Technical Documentation

DOCUMOTO XML CONTENT IMPORT GUIDE

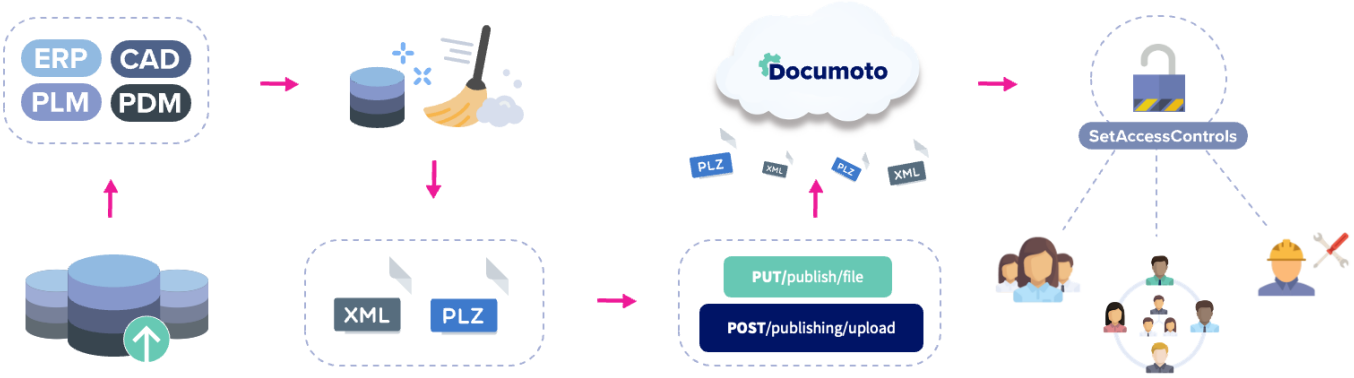
TABLE OF CONTENTS

Overview

This guide explains how to prepare, package, and import XML-formatted content into Documoto. It covers:

- Required content components
- File structure and formatting
- Packaging conventions (.PLZ, .MDZ)
- Import options (UI and API)
- Metadata and translation settings
- FAQs

Automated Publishing Workflow



Step 1.
Extract
Extract the data from your source systems

Step 2.
Cleanse & Transform
Cleanse and transform the content to match Documoto's standard XML specification

Step 3.
Load
Load the XML book and PLZ files using the following Documoto's **PUT/publishing/publish/v1/{file}** and **POST/publishing/upload/v1** APIs

Step 4.
Set Access Controls
Set who can view the content using Documoto's **PUT/authorization/media/access-controls/v1** API

Required Components for a Parts Book Import

To import a Parts Book into Documoto, each **Media (Book)** and its **Pages** must be prepared using specific file types, detailed below.

Media Table of Contents (TOC) XML

Defines the structure of the Book:

- Defines which pages are included in the book
- Specifies chapters (optional)
- The order in which the chapters (optional) and/or pages are displayed

Page Content Files

Each page requires up to three files:

File Name	File Type	File Description
Page Parts List	.XML	Structured list of parts with metadata
Page Hotpoint Mapping	.SVG	Maps hotpoints to the image
Page Diagram Image	.PNG	Diagram image of the assembly

Page Requirements & Specification

Page Parts List XML Requirements

A page .XML file:

- Must include the page's metadata and part information
- Must include a link to the page's .SVG file (via the pageFile attribute in the XML page element)
- May include <SecondaryPageFile> tags for multi-image BOMs

Page Parts List XML Specification

```
<?xml version="1.0" encoding="UTF-8"?>
<Page xmlns="http://DOCUMOTO.com/documoto/partslist/1.6" pageFile="917161J_1-1.svg"
tenantKey="DOCUMOTO">
  <Translation locale="en_US" description="" name="TIRE ASSEMBLY"/>
  <Tag name="SERIAL NUMBER" value="123456"/>
  <Part instanceId="1" item="1" partNumber="055648" quantity="2" supplierKey="DOCUMOTO"
unitOfMeasure="EA">
    <Translation locale="en_US" description="" name="TIE/WIRE"/>
  </Part>
  <Part instanceId="2" item="2" partNumber="062023" quantity="1" supplierKey="DOCUMOTO"
unitOfMeasure="EA">
    <Translation locale="en_US" description="" name="CLAMP/HOSE #M5"/>
  </Part>
  <Part instanceId="3" item="3" partNumber="064260" quantity="2" supplierKey="DOCUMOTO"
unitOfMeasure="EA">
    <Translation locale="en_US" description="" name="NUT/HEX 5/16"/>
  </Part>
  <Part instanceId="4" item="4" partNumber="067847" quantity="1" supplierKey="DOCUMOTO"
unitOfMeasure="EA">
    <Translation locale="en_US" description="" name="NUT/HEX 3/8"/>
  </Part>
</Page>
```

