limitless Document Possibilities

Leverage Nutrient's document expertise directly within Salesforce. With Nutrient Salesforce Docs, you gain access to an advanced suite of tools, drawing from our powerful SDK that enables virtually limitless document functionality. Protect sensitive information with redaction, bundle multiple files into a single document, and apply watermarks for added security—all within Salesforce. These robust features put full control over your document workflows in your hands, allowing you to handle even the most complex document needs with confidence and ease.

Setup

This section outlines the setup process for Nutrient Salesforce Docs. To proceed, ensure you have access to the latest version, either from the Nutrient website or through a Nutrient representative. Please note that Nutrient was formerly known as PSPDFKit, so some files may still reference the former name. These will be updated in the near future.

Nutrient Solution Engineers are happy to assist with any step in the setup process.

Login to Salesforce Sandbox

In order to connect your Terminal to your Salesforce Sandbox, use a command similar to the following:

```
sfdx auth:web:login -a MySalesforceOrg -r https://ssc--pspdf.sandbox.my.salesforce.com
```

Note that your Sandbox may contain a lighting.force keyword which needs to be changed to my.salesforce. The following command will not work.

```
sfdx auth:web:login -a MySalesforceOrg -r https://ssc--pspdf.sandbox.lightning.force.com
```

Uploaded classes and components

Apex Class:

- PSPDFKitController.cls
  - Handles most of the data retrieval from Salesforce.
- ObjectFieldRetrieve.cls
  - A helper class.
- lookUpClass.cls
  - A helper class.

Lightning Web Components:

- pspdfkitFileSelector
  - Handles the importing of templates in the *Document Templates* class.
- pspdfkitGenerateDocument

    - Handles the generation of new documents in any object it is put into, e.g. Case.
- pspdfkitSignDocument
    - Once a document is saved in the *Pending Documents* list, opening the document there will open it with the pspdfkitSignDocument component, which is targeted at finalising and signing a generated document.

Visual Force Pages:

- PSPDFKit_InitPSPDFKit.page
    - Used in both in importing new templates as well as generating documents from templates. Loads the actual PSPDFKit for Web viewer.
- PSPDFKit_SignDocument.page
    - Used in particular for the finalising and signing documents after they are stored in Pending Documents.

Office Template Generation

This chapter explains how a user can create, upload and generate documents from Office Templates with the sample project provided by Nutrient. Any customization or deviation from this workflow needs to be implemented by the customer by modifying the classes mentioned in the previous chapter.

Document Template Object

During the Setup process of the Nutrient, a new custom object called *Document Templates* was created, which can be access through the Salesforce UI. Each record in Document Templates is meant to hold a new template.

There are a few things that have to be configured for each record. This includes the *Available Object*, which is the main object a template is supposed to pull data from, e.g. *Case*. The other is the *Related Objects* field that can have a list of objects that are related to the Case object, e.g. *Lien*, *Contact* etc. Related Objects are mostly those objects that can have multiple instances linked to the Available Object. Since the Related Objects and Available Objects are linked through some unique reference field, the Related Object should be entered in the following form, all separated by ";".

<RelatedObjectName>[<ReferenceField>] ; ...

Upload and Import Templates

Once a Document Template record has be configured, simply upload a new Office Template into the files section (the creating of Office Templates will be explained later in this document).

At this stage, the template has not been imported yet, the user needs to click on the *Import* button to see the uploaded document in the Nutrient Salesforce Docs Viewer. At this stage the user can cross check if all placeholders in the template have been recognized by Nutrient.

Finally, to save the template, the user needs to click on the *Save* button, which will populate the *PSPDFKit_TemplateJSON* field. This field will be later pulled to generate documents.

Creating New Templates

In order to create a template, the following placeholders can be inserted into an MS Office file using MS Word or a similar application.

Date

```
{{$DATE}}
```

A static placeholder that pulls the current date in a pre-defined format. If the format needs to be changed, the code in the Lighting Web Component *pspdfkitGenerateDocument* in the function *getAllRecordsNew()* has to be changed to the desired format. The same goes for any **Numbers**.

Regular Fields

```
{{Case Number+Case.CaseNumber}}
```

```
{{Client Name+Case.ContactEmail}}
```

Placeholders that are supposed to pull directly from the Available Object mentioned previously are considered *Regular Fields*. The are usually meant to pull a field that is directly on the Available Object with no lookup or reference to any other object.

The format consists of Display Name (grey in the sample above) that can be set to any value is meant only for the user generating the document to see what data has been pulled for which placeholder.

The Display Name is connected with a "+" sign to the actual field that data is supposed to be pulled from. In the case of a Regular Field, it's the name of the Available Object (blue) set in the Document Template Record followed by a "." and the technical name of the Salesforce Field (green).

Lookup Fields

```
{{Client FirstName Lookup+Case.ContactId.FirstName}}

{{Client Phone Lookup 2+Case.ContactId.MobilePhone}}
```

Lookup Fields operate very similar to Regular Fields with the difference that they have a third parameter (yellow) which is the field behind a lookup field (green).

