
White Paper | June 2020
INTRODUCTION

This guide will walk you through how to determine which document automation or document generation product is best for you. No one product, not even ours, is best for all use cases.

This guide only discusses the template design component. While just a part of any solution, template design generally the most important part as it’s where the lion’s share of users’ time will be spent and limitations in this restrict the types of documents that can be created.

DEFINITIONS

**Document Automation** (also known as document assembly) is the design of systems and workflows that assist in the creation of electronic documents. These include logic-based systems that use segments of pre-existing text and/or data to assemble a new document.

**Document Generation** is the process of creating hundreds, thousands, or even millions of personalized and distinct documents for internal or external use from a single template. While Document Generation is a subset of Document Automation, for some products (not all) you can't get just the Document Generation component of a Document Automation solution.


Going forward, the word docgen will be used to stand for Document Automation & Document Generation in this guide. When something is Document Automation only, the word docauto will be used.

**Tags** are elements placed in the template (DOCX, PPTX, XLSX) that the docgen system acts on when generating a document. These tags can be data to insert, business logic rules to conditionally display or suppress content, and much more. Each vendor has their own term for “tags.”

NOTE:

Going forward, the word docgen will be used to stand for Document Automation & Document Generation in this guide. When something is Document Automation only, the word docauto will be used.
THE DESIGNER – MICROSOFT OFFICE

Every modern docgen product uses Microsoft Office as the template designer. While you can find a few very old products that have their own designer, you want to limit your consideration to those built on Office as it is far superior.

Some solutions work with Word, Excel, & PowerPoint while others are Word only. If you need Excel & PowerPoint, then obviously, go with a solution that supports them too. If you only need Word, think carefully if you might want Excel or PowerPoint someday in the future.

Again: if you go with a Word only solution, be very sure you won’t ever want Excel or PowerPoint. Ever!

Google Docs, Star Office, etc.

The docgen solutions that have a separate add-in or no add-in can usually work with any word processor that can save as a DOCX file. It all tends to work exactly the same. For a full Word clone, this can work every bit as well.

Google Docs in this case though tends to be problematic because Google Docs does not have the layout and formatting capability of Microsoft Word. Not even close. Your limit here is not the docgen app; it’s Google Docs. For most use cases, Google Docs is not up to the job.
CRITICAL FUNCTIONALITY

The following 8 items are key to the success of any docgen solution. If any of these is a mismatch for your needs and requirements, you will at best have a lousy solution. And you could very well fail. Understanding how each aligns to your use cases is critical to your success.

DESIGNER ADD-IN

Some docgen solutions include an add-in to help you place & edit the tags in the template. These come in two flavors; one much better.

First, some docgen solutions have no add-in to assist in crafting tags. You usually end up with notepad open where you write all the various tags and you copy from there and paste into Word. And for special uses, you type in, from memory or other notes, the additional properties.

This “no add-in” approach is slow, painful, & error prone. If you have 5 templates, each with 5 tags – then no big deal. But if every month you’re creating 100 templates, each with 150 tags, you’re now in hell.

Not only is it slow & expensive, but because it is a death march, designers will not put in the effort to make a template shine. They just want to be done.

The second approach (much better) is a second application (usually in a browser) that helps you write tags. You must still have to copy & paste between this second app and Word, but the add-in provides all possible choices for tags and helps you write your queries.

Not all the side-by-side add-in approaches are the same. Play with each carefully to see how it works for you, not in simple sample cases, but in the more complex templates you will need to create.
The third approach (best) is an add-in that becomes part of Word; adding additional tabs to the ribbon. This makes adding and revising tags a breeze because it works on the tag in the template. And while helping to write each tag, it can do so in the context of where it is in the template.

The incorporated add-in approach is by far the best but is limited to Office on Windows.

This add-in is one of the two features (the query wizard below is the other) that determines how much time your team will spend designing templates, day after day, week after week, year after year. If one approach is 15 seconds longer, and you are going to create 500 templates each with just 35 tags (that’s low), that’s 73 hours.

**CODE BEHIND**

While all the Document Generation solutions require you write code to call them (docauto is a no-code solution so not an issue), some of them require additional code for each template. This is called “code behind.”

