Copyright © 2019 Infor

Important Notices

The material contained in this publication (including any supplementary information) constitutes and contains confidential and proprietary information of Infor.

By gaining access to the attached, you acknowledge and agree that the material (including any modification, translation or adaptation of the material) and all copyright, trade secrets and all other right, title and interest therein, are the sole property of Infor and that you shall not gain right, title or interest in the material (including any modification, translation or adaptation of the material) by virtue of your review thereof other than the non-exclusive right to use the material solely in connection with and the furtherance of your license and use of software made available to your company from Infor pursuant to a separate agreement, the terms of which separate agreement shall govern your use of this material and all supplemental related materials ("Purpose").

In addition, by accessing the enclosed material, you acknowledge and agree that you are required to maintain such material in strict confidence and that your use of such material is limited to the Purpose described above. Although Infor has taken due care to ensure that the material included in this publication is accurate and complete, Infor cannot warrant that the information contained in this publication is complete, does not contain typographical or other errors, or will meet your specific requirements. As such, Infor does not assume and hereby disclaims all liability, consequential or otherwise, for any loss or damage to any person or entity which is caused by or relates to errors or omissions in this publication (including any supplementary information), whether such errors or omissions result from negligence, accident or any other cause.

Without limitation, U.S. export control laws and other applicable export and import laws govern your use of this material and you will neither export or re-export, directly or indirectly, this material nor any related materials or supplemental information in violation of such laws, or use such materials for any purpose prohibited by such laws.

Trademark Acknowledgements

The word and design marks set forth herein are trademarks and/or registered trademarks of Infor and/or related affiliates and subsidiaries. All rights reserved. All other company, product, trade or service names referenced may be registered trademarks or trademarks of their respective owners.

Publication Information

Release: Infor Document Management 12.0.x

Publication date: November 12, 2019
Contents

About this guide ......................................................................................................................... 5
  Contacting Infor .................................................................................................................. 5

Chapter 1 Installing the add-in on Microsoft Word ..................................................................... 6

Chapter 2 Settings .................................................................................................................. 7
  Server settings .................................................................................................................... 7
  Tool settings ....................................................................................................................... 7
  Document Properties ......................................................................................................... 9

Chapter 3 Content Source .................................................................................................... 12
  Data File ........................................................................................................................... 12
  IDM Image .......................................................................................................................... 12

Chapter 4 Property Editor .................................................................................................... 13
  Selected Control Properties ............................................................................................. 13
  Selected Control JSON ..................................................................................................... 14
  Usage Finder .................................................................................................................... 14

Chapter 5 Document Navigator ............................................................................................ 15

Chapter 6 Document Validation ............................................................................................. 16

Chapter 7 Generate Test PDF ................................................................................................. 17

Chapter 8 Google Cloud Print ............................................................................................... 18
  Creating a Google account .............................................................................................. 18
  Creating a project ............................................................................................................. 19
  Creating a service account .............................................................................................. 19
  Registering printers ......................................................................................................... 20
  Sharing a printer with a service user .............................................................................. 20
Contents

Using the Printing UI ........................................................................................................................................... 21
  Manage Printer Credentials ................................................................................................................................. 21
  Accept Printers .................................................................................................................................................... 21

Chapter 9  Document Output APIs .................................................................................................................. 22

REST endpoints ..................................................................................................................................................... 22

Creating a job ...................................................................................................................................................... 23
  input .................................................................................................................................................................... 23
    generate (Java class GenerateFile) .................................................................................................................. 24
    pid (Java Class ItemPidFile) .......................................................................................................................... 26
    xquery (Java class ItemXqueryFile) ................................................................................................................ 26
    list (Java class ItemListFile) ........................................................................................................................ 27
    template (Java class ItemTemplateFile) ......................................................................................................... 28
    assemble (Java class AssembleFile) ................................................................................................................ 28
    data (Java class DataFile) ............................................................................................................................. 30
    priority (Java class ItemPriorityFile) ............................................................................................................ 30
  target .................................................................................................................................................................. 31
    email (Java class EmailTarget) ....................................................................................................................... 32
    print (Java class PrintTarget) ....................................................................................................................... 34
    Item (Java class ItemTarget) .......................................................................................................................... 35

Submitting a Job .................................................................................................................................................. 36
About this guide

This document serves as a guide for using the IDM add-in in Microsoft Word to develop a template used in IDM Output Management.