Page Hotpoint Mapping SVG Requirements

To enable interactive diagrams, each page must link image hotpoints to the corresponding item number in the parts list ("BOM"). This is achieved using an .SVG file, which serves as a bridge between the diagram image and the parts list. There are two supported methods to create the .SVG mapping file.

Option 1: Export SVG from CAD System

Use this option if your CAD system supports native .SVG export.

A valid CAD-generated SVG file must:

- Be directly exported from a CAD system
- Includes <g> (group) elements to define each callout location
- Does not contain raster images (e.g., embedded .PNGs or .JPEGs).

***NOTE**: If a valid .SVG file is used, you do not need to include a .PNG image in the .PLZ package.*

Option 2: Create SVG Wrapper

If a valid .SVG file cannot be generated, use a raster image (e.g. .PNG, .JPG, etc.) and create a .SVG wrapper file.

A .SVG wrapper file must:

- Include an <image> element that references the .PNG used for the diagram image
- Define the image dimensions (height, width)
- Include hotpoint data (optional – see hotpoint data structure below for more details)

Hotpoint Data Structure

The hotpoints are <g> elements with two child elements:

- **<ellipse> element**: defines the circle and hotpoint that gets overlaid on the diagram image text element – which is the text for the hotpoint
- **<text> element**: displays the item number for the hotpoint.
    - **IMPORTANT**: the value inside the <text> element must *exactly match* the item number in the parts list ("BOM"). Ensure that no leading or trailing characters, such as periods, spaces, or any extra symbols are included.

BOM Item Number	Valid <text> Values	Invalid <text> Values
1	<text>**1**</text>	<text>**1.**</text> <text>**_1**</text> <text>**1** </text?

.SVG Wrapper Example

If the part list has item numbers 1, 2, 3... 10, and the diagram has callouts that exactly match the BOM, the .SVG wrapper file should:

- Contain one (1) <image> element
- Contain ten (10) <g> elements, where each <g> element has two children: the <ellipse> for the circle, and the <text> element with the precisely matching item values for the BOM

If a diagram has no callouts, the wrapper SVG may include just the image element without any <g> groups.

Page Hotpoint Mapping XML Specification

Using the Page XML Specification provided above, below is an example of what the .SVG file might look like. *Note how the <image> element links to the page's PNG file, establishing a link between the diagram and the hotpoints.*

```
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:html="http://www.w3.org/TR/REC-html40">
    <image id="snapshot" x="0" y="0" width="1620" height="1287" xlink:href="917161J_1-1.png"/>
    <g>
        <ellipse rx="15" ry="15" fill="#4A4A4A" fill-opacity="1.0" stroke="#FFFFFF" stroke-
width="2" cx="93" cy="294"/>
        <text x="93" y="294" fill="#FFFFFF" fill-opacity="1.0" text-anchor="middle">1</text>
```

```
    </g>
     <g>
        <ellipse rx="15" ry="15" fill="#4A4A4A" fill-opacity="1.0" stroke="#FFFFFF" stroke-
width="2" cx="1016" cy="148"/>
        <text x="1016" y="148" fill="#FFFFFF" fill-opacity="1.0" text-anchor="middle">3</text>
    </g>
    <g>
        <ellipse rx="15" ry="15" fill="#4A4A4A" fill-opacity="1.0" stroke="#FFFFFF" stroke-
width="2" cx="889" cy="340"/>
        <text x="889" y="340" fill="#FFFFFF" fill-opacity="1.0" text-anchor="middle">2</text>
    </g>
    <g>
        <ellipse rx="15" ry="15" fill="#4A4A4A" fill-opacity="1.0" stroke="#FFFFFF" stroke-
width="2" cx="777" cy="310"/>
        <text x="777" y="310" fill="#FFFFFF" fill-opacity="1.0" text-anchor="middle">3</text>
    </g>
    <g>
        <ellipse rx="15" ry="15" fill="#4A4A4A" fill-opacity="1.0" stroke="#FFFFFF" stroke-
width="2" cx="1051" cy="301"/>
        <text x="1051" y="301" fill="#FFFFFF" fill-opacity="1.0" text-anchor="middle">4</text>
    </g>
    <g>
        <ellipse rx="15" ry="15" fill="#4A4A4A" fill-opacity="1.0" stroke="#FFFFFF" stroke-
width="2" cx="1106" cy="248"/>
        <text x="1106" y="248" fill="#FFFFFF" fill-opacity="1.0" text-anchor="middle">5</text>
    </g>
</svg>
```

