RAVENAPPS

100% native

.docx + PDF

Document Generation

for Salesforce

User Guide

Light. Fast. Native.



Native document generation for Salesforce

Version 1.1 · November 2026 · raven-apps.com

Contents

1. Introduction 3

2. Quick start 4

3. Building templates 5

4. Writing the SOQL query 7

5. Generating documents 8

6. Troubleshooting 9

7. Pricing and trial 10

8. Support and resources 10

1. Introduction

DocGen creates editable Word (.docx) documents straight from your Salesforce records. Order forms, quotes, letters, account summaries. One click on the record page, and the finished document is saved against the record and opened for review.

Everything runs inside Salesforce. Your data never leaves the platform.

Why we built it

We kept hitting the same problem ourselves, and customers flagged it too: copying Salesforce data by hand into order forms and other outbound documents. We wanted a quick, native way to produce documents that can still be edited afterwards. So we built one.

Why DocGen

- **100% native.** Built in Apex and Lightning Web Components. No middleware, no external services.
- **No custom objects.** Templates and generated documents live in standard Salesforce Files.
- **No heavy integration.** Drop a component onto a record page and you are done.
- **Editable output.** .docx files stay easy to tweak, for example updating T&Cs before sending. PDF output is also available.
- **Works on mobile.** Generate documents from the Salesforce mobile app.
- **One package.** DocGen ships in the same managed package as Grids, so there is a single RavenApps package to install and upgrade.

How it works

DocGen has three moving parts:

- **A template.** A .docx document, designed in Microsoft Word or Google Docs, containing merge tokens such as `{{Account.Name}}`.
- **A SOQL query.** Tells DocGen which fields to fetch from the record and its related records.
- **The DocGen Button.** A Lightning component on the record page that links the template and query together.

When a user clicks the button, DocGen runs the query for the current record, replaces each token with its value, repeats table rows for related records, and saves the result to the record's Files.

2. Quick start

You can have a working document button on a record page in around ten minutes.

Step	What to do
1	**Install or upgrade** the RavenApps package from AppExchange or the Package Version page.
2	**Create a .docx template** in Microsoft Word or Google Docs, or start from one of the sample templates (see below).
3	**Edit the record page** in Lightning App Builder and add the **DocGen: Manager** and **DocGen Button** components. Save and activate.
4	**Upload the template** using the DocGen: Manager component on the record page. It's stored in the **DocGen Templates** file library.
5	**Configure the button** in Lightning App Builder: refresh the template on the DocGen Button component and select your template, add your SOQL query, and set the Component Name (the button label users see).
6	**Save** the page, then open a record and click the button.

The DocGen: Manager component

DocGen: Manager is where admins add and update templates. Uploading through it saves each template as a standard Salesforce File in the **DocGen Templates** library, so there's no need to upload to Files manually or copy File IDs.

Sample templates

The DocGen Templates library comes with ready-made samples you can use as they are or adapt, including an Order Form, Invoice, Non-Disclosure Agreement, Master Services Agreement, Inventory / Stock list and Project Kickoff. Copying a sample is the quickest way to start a new template.

Updating a template? Upload the new version through DocGen: Manager, then refresh the template on the DocGen Button component so it uses the latest version.

Tip. DocGen: Manager is an admin tool. Use Lightning App Builder's component visibility settings to show it only to admins, so everyday users just see the DocGen Button.

3. Building templates

A template is an ordinary .docx document. Design it in **Microsoft Word** or **Google Docs** with your own fonts, logo, tables, headers and footers, then add tokens wherever data should appear. Everything that isn't a token comes through exactly as you designed it. In Google Docs, use *File > Download > Microsoft Word (.docx)* before uploading the template.

Open documents in the same app you designed in. Word and Google Docs lay out .docx files slightly differently, for example spacing, table widths and page breaks. Open generated documents in the same application the template was designed in, so they look exactly as intended. If your team works in Google Docs, design the template there; if they use Word, design it in Word.

Field tokens

Wrap a field's API name in double curly braces. Use dot notation to reach parent records.

Token	Output
`{{Name}}`	The record's Name field
`{{Amount}}`	A currency field, formatted with its symbol, e.g. US$12,500.00
`{{Account.Name}}`	A field on the parent Account
`{{Account.BillingAddress}}`	A compound address, printed over multiple lines
`{{Account.Primary_Contact__r.Name}}`	A field through a custom lookup, using the __r relationship name
`{{Custom_Field__c}}`	Custom fields work the same way, using the API name

Important. Every field used in the template must also appear in the SOQL query. A token whose field isn't queried renders as blank.

Example: template and result

Here's part of the sample Order Form. Tokens in the template on the left become record data in the generated document on the right.

TEMPLATE		GENERATED	
Bill To:	`{{Account.Name}}`	**Bill To:**	Axentia Finance
Address:	`{{Account.BillingAddress}}`	**Address:**	901 Redwood St. Palo Alto CA 94301 USA
Phone:	`{{Account.Phone}}`	**Phone:**	+9012345678
Primary Contact:	`{{Account.Primary_Contact__r.Name}}`	**Primary Contact:**	Abbi Beldon
Email:	`{{Account.Primary_Contact__r.Email}}`	**Email:**	abeldonky@over-blog.com

Built-in functions

Token	Output
`{{TODAY()}}`	Today's date in the running user's time zone, as yyyy-MM-dd (e.g. 2026-11-16)

Repeating rows for related records

To list related records, such as products on an Opportunity, wrap a table row in `START` and `END` markers named after the child relationship in your query. DocGen repeats the row once per related record.

```
| {{START:OpportunityLineItems}}{{Product2.Name}} | {{Quantity}} | {{TotalPrice}}{{END:Opport
unityLineItems}} |
```

Inside the loop, tokens refer to fields on the child record. Keep the markers in the same table row as the tokens they wrap, with the header row and any totals row outside the loop.