Contacting Infor

If you have questions about Infor products, go to Infor Concierge at https://concierge.infor.com/ and create a support incident.

If we update this document after the product release, we will post the new version on the Infor Support Portal. To access documentation, select Search > Browse Documentation. We recommend that you check this portal periodically for updated documentation.

If you have comments about Infor documentation, contact documentation@infor.com.
Chapter 1  Installing the add-in on Microsoft Word

1  Go to the IDM Control Center.
2  Select Development > Downloads > Output Management Word Add-In. The DocgenEditor.zip file is downloaded.
3  Open the folder and click IDMDocGenWordAddinInstaller.msi.
4  Follow the instructions in the installer to install the add-in.
5  Launch Microsoft Word 2016 or 2016 for Windows.

The Infor Document Management tab is displayed in the top toolbar.
Chapter 2  Settings

Server settings

The server settings connect the add-in to your IDM environment.

1. Verify that the IDM Service URL is correct. The URL must have this format:
   <Infor OS login URL>:9543/ca

2. Click Verify. A “Successful Server Response” message is displayed. This is a confirmation that this URL is valid.

3. Verify that the login URL is correct. This is the URL you use to access your instance of Infor OS.

4. Sign in using your Infor OS credentials. If your login was successful, a "Login was Successful!" message is displayed.

Tool settings

Use the tool settings to set preferences for your usage of the add-in:
- **Load last used content file as content source in tool**
  Automatically load in the last XML file you used to create a template.

- **Remove XML namespaces when loading content file**
  Reformat XML data to properly fit a Word document.

- **Enable XPath element context support as a drop-down list**
  Allow XPath suggestions in the property editor.

- **Content data file**
  Choose a data XML with which to create your template.

- **Highlight document selection in content and navigation tree**
  For more information, see Document Navigator.

- **Block auto switch to Draft view mode**
  Addresses an unwanted feature in Word. We recommend that you keep this check box selected.
  Click **Set Print Layout View** to go back to print view if necessary.

- **Default Date Input Format** and **Default Time Input format**
  This is based on locale and the format of the date and time fields in the data XML you are using.

- **Image XQuery, Replacement ID**, and **Image Tool Search XQuery**
  Values that configure the IDM image search. The XQuery should yield photo files (.jpg or .png) when searched in IDM. This figure shows examples of the search settings:

  ```xml
  Image XQuery
  /MDS_File[@MDS_Name LIKE "%.text%"]
  
  Replacement Id
  $Text
  
  Image Tool Search XQuery
  /MDS_File[@MDS_Name LIKE "%.text%"]
  ```

- **Show Text/Picture/Repeating Section Control box as**
  Specify how content controls of each type are represented in the layout document.
  Allowed values:
  - **Start/End tag**:
    This screenshot shows an example:
• **Bounding box:**
  This screenshot shows an example:

  ![XML](image.png)

• **None:**
  This screenshot shows an example:

  ![XML](image.png)

**Document Properties**

This panel shows the properties of the document you are working on. Some of these properties are locked and are only displayed for information.

These fields are locked, and are filled in if this document is being edited by someone other than who created it:

- **Document Title**
  The ORIGINAL document title.

- **Document Author**
  The ORIGINAL document author.

- **Original Source Data XML**
  These fields can be edited according to the user preference for the layout:

- **Generate PDF font mode**
  This is to ensure that the fonts you used are supported on other systems.
  Allowed values:
  - **None**
    Generate the PDF with no accompanying fonts.
  - **Subset**
    Generate the PDF with the fonts that you used. This value is recommended.
- **Full**
  
  Generate the PDF with all supported fonts.

- **Output format**
  
  Follows the {"outputformat":"value":"<VALUE>"} format. 
  
  Valid values include:
  - PDF
  - PDFA

- **Paper Size**
  
  Follows the {"papersize":"value":"<VALUE>"} format. 
  