In some cases, this code behind is defining different data specifications, such as you now also need the hire date. For these solutions, you don’t need code for each template, but a fair number of times templates will require additional data, or data ordered differently, and you have a code change.

Even worse, some require code behind for each template. Therefore, each new template means additional code. This is a giant hit.

Why? First you have programmers involved in template design. That’s expensive and slows the process down. Second, each new template requires rebuilding your application and pushing it through test & staging.

The one advantage to code behind is the developers can build data on the fly as it’s needed, including data generated according to business rules within the code. But in almost all cases, doing so directly in the template, as opposed to in the code behind, is superior.

In other words, you want the template file to be everything.
DRAG & DROP DOCLETS

One (or several) users can create content for a template that are saved off. Then they or other users can drag those saved doclets to drop onto a template. This provides template designers a way to create very complex templates easily by dragging the needed components. It also eliminates repetitive tasks.

For each docgen app, evaluate their drag/drop on the following criteria:

1. How do you create a doclet?
The best solution is to select content in Word and save that as a doclet. If it’s more restrictive than this, will those restrictions stop you from creating very useful doclets?

2. Does it bring the full formatting of the doclet into the document it is dropped into?
This is actually a very hard thing to do in Word if the doclet uses styles that exist in the template with the same name – but different settings.

3. What can be saved?
Just template content? Or can you also save datasources, parameters, and more?

4. After you drop is it complete? Or do you need to perform additional steps? For example, if a doclet uses a different datasource, is that datasource now also tied to the template?

5. Can doclets in a template be updated? If a doclet is the company logo and the logo changed, can all the templates using that doclet be updated to the new logo universally?

FROM OUR CEO

The optimum approach is linked doclets so you see the imported doclet in full, and it automatically updates.

David Thielen

windwardstudios.com
The dropped doclets come in several flavors. The optimum are linked doclets where the content of the doclet is displayed in your template in full, fully laid out and formatted. And as it is linked, when the doclet itself is revised, that change immediately appears in your template and is used in every generated document.

Once you drop a doclet into your template, you can adjust it any way you wish from formatting to tags in the content. But if the original doclet is changed, that change is not applied in your template. In some uses this is preferable when you don’t want changes applied to existing templates.

The third approach is there is a tag that will import the doclet. You don’t see the contents of the doclet in your template, but when the template is processed, it will pull the live copy of the doclet. This is valuable when you have a select that will determine which doclet to import. This is useful for cases like you need to pull in content based on the State the recipient of the document lives in.

The optimum of course is to have all three flavors available to use each as appropriate.
QUERY WIZARDS

Your most common activity creating templates will be writing the queries to select the data. You do this to select blocks of data such as all stocks you hold for a portfolio statement. You also do this for conditional logic in the template such as adding insurance requirements for an offer letter if they reside in California. Or, when placing a name in loan papers.

Some docgen products do not have query wizards. With no wizards, then template creation is a developer-only task. And for developers, it will be slower. No wizards mean you can never turn template creation over to business users.

In the case of conditional tags (if, switch, etc.) make sure it also works well on elements returned by other tags (usually the iterative tags). Because in this case, it’s not a query of the data, it’s a condition on data already returned.

Finally, keep in mind that no matter how brilliant the query wizards are, the user will also generally struggle with the structure of the data (the metadata). This can be displayed to the user, but they still need to learn what is where. Reducing what metadata is displayed, providing the descriptions for each node in the metadata, etc., can make the difference between a usable and unusable solution for business users.

FROM OUR CEO

You will do this hundreds of times in complex templates. Thousands of times across all the templates. You want this to be quick & easy. This functionality, more than everything else put together, determines how much time you will spend designing templates, and how pleasant it is.

– David Thielen

When you evaluate different docgen solutions, have a business user use the system to craft the queries and see how well they do, which is key to see if they can learn it and then be successful on their own.
MULTIPLE DATASOURCES

If you have a single datasource, then skip this section—you don’t care.

Ok, so you have multiple datasources: for example Salesforce & Marketo. And you have documents you want to populate with data from each. In this case you must get a docgen solution that lets you have tags in a single template that are marked for which datasource that tag is to be applied to.