Related Records

```
{{Evidences+BD_Evidence__c.Name}}
```

Related records operate similar to the previously mentioned fields with one key difference. The parameter marked in blue is not the Available Object, but one of the Related Objects set in the Document Template record.

Related Records with Lookup

```
{{Evidences  Lookup+BD_Evidence__c.Evidence_Source__c.FirstName}}
```

Operates similar to Regular Lookup Fields with the Related Object in blue instead of the Available Object.

Related Records - Grouped

{{Client+CMS_Role__c.Name}}

{{Client.Email+CMS_Role__c.Email__c}}

{{Client.MobilePhone+CMS_Role__c.Mobile_Phone__c}}

Sometimes multiple fields can depend on the selection of one *Parent* field, e.g. once a contact is selected, data related to that contact such as E-Mail and Mobile Phone number should be fetched automatically, without the user having to select all three options seperately. This can be done by using the Display Name of the parent in the child or dependant fields seperated by a ".".

Related Records with Lookup - Grouped

{{Builder+CMS_Role__c.Name}}

{{Builder.Street+CMS_Role__c.BD_Company_Role__c.BD_Street__c}}

{{Builder.City+CMS_Role__c.BD_Company_Role__c.BD_City__c}}

{{Builder.State+CMS_Role__c.BD_Company_Role__c.BD_State__c}}

{{Builder.Zip+CMS_Role__c.BD_Company_Role__c.BD_Zip__c}}

Similar as above, but now with Lookup Fields.

Signatures

<Sig1>

<Sig2>

The above placeholders have been configured to create signature fields automatically when a document is generated. The placeholders need to contain the word "Sig" in them, followed by any number or text afterwards. These signature fields are not

assigned to any user at this point. This workflow is left for implementation by the user and can edited in the file *PSPDFKit_InitPSPDFKit.page*.

Conditions

```
{{#Condition+CMS_Case__c.Damages_Expert__c.Name!=””}}
      Condition is true!
{{/Condition+CMS_Case__c.Damages_Expert__c.Name!=””}}
```

Conditions render the part of the document that is between it's start and end tag, if the condition is true. It has Display Name that has to contain the word "Condition" followed by any optional character and a condition that is checked against after the "+" sign. All available operators can be found in the file *pspdfkitGenerateDocument.js* in the function *evaluateCondition()*.

Conditions can either compare two Salesforce fields against each other or a Salesforce field against a static value.

Loops

{{#Loop1}}

This is a Loop

Name	Email	Phone
{{Member_Name}}	{{EMail}}	{{Phone}}

{{/Loop1}}

Due to the complex nature of Loops and limited space on a template, Loops operate in a different way. There is a start tag that has to contain the word "Loop" in it followed by placeholders that are merely the Display Names for the placeholders, not the actual Salesforce Fields.

The Salesforce Fields that should be pulled from have to be entered in the input fields

after a template has been imported in the Document Templates record. This design choice was made to save space in a template where multiple placeholders on the same line as in the table above can lead to cramped templates.

Note: The functionality to filter objects pulled from Salesforce has only been partially implemented and will be delivered with a future update.

Upper or Smaller Case Placeholders

```
{{Builder+CMS_Role__c.Name[lower]}}
{{Builder+CMS_Role__c.Name[upper]}}
```

The text of a placeholder can be automatically turned to upper or lower case by defining the keywords *upper* or *lower* in square brackets after the Salesforce field name (see example above in red).

Generating Documents from Templates

In order to generate documents from templates, the Lightning Web Component *pspdfkitGenerateDocument* has to be placed at any location in the *Available Object*. Once placed, a dropdown will present all documents that can be used to a generated a new document. This list can be filtered by modified or filtered by modifying the *pspdfkitGenerateDocument* component.

After selecting a template, a new document will automatically be generated and shown in the Nutrient Salesforce Docs Viewer along with the data that has been pulled from Salesforce in input fields on the left side to the viewer. The Display Name set in the placeholders can be used to identify which data went into which placeholder.

If the placeholder was a for a field on the Available Object, it will be shown as a text input field. If it was for a Related Object, where the user has to pick from one of multiple options, a dropdown will be shown with all available options.

Note: If the placeholders were not defined correctly, e.g. the syntax of a placeholder was not followed or the Salesforce field name is wrong, no data will be shown.

The Nutrient Salesforce Docs Viewer will show an initial version of the document generated, which will not include values from Related Objects as these have to be selected by the user generating the document. In order to regenerate the document

after all values have been selected, the user has to click on the *Generate* button to see an update version of the document.

Once a document has been generated, the user can click on the pen icon which will open a new popup that will let the user edit the document in the Nutrient Document Authoring tool, an in-house built document editor similar to MS Word. This tool can be used to enter or delete new content like sentences or paragraphs.

The button *Save to Pending* will save the generated document to Salesforce Files and create a new record in the *Pending Documents* object. This behaviour can be changed in the Visual Force page *PSPDFKit_InitPSPDFKit.page* by editing the behaviour of the Save button.