  Valid values include:
  - A3
  - A4
  - A5
  - B4
  - B5
  - EXECUTIVE
  - FOLIO
  - LEDGER
  - LEGAL
  - LETTER
  - ENVELOPE_DL
  - QUARTO
  - STATEMENT
  - TABLOID
  - PAPER_10_X_14
  - PAPER_11_X_17

- **Orientation**
  
  Follows the {"orientation":"value":"<VALUE>"} format. 
  
  Valid values include:
  - PORTRAIT
  - LANDSCAPE

- **Watermarks**
  
  Follows this format:
  
  {"watermarks":{"field1a":"VALUE1a","field1b":"VALUE1b","field1c":"VALUE1c","field1d":"VALUE1d",...}}
You can add multiple watermarks. This table shows some of the valid fields and their values:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xpath</td>
<td>XPath</td>
<td>An XPath from your content source; the only required value.</td>
</tr>
<tr>
<td>color</td>
<td>&quot;red&quot;: x, &quot;green&quot;: y, &quot;blue&quot;: z</td>
<td>Where x, y, and z are numbers from 0-255 that signify the saturations of red, green, and blue.</td>
</tr>
<tr>
<td>fontSize</td>
<td>Any positive integer</td>
<td>Specifies how big the text is.</td>
</tr>
<tr>
<td>fontFamily</td>
<td>The name of a font</td>
<td>The font of the text</td>
</tr>
<tr>
<td>textRotation</td>
<td>0-360</td>
<td>The degree to which the text is rotated</td>
</tr>
</tbody>
</table>

These are examples of valid watermark inputs:

- `{"watermarks": [{"xpath": "/M3OutDocument/Watermark/@Label"}, "color": {"red": 192, "green": 0, "blue": 0}, "fontSize": 60, "fontFamily": "Times new roman", "textRotation": -35}, {"text": "DRAFT", "textRotation": 0}]`
- `{"watermarks": [{"text": "TEST", "color": {"red": 192, "green": 0, "blue": 0}, "fontSize": 60, "fontFamily": "Times new roman", "textRotation": -35}, {"text": "DRAFT", "textRotation": 0}]`
- `{"watermarks": [{"text": "DRAFT"}]}`
Chapter 3  Content Source

Data File

Use this panel to import the data XML that is being used to create this template. After the data XML is imported into the Data File panel, the Content Source panel shows each content control that can be added into the template.

To insert a content control, right-click your desired field in the content source, and choose the format in which you want to specify the content:

- Plain Text
- Rich Text
- Image/Barcode
- Repeating section

If you specify one of the above types in the Drop As bar, you can also drag and drop the content control.

To search for content controls you require, use the top bar of the Data File panel.

Repeating sections are for tabular data that follows the same format multiple times. Create the column headers for the table in Word. Then insert a repeating section around it by highlighting the row and inserting a content control. You can specify content controls within the repeating sections for each column. This ensures that the data in the table repeats for as long as the data exists in that format.

IDM Image

You can also search for images in your IDM environment directly from the add-in. In the IDM Image panel, you can search by keyword. Ensure you are logged in correctly before doing this.

Then, to insert the picture into the template, click Insert IDM Query Picture Control. The picture is included in all files you generate from this template.
Chapter 4  Property Editor

Selected Control Properties

This panel is for viewing and editing the properties of a specific content control. To enable editing, select one content control. You can change the following:

- **Value XPath**
  The path that indicates the location in the data XML's structure where this content control was taken from.

- **IDM XQuery**
  This only applies to images that were inserted after being searched in the IDM Images panel.

- **Value type**
  The type of value (date, time, number, hyperlink) the content control is meant to hold.

- **Label XPath**
  This only applies to label content controls.

- **Custom Locale**
  This changes the date and number formatting. This only applies if you click the *Use this locale when formatting value* button.

- **Input format**
  Tells Output Management what to expect from the data XML file in terms of format.

- **Output format**
  What the data should look like when the document is generated.

- **Visibility**
  Determine when content found in this content control should be visible.
  Allowed values:
  - **Dont hide**
    This control is always visible.
  - **Hide control if value XPath result is empty**
    If the data XML does not have any values for this XPath, this control is hidden.
• **Hidden control if XPath condition result is true**

Create a condition by which the XPath is made hidden or visible through value comparisons. For example, to hide a line item if the cost is less than $1, you would add that condition to the box below this option.