Based on the **Page XML** and **Page Hotpoints-to-Image-to-Parts-List Mapping SVG** referenced above, the following files must be provided to Documoto:

- 917161J_1-1.xml
- 917161J_1-1.png
- 917161J_1-1.svg

Media TOC XML Requirements & Specification

Media TOC XML Requirements

The Media TOC XML is what contains a book's Table of Contents (TOC) and metadata. To correctly reference a Page within the TOC, you will need to know the Page's pageFilename and the Documoto language translations to be associated with it (via the Translation attribute and locale property). A list of valid language locale values is provided at the end of this guide.

Media TOC XML Specification

The below example will create a Media TOC XML file with one chapter that contains two pages (one of which was shown in the above example):

```
<?xml version="1.0" encoding="UTF-8"?>
<Media xmlns="http://digabit.com/documoto/book/1.2.1" identifier="ET17364A" tenantKey="DIGABIT">
    <Translation description="Parts Catalog for ET17364A" locale="en_US" name="ET17364A - 10SC32 -
- SHANXI DEYUAN FUGU ENERGY CO"/>
    <Tag name="Model" value="10SC"/>
    <Tag name="Mine Name" value="SHANXI DEYUAN FUGU ENERGY CO"/>
    <Tag name="Equipment Type" value="Shuttle Car"/>
    <Tag name="New Serial Number" value="ET17364A"/>
```

```
    <Tag name="Country" value="China"/>
    <Tag name="Region" value="China"/>
    <Chapter>
        <Translation description="" locale="en_US" name="GENERAL GROUP" />
        <Translation description="" locale="ja_JA" name="
        <Page pageFile="917161J_1-1.svg">
            <Translation description="" locale="en_US" name="ARRANGEMENT, GENERAL, MACHINE CN-
ET002927-0017" />
            <Translation description="" locale="ja_JA" name="
        </Page>
        <Page pageFile="917161J_1-2.svg">
            <Translation description="" locale="en_US" name="CAUTION STATEMENT CN-IL000046-0000" />
            <Translation description="" locale="ja_JA" name="
        </Page>
    </Chapter>
</Media>
```

Processing Instructions

Processing instructions determine whether existing values in Documoto, such as those on parts and pages, should be **replaced** or **updated** when publishing new or updated XML content. Below are the available processing instruction options you can define within your XML:

Setting	Default	Update/Add Behavior	Replace Behavior
Update or Replace Part Tags	Update	Adds **new** tags from the PLZ or Media XML to the existing parts.	Removes **all** Part tags from the PLZ or Media XML and replaces them with **only** the tags in the current file being published.
Update or Replace Page Tags	Update	Adds **new** tags to the PLZ or Media XML.	Removes all Page tags from the PLZ or Media XML and replaces them with only the tags in the current file being published.
Add or Replace Attachments	Add	Adds **new** attachments to the PLZ or Media XML.	Removes **all** attachments from the PLZ or Media XML and replaces them with **only** the ones in the current file being published.
Lock Part Translations	Update (false)	If set to false, existing Part Translations **will be updated** to match the PLZ or Media XML.	If set to true, existing Part Translations in Documoto will **not** be updated, even if different from those in the PLZ or Media XML.
Update or Replace Hotpoint Links	Replace (PLZ) Update (Media XML)	Adds new Hotpoint Links to the PLZ or Media XML.	Removes **all** Hotpoint Links and replaces them with **only** those in the current PLZ or Media XML.