Some providers implement this in two passes: first by applying all the Salesforce tags and then starting over and applying all the Marketo tags. This works fine if you are not intermixing the data.

Sometimes you need to intermix the data: for example, if your document lists all Account Executives (from Salesforce) and then within the data for an AE it lists the emails they were sent (from Marketo). Then you need a solution that processes all datasources simultaneously.

If you have multiple datasources, you almost certainly will eventually need a solution that processes multiple datasources simultaneously. If it’s not a must-have today, it probably will be a must-have in a year.
TAGS START & END LOCATION

Some tags have a start and end location, such as the if and forEach (iterative) tags. Generally, these are used to repeat or conditionally include a row in a table or a paragraph of text. All solutions do this.

But as time goes on and you create more advanced and complex templates, you will find yourself wanting to start the iteration in the middle of a table—or an if that removes two cells and adjusts the table correctly.

In addition, you almost certainly will need a forEach (iterative) tag that adds columns in a table, as opposed to rows. You may want a column for each product or each month in a dataset. Finally watch out for any limitations on combinations. At the start you need a single forEach tag. A year later you are nesting five forEach tags within each other as it’s the only way to get what you want.

This is an area where it's impossible to give guidance on what you may someday need. **Your best bet is to select a solution that has no limitations on the start & end location.**

OPTIONALLY HIDING CONTROL TAGS

For a simple template, this doesn't matter (much). But as the logic expands in a template, you find that you are adding a lot of control tags. The most common are the iterative (forEach) and conditional (if) tags. But even a moderately complex template will also have numerous query and set tags along with several additional tags.

These tags, if displayed, pollute the template and enlarge the layout in the template. Usually you'll find the template looks quite different from the final generated report. This makes it difficult to truly imagine the final document from the template. It’s frustrating to have to constantly run test documents to see what you’re going to get.

You’ll be much happier if the designer can at the click of a button hide or show the control tags. Show them when you’re working on the template logic. Hide them when you’re working on the final layout and formatting. This option will save you time and more importantly will make the design experience more pleasant.

Even on something as simple as this table, the ability to hide the control tags is a clear benefit.
IMPORTED TEMPLATES

The best way to use content across multiple templates is to have that content in a child template that the parent templates all import. These imported templates can be brought in as an explicit filename or as a data query that returns the filename.

Trust me: unless your needs are incredibly simple, you need this. You can work around it even if you repeat the same content in 100 templates, but you’re giving yourself too much extra work when wording changes due to company directives or legislation.

One critical detail on imports: does the system process tags in the imported child template? If all your child templates are static text (legal clauses), then this does not matter. But if you need to include anything live (a person’s name, a date, a state of residence), then you need a solution that process tags in the imported child template.

Finally, for MS Word only, how does it handle style mismatches? If the parent has the Normal style set to Times New Roman 12pt and the child has Normal set to Verdana 10pt, then what should the child paragraphs be styled as? This can be a royal pain because different users never have their styles matching.

Some systems convert the child to the parent formatting. Some retain the child formatting. And some (best solution) give you the option of either. The option is best but if it’s forced one of the two ways, make sure the system you get works that way.

Not having the expected styling on output is guaranteed to get upper management upset.
IMPORTANT FUNCTIONALITY

One of the below might be critical to your use case. Several might be useful. But in most cases, none is a must-have. They do, however, provide a picture of the breadth of each product.

FUNCTIONS (MACROS) INCLUDED & CUSTOM

For the solutions that allow queries in the tags, you want one that also supports complex functions operating on the data. And not just simple functions like SUM() and COUNT() but most of what’s available in Excel. You will use Text and DateTime a lot.

In addition, can you add your own functions? Adding custom functions is often a significant component of providing a simple & easy design experience to business users. It’s also a lot safer. For complex calculations you write it once in the function and test it carefully. No worries about someone screwing it up writing it by hand in a template.

DOCUMENT LOCKING

If you want to create DOCX or XLSX files that an employee can edit parts, document locking is incredibly valuable. For example, you are generating portfolio statements and the legal disclaimers and actual financial results must not be changed, but there is a paragraph where the financial advisor can write up more summarizing the performance.