Formatting rules

- **Currency** fields show the currency symbol and always two decimal places (US$200.00).
- **Multi-currency orgs:** each record's own currency is used. Include `CurrencyIsoCode` in the main query *and* in every subquery that returns currency fields.
- **Numbers** drop unnecessary trailing decimals (a quantity of 2.00 prints as 2).
- **Addresses** from compound fields such as BillingAddress print as a multi-line block.
- **Fonts** embedded in the template are carried through to the generated document. Use TrueType or OpenType fonts.

Template best practice

- Type each token in one go. If spell-check or autocorrect splits a token across formatting runs, retype it without stopping.
- Use tables for layout. Borderless tables give you clean, aligned label/value blocks.
- Put a footer on longer documents, such as *Page X of Y* at 9pt grey.
- Keep a test record with every field filled in, so you can check the full layout quickly.

4. Writing the SOQL query

The query tells DocGen which data to fetch. Write it against the object the button sits on. DocGen runs it for the current record automatically, so there's no need for a WHERE clause on the record Id.

Example: Opportunity with products

```
SELECT CurrencyIsoCode, Amount, Account.Name, Account.Phone,
       Account.BillingAddress,
       (SELECT CurrencyIsoCode, Product2.Name, Quantity, TotalPrice
        FROM OpportunityLineItems)
FROM Opportunity
```

This query supports a template using `{{Amount}}`, `{{Account.Name}}`, `{{Account.Phone}}`, `{{Account.BillingAddress}}` and a loop over `OpportunityLineItems`.

Query checklist

- Every token in the template has a matching field in the query.
- Subquery names use the **child relationship name** (plural, e.g. `OpportunityLineItems`, `Contacts`, `Cases`), and match the START/END markers exactly.
- Include `CurrencyIsoCode` wherever currency fields appear, in multi-currency orgs.
- Add `ORDER BY` inside a subquery to control row order, e.g. `ORDER BY SortOrder`.
- Test the query first in Developer Console or the Salesforce CLI (adding a temporary WHERE Id = '...').

5. Generating documents

Once the button is on the page, generating a document takes one click.

For users

- Open the record, such as an Opportunity.
- Click the DocGen button, for example **Order Form**.
- The document is created, saved to the record's **Files** related list and opened for review.
- Edit the .docx if needed, using the same application the template was designed in (Word or Google Docs), then send it.

Output formats

DocGen produces editable .docx files by default. PDF output is also supported, for documents that should be sent in a fixed format.

Multiple documents on one page

Add a separate DocGen Button component for each document type, such as Order Form, Quote and Welcome Letter. Each button has its own template, query and label.

Permissions and sharing

DocGen runs with the clicking user's access. Users need:

- access to the DocGen Templates file library;
- read access to every object and field in the query;
- edit access to the record, so the generated file can be attached to it.

Because templates and outputs are standard Files, your existing sharing, storage and retention settings apply with no extra configuration.

6. Troubleshooting

Symptom	Likely cause and fix
A token prints as blank	The field isn't in the query, or the API name is misspelt. Add it to the SELECT and check the spelling and case.
A token prints literally, e.g. {{Name}}	The editor has split the token across formatting runs. Delete it and retype it in one go with the same formatting.
New template doesn't appear on the button	Refresh the template on the DocGen Button component in Lightning App Builder, select it, and save the page.
Wrong currency symbol	CurrencyIsoCode is missing from the query or subquery. Add it to every SELECT with currency fields.
Repeating rows don't appear	The START/END name doesn't match the subquery relationship name, or the markers are in different rows.
Only one line item appears	The loop markers wrap more than one row, or sit outside the table. Keep them inside a single row.
Address prints on one line	A non-compound field is used. Use the compound field (e.g. BillingAddress) or separate street, city and postcode tokens.
Layout looks different from the template	The document is being opened in a different application from the one the template was designed in. Open it in the same app (Word or Google Docs), or redesign the template in the app your team uses.
The file won't open	The template may contain unsupported content or broken tokens. Re-save the template as .docx and retest with a simple version.
Button errors with an access message	The user can't see the template file or a queried field. Check library membership and field-level security.
Date is not in my preferred format	{{TODAY()}} uses yyyy-MM-dd. Contact us if you need another format.

Still stuck? Send us the template, the query and a screenshot of the error, and we'll take a look.

7. Pricing and trial

DocGen

$5 user/month

Already have Grids

$3 user/month

Billed yearly, plus 15% support and maintenance. Minimum purchase of 10 users.

Try it in a sandbox

Install DocGen in a sandbox and validate it against your own data and templates. There's no trial clock and no usage limits.

One package, one upgrade

DocGen and Grids ship in a single managed package. Existing Grids customers get DocGen by upgrading, with nothing extra to install.

8. Support and resources

Resource	Link
Install or upgrade	Latest package version
AppExchange	Listing and reviews
Book a call	calendly.com/raven-apps/45min
Contact us	raven-apps.com/contact
Demo video	youtube.com/@Raven-Apps
Website	raven-apps.com

Have a use case in mind? Get in touch for a demo or a chat.

Thanks for using DocGen. Light. Fast. Native.