Encoding and Localization

Character Encoding Requirements

All data provided must be:

- UTF-8 encoded

- XML-encoded

NOTE: Details on characters that must be XML encoded can be found here: https://webfocusinfocenter.informationbuilders.com/wfappent/TLs/TL_lang/source/xmlencod.htm

Supported Locales

Locale Name	Language Code
Arabic (Egypt)	ar_EG
Armenian	hy_AM
Bulgarian	bg_BG
Croatia	hr_HR
Czech	cs_CZ
Danish	da_DK
Deutsch	de_DE
English	en_US
español	es_ES
Estonian	et_EE
Finnish	fi_FI
français	fr_FR
Greek	el_GR
Hebrew	he_IL
Hindi	hi_IN
Hungarian	hu_HU
Icelandic	is_IS
Bahasa Indonesia	id_ID
Italian	it_IT
日本	ja_JP
한국어	ko_KR
Latvian	lv_LV
Lithuanian	lt_LT
Mongolian	mn_MN
Nederlands	nl_NL
Norwegian	nn_NO
polski	pl_PL
português	pt_PT
Romanian	ro_RO
русский	ru_RU
Serbian	sr_SP
Slovak	sk_SK

Locale Name	Language Code
Slovenian	sl_SI
svenska	sv_SE
Turkish	tr_TR
Ukrainian	uk_UA
中国	zh_CN

Packaging Content for Import

Parts List Zip (.PLZ)

A .PLZ file is the file format used by Documoto to package page data. It is a ZIP-compressed archive that contains the following components:

- Parts List (.XML)
- Hotpoint Mapping (.SVG)
- Diagram Image (.PNG)

Creating a .PLZ File

To create a .PLZ file, simply ZIP the required compnents together and update the file extension from .ZIP to .PLZ.

Example (Linux/Mac):

```
zip [name-of-plz].plz /path/to/files/*.*
```

Replace [name-of-plz] with your desired filename and ensure you include all required components in the specified path.

***NOTE**: A .PLZ file can consist solely of a Page XML file. If you do not have a diagram image or hotpoint SVG, you can skip the .PLZ packaging step and upload the XML file directly; see below for instructions.*

Media Document Zip (.MDZ)

A .MDZ file is the file format used by Documoto to package media data. It is a ZIP-compressed archive that contains the following components:

- Page Content (all referenced .PLZ files)
- Media TOC (.XML)
- Thumbnail images (optional)
- Static documents used as Related or Attachments (optional)

Creating a .MDZ File

To create a .MDZ file, simply ZIP the required components together and update the file extension from .ZIP to .MDZ.

Importing Content into Documoto

Using the Documoto Admin Center

Every Documoto customer is given access to an SFTP directory where they can upload content to Documoto. After your content has been uploaded, you will need to login to Documoto to process the content. To process your content, please follow these steps:

1. Within the Documoto Library, click ***Admin*** > ***Admin Center***

2. Click **Content**
3. Click **Process Content**
4. Select the content you wish to process
5. Click **Submit**

Using Documoto Web Services

Alternatively, you can upload and process all your content using Documoto's Web Services alone. Specifically, the following REST APIs can be used to automate the publishing and processing of your content:

1. **PUT /publishing/publish/v1/{file]**: Starts an asynchronous job to process a file that was uploaded to the Documoto website. On response, the service returns the job ID.
2. **POST /publishing/upload/v1**: Upload a file to the Documoto website, and conditionally submit the file for processing. If the file was submitted for publishing, the job ID is returned.
3. **GET /publishing/job-status/v1/{jobId}**: Gets the status of a submitted job by job ID.
4. **PUT /authorization/media/acess-controls/v1**: Updates the associated organizations or media categories for a specified media identifier

For a complete list of all Documoto Web Services, please refer to our: **Swagger Specification**.

FAQs

What are the different types of tags?

There are currently three different types of tags depending on how you want to present your metadata. There is a text field, list, and range tag. For more information on these, please visit the Documoto Knowledge Base **here**.

What is a hotpoint link?