In this case, some of the solutions will carry document locking in DOCX & XLSX (PPTX does not have this) over to the output. So, if the template has locked all except one paragraph, then the generated DOCX will be locked except for that one paragraph.

ACCESS PROVIDERS

All of the products (I believe) support reading files from BASIC, Digest, Negotiate, & Oauth2. But what about a special Authenticate & Authorize you created in your company for one set of files? Or something special to get to a JSON file from a REST service that is home-grown?

First off, make sure the solution supports the standard protocols you use. You should get a yes. And if that’s all you have—fantastic; you can skip to the next section. If you have a home-grown A&A, find out what needs to be done to have the system access it. This is a custom Access Provider. And make sure that the same Access Provider is used for reading data files (XML & JSON), accessing OData, and importing files (templates & pictures).

FROM OUR CEO

Having the document locking functionality tends to make your lawyers very very happy. It eliminates a source of serious legal liability.

– David Thielen
**VALIDATION, ERROR & WARNING HANDLING**

What is provided here is all over the board. And it’s difficult to get specific about what is most useful to you, as opposed to the next person. The best advice here is just look at what they have and try it out when evaluating.

One tool is validating a template. Not running it, but inspecting it and providing information on errors found. A second tool is to generate the document and deliver a list of errors and warnings. For example, if some content is placed off the page, it was rendered but you don’t see it. In this case it’s useful to have a listing of content off the page.

In this category you can include tag settings: what to do if a select fails, returns nothing, etc. Some of these are particularly useful but in other cases, you can find yourself investing more time than it’s worth.

**PROCESS EMBEDDED OFFICE OBJECTS**

What if you are generating portfolio statements using a Word template? It has descriptive text, a chart showing performance, legal disclaimers, etc. But where it has a table showing the actual numbers, you want to place an embedded spreadsheet with the numbers.

Why? Because this way the recipient can open that spreadsheet and then, using Excel, measure that data any way they want. It’s a much-improved portfolio statement and something that makes the recipient go WOW.

If you want this, verify that the vendor you select not only carries embedded objects to the output, but that the embedded object, if a DOCX/PPTX/XLSX file, has tags in it processed. To make good use of this functionality the embedded object must be treated as a live template, not a static document.

If fully implemented, the output to any format, such as PDF, will include the displayed embedded object.

This is generally not required, but it is an opportunity to make people love what you create.

**WORD FORM FIELDS**

This is a DOCX -> PDF issue. Do you need to have form fields in the DOCX such as drop down, list or check box become the equivalent thing in PDF output? If so, you need to verify that this feature is supported.

In addition, make sure that the initial content/value in the form field can be set from data. If it’s just static values from the template, that’s usually insufficient for all use cases.

And a suggestion: When you need an empty or checked box depending on data, don’t use a form field. Use the Wingdings characters   and  .

**EXCEL REFERENCES & PIVOT TABLES**

This is two XLSX -> XLSX issues. First, verify that a formula like SUM(D5:D5) expands to SUM(D5:D15) for the case where the row 5, inside an iterative loop, becomes rows 5 to 15. It’s very useful to have the formula adjusted (some products just write the literal value) on the output. This way, when someone adjusts say D7 to see what happens, all the formulas now adjust to that difference.

The same for pivot tables. If a pivot table is for D1:H5 and the generated XLSX now has those rows as D1: H125, the pivot tables are adjusted to match. This is necessary to use the pivot tables in the generated XLSX.

If you’re going to generate XLSX for Excel Power Users, this is key.
CAPABILITIES

These are mostly yes/no items. If you have data in SQL, the product needs to support SQL and you don’t care if it supports JSON. Same for programming language & output formats. So fast check-off here.

PROGRAMMING LANGUAGES

This is not an issue for docauto, just document generation.

There are three ways to call a docgen engine: Direct calls to a library, calls to a RESTful server on premises, and calls to a hosted (SAAS) RESTful server. Ask if they have what you want.