A hotpoint link is a special hotpoint that will redirect a user to another page or parts book. To learn more about hotpoint links in your Pages, please refer to the Documoto Knowledge Base **here**. An example of the hotpoint link element, that links to a new page:

<HotpointLink item="2" locale="en_US" targetPageFile="pageToLinkTo.svg" targetPageName="Rotary group, cpl" type="Page"/>

What is a secondary page file?

Documoto supports the ability to have multiple images (or drawings) for a single BOM. An example of this might be a part list that contains 100 rows (item 1-100), but a single engineering drawing depicting all the parts would be too busy or not readable. In cases like this, the client's engineering team typically breaks the drawing into 2 or more drawings. For example, they may have 3 engineering drawings for such a large assembly, where items 1-30 are shown in the first drawing, items 31-60 are shown in the second drawing, and items 61-100 are shown in the third and final drawing. Each SVG for each drawing must be referenced in the page XML as follows:

<Page xmlns="http://DOCUMOTO.com/documoto/partslist/1.6" xmlns:xs="http://www.w3.org/2001/XMLSchema" hashKey="2606N6230L-21708" pageFile="2606N6230L.svg" tenantKey="IEC-MODEL">
<Translation description="" locale="en_US" name="BRAKE, CLUTCH AND PINION SHAFT ASSEMBLY-21708"/>
<Translation description="" locale="de_DE" name="Bremse, Kupplung und Vorgelegewelle-21708"/>
<SecondaryPageFile displayOrder="1" fileName="2606N6230L 2.svg"/>
<SecondaryPageFile displayOrder="2" fileName="2606N6230L 3.svg"/>
<SecondaryPageFile displayOrder="3" fileName="2606N6230L 4.svg"/>

What is a part code?

A part code is used to represent a part on a parts list that does not have a part number (ex. REF, PNNA, NSS, -). Part codes must be setup in Documoto in *Admin* > *Admin Center* > *Content* > *Part Codes* **prior to publishing any content.** For more information on part codes, please refer to the Documoto Knowledge Base **here**.

What makes a part unique?
Part uniqueness is the Part Number + Supplier Key. It is important to consider for large organizations where multiple regions across the globe will be using Documoto. If content across regions is different, yet the engineering teams share part numbers, an important consideration is whether or not the name of the part is the same. If it is not, should it become a single standard name, or do the parts need to retain a different name for each region? In the latter case, one recommendation is to use a different supplier key for each of the different regions.

What makes a page unique?
Page uniqueness is important to consider for all organizations using Documoto. There are two methods to determine page uniqueness within Documoto: 1). the Page's Hash Key, and 2). the Page's Page Filename + Any Translation Name for that Page. Hash keys are assigned in one of two ways:

1. Whenever a page is published to Documoto, a hash key is automatically assigned to the page, and when the page is edited via Page Builder, the hash key will be retained to ensure the same exact page is modified/updated when the page is republished.
2. For clients that have their own automated publishing, or generate their own content outside of Page Builder:
    a. They can either not set a hash key, and Documoto will generate a unique GUID for the hash key based off the page file plus page name combination, OR
    b. They can specify a hash key (unique identifier) using their TENANTKEY-[identifier] as the format for applying their own unique identifier to the page.

Furthermore, it is not possible to publish media xmls if there are multiple pages with the same page file plus page name combination. Therefore, it is important that all pages have two layers of uniqueness:

1. The Page Filename + Page Name combination
2. The hash key (whether customer-defined by automation or Documoto-assigned)

What is auto-hotpointing and how does it work?
Auto-hotpointing is a form of hotpointing that relies on Documoto algorithms to hotpoint your parts pages instead of having a person manually hotpoint your pages. Additional information on how SVG auto-hotpointing can be achieved can be found in the Documoto Knowledge Base **here**.

What is a media identifier and why do I need one?
The purpose of a media identifier is to distinguish the media to be unique from all other books and documents in Documoto; it is assigned at the time the media is created. You can think of the identifier as a unique 'key'.

What are the different types of media I can create?
There are currently six different types of Media you can create. The most common Media types are 'Book' and 'Document'. For more information on these, please visit the Documoto Knowledge Base **here**.

I have thumbnails for my books, is there a way to bulk assign them?
Yes! To bulk assign thumbnails for parts, pages, chapters, and books, please refer to the Documoto Knowledge Base **here**.