One note on Hosted solutions: You will be sending data to that system. First, you want to make sure that the vendor is providing adequate security. Second, if your data is not allowed to go outside your country or region (E.U.), find out not just where the default server is, but also the failover server.

SUPPORTED DATASOURCES

If all your data is JSON (or any other type), you don’t have to worry about what else the system can access. With that said, everything is getting more interconnected and odds are sooner or sooner you will have to access other datasource types.

Life is a lot safer if the solutions can use data from SQL, XML, JSON, & OData. And why OData? 150 other vendor’s datasources, from ACT to Salesforce to Zoho. Not a deal breaker but it will turn out to be useful.

See if you can create datasets from datasources. This is akin to views in SQL but you are creating them in the template (no DBA needed). And you want them for XML, JSON, & OData too. A good guide to how robust the dataset implementation is–do they basically become another datasource? If so, that’s a full implementation.

Furthermore, it can take time and bandwidth to download the metadata from a datasource. We saw one DB2 database take 28 minutes to download the full metadata (yes – truly!). If you have datasources with large metadata structures, find out if they have a way to read the schema once and reuse that. (This is unlikely to ever be needed for XML or JSON–it’s SQL, OData, & any custom datasources.)

Finally, for XML, make sure it uses the XML schema if one is available.
OUTPUT FORMATS
Check that it renders in the output formats you need. Everyone does PDF, HTML, DOCX, XLSX, & PPTX (last two if they support that template type). Additional output formats might be useful, but odds are you'll never need them.

Check the accuracy of the PDF output. Everyone is imperfect on this. And in their, and our, defense, Microsoft does not document how Word calculates page layout. It does not specify the calculation between 2 lines of single-spaced text. And it's impossible to reverse engineer accurately—Word is clearly performing complex calculations, not just using the font metrics.

Everyone does their best. Some come closer than others. Look for a good match but accept it won't be perfect.

MISCELLANEOUS
Are you still here? Wow — congratulations! We're now into some features that you may find useful but are unlikely to be major. But these do make good tiebreakers in your decision. And you may find one of the below to be major; for example, output that must be auto-hyphenated.

PARAMETERS
All products have a way to pass parameters to the template to use in the queries. Check that they have all the data types you need (they probably do).

Check that parameters can be set in a select as both a parameter (avoid injection attacks) and as a string substitution if desired. Setting as a parameter is valuable not only to avoid an injection attack, but to handle the cause of passing the name O’Malley.

CONDITIONAL FORMATTING
Excel has conditional formatting for a cell. But Word and PowerPoint do not. If you need conditional formatting, check if the solution you are looking at has it, and if so, if it’s sufficient.

AUTO-HYPHENATION
For output to PDF and a printer, if you want auto-hyphenation, make sure the solutions you are looking at offer it.
**TAG TREE**

Does the designer have a way to show the structure of the tags in the document? And clicking on one, go to that tag? There is no need for this in simple templates. But when you get to 30+ tags it becomes useful. And at 80+ it becomes essential.

If you'll always be under 50 tags, no big deal. But if you start under 50 tags and will grow to 200+ tags in a template, not having this will become a big deal. So, think about where you'll be in 5 years.

**GENERATE CODE**

This is used once and saves at most 15 minutes—but it is very nice to have. This is irrelevant for the solutions that have code behind; they create code for each template.

For the one-time code to illustrate what code is needed to add to your application to use the docgen system, it's ideal if they include a generate code feature that provides you sample code. And in addition, you know the correct way to call the engine.

Nice, not essential.

**DEBUGGERS**

Fortunately, these are rarely needed. But when needed, they can be a big timesaver. There are several different debuggers that may be in a docgen template designer add-in.

- **Template Debugger** – This is a means to step through applying data to a template to generate the document. You want the common features of breakpoints, single step, and viewing all data and state when you brake into the debugger. This helps you debug the business logic in your template.

- **Connection Debugger** – This is a means to help you find the right connection string to a datasource. (Usually SQL but it can also be a URL to XML/JSON/OData.) This will attempt the connection and if it fails, provide all exception information. It will also help with writing the connection string.

- **Query Debugger** – This is a means to determine why a select is invalid. Again, the main use here is try different selects and then see the exceptions returned. It’s a fast way to go through trial & error.

As stated above, these are rarely needed so they’re in the “useful but not important” category—except that one time you really really need it.

**DATA COUNT**

If you run a template and it takes forever, or it completes but it’s 2,00 pages long when you expected 2 pages—why? You can ask a DBA and they can track your selects and tell you the problem.

It’s faster and easier if the template add-in has a tool that tells you for each iterative select how many rows of data it returns and how long the query took to complete. From this you can quickly find what is wrong.

Useful, not essential.
TAGS

Every product has different names for the various tags. Here we use the tag names from Windward, but everyone has most of these (albeit under different names).

It's also important to look at the specific functionality of some tags. Can the import tag optionally insert a section break (Word) before/after the import? Can the forEach tag insert a section break, new workbook, and/or new slide on each iteration? Are bitmap & chart tags actual Office pictures & charts?

- **out**
  Place the data from a select at this location

- **import**
  Data returns a filename or URL, place what’s in that file at this location

- **bitmap**
  Some solutions handle bitmaps as part of the out & import tag. Others have a distinct tag for bitmaps (sometimes called pictures).

- **set**
  Set the value of a parameter (new or existing)

- **query**
  Reads one row of data to be used by other tags

- **forEach, endforEach** — Iterate through the rows of data in the query repeating the template content between them once for each row of data returned.

- **if, else, elseif, endif**
  Conditionally include content between if and else or else and endif based on the result of the query. The else is optional. Note: Windward does not have elseif.

- **switch, case, endif Switch**
  Like if but has multiple case statements within the switch.

- **link, endlink**
  The query returns a URL for the link. The link is applied to all content between the start and end tag.

- **bookmark**
  A link anchor inside the document. The name set from the data.

- **chart**
  A chart built from data. Some systems for DOCX/XLSX/PPTX output create an Office chart object (good) while some create a bitmap rendering of the chart (poor).

It’s fascinating the power of what you can create with this set of 11 tags. There really is no limit, yet it’s with a moderately sized set of constructs. The power is in what each of those do under the covers.
DOCUMENT GENERATION PRODUCTS

Here is a list of document generation software that you can embed into your applications or solutions. The template design functionality is most of what’s key for these solutions.

- **Docmosis**
  Generate documents and reports based on templates. Output in PDF/Doc/ODT from Java, PHP, C#, Ruby and more.

- **Ecrion**
  At Ecrion, we make customer communications management software for companies who want to establish genuine connections across multiple engagement channels.

- **Formstack**
  Use Formstack document generation software to merge data into custom-built documents. Document automation tools save hours of time and money.

- **HotDocs**
  No matter your industry or company size, HotDocs from AbacusNext has a solution to help speed up your document creation workflow.

- **Windward Studios**

- **XPertDoc**
  Our specialty is document generation and document automation. Our mission is to enable organizations to digitally transform their document processes.

DOCUMENT AUTOMATION PRODUCTS

Here is a list of Low Code/No Code solutions that provide an end-to-end document automation solution. This guide focused solely on the template design step of these total solutions. You need to also evaluate the additional functionality each provides.

- **Conga**
  Conga's end-to-end AI digital document transformation increases business-critical efficiencies, leads, and revenue generation. Automate for ROI...

- **Formstack**
  Use Formstack document generation software to merge data into custom-built documents. Document automation tools save hours of time and money.

- **Nintex**
  Nintex is the market leader in end-to-end process management and workflow automation. Easily manage, automate, and optimize your processes with no code.

- **Templafy**
  Templafy helps companies perfect every aspect of business document creation. Enable your employees to work faster & within company standards every time.

- **Windward Studios**
  The Global Leader in Document Automation Solutions. Revolutionize your document generation. From a comprehensive SaaS or desktop solution, to seamless integration in your CRM, we have you covered.
For over 10 years, Windward has lead the industry with our world-class document generation platform that creates visually stunning, data-powered documents designed exactly the way users want and are created in a fraction of the time and cost compared to existing solutions. Proudly located in Boulder, Colorado, Windward Studios is the premier solution for developers and business users adding reporting and document generation capabilities to their applications in over 70 countries around the world.