**Selected Control JSON**

This panel shows the JSON expression of a specific content control. Users who are familiar with the JSON representation of content controls can change the code to edit a control.

To ensure the JSON code syntax is correct, click **Validate** at the bottom. In this panel, you can also view logs for debugging.

**Usage Finder**

Use this panel to search for where a specific content control is used on the template you are creating. You can search by the type of control, or the value, or both. Any search result you click is highlighted in the Document Navigator and in the document itself.
Chapter 5  Document Navigator

This panel shows the structure of the layout you have created, and highlights the selected elements on the document.

You can search for a specific content control, or toggle options to view repeating sections or full XPaths.
Chapter 6  Document Validation

This function checks whether all controls have valid JSON expressions, and whether all required fonts are included.

If this validation does not pass, verify the Selected Control JSON panel to find invalid JSON expressions, or change the PDF font mode.
Chapter 7  Generate Test PDF

You must be logged in to do this. Use this function to generate a PDF that merges the layout you created with the data in your data XML.
These tasks are required to use Google Cloud Print with Infor Document Management:

1. Create a Google account.
2. Create a project.
3. Create a service account.
4. Register a printer.
5. Share a printer with a service user.
6. Accept a shared printer.

There is a limit to how many print jobs a service account can handle. To ensure an error-free experience, create one service user per 500 print jobs per day.

If there are non-cloud printers included in the environment, the Google Cloud Print Service Connector must be installed on the server.

Creating a Google account

Go to gmail.com and follow the instructions to create a Google account.

One account per company is required. If the company has several tenants, the tenants can share one Google account if they use the same printers.
Creating a project

1. Log in to your newly created Google account and navigate to https://console.cloud.google.com.
2. Click the list icon in the top left corner. Select IAM & Admin > Service Accounts.
3. On the resulting dialog, click Create.
4. Complete the information on the New Project dialog to finish creating the project. Make note of your Project name and Project ID.

Creating a service account

2 Click **Create service account**.

3 Specify a **Service account name** and, optionally, a **Role**. Click **Create**.

![Create service account](image)

The created service account is displayed on your console.

4 Click the righthand menu and select **Create Key**.

5 Select **JSON** and click **Create**. A file is downloaded to your computer. Store this file securely, as it contains credentials to your service account.

![Output Management](image)

6 Make note of your Service Account ID.

### Registering printers

1 Navigate to https://www.google.com/cloudprint#printers.

2 Click either **Add a Cloud-Read Printer** or **Add a Classic Printer**, depending on which is appropriate for your situation. Follow the instructions provided.

**Note:** The printer is always associated with the Google account, rather than the service user.

### Sharing a printer with a service user

1 Navigate to https://www.google.com/cloudprint#printers.
2. Select the printer to share and select **Share**.

3. Enter the **Service Account ID**, which you saved when you created a service account.  
   
   **Note:** All service accounts are required to have access to all printers. Repeat these steps for all printers and share them with all service users.

### Using the Printing UI

To access the Printing UI in Infor Document Management, navigate to the Control Center, and then click **Administration > Configuration > Print** on the left side bar. **Google Cloud Print** should be selected to be the current printing service. Click **Manage Service** for the following actions:

### Manage Printer Credentials

1. Click **+Add**.
2. Upload the credentials file you downloaded when you created a service account.

### Accept Printers

2. Select the printer to accept and click **Details**.
3. Select **Advance Details** and copy the **Printer ID** value.
4. Enter the printer ID in the **Printer ID** field.
5. Click **Accept Printer**.

You should now be able to use Google Cloud Print to print documents.
Document Output has several REST endpoints that can be called to perform Document Output services. This includes these actions:

- Submitting a job to Document Output
- Querying the status of an ongoing job or several ongoing jobs
- Querying the status of an ongoing batch job
- Getting a list of printers

REST endpoints

These REST endpoints are available:

- **POST /ca/api/distribution/v1/submit**
  
  Submit can generate documents, retrieve documents from IDM, merge documents, and distribute the output to different targets. Example of targets are email, print, or stored back to IDM.

  Submit is an asynchronous method that puts the job in a queue. The job is picked up as soon as possible.

- **POST /ca/api/distribution/v1/retrieve/stream**
  
  You can use retrieve to generate one document or retrieve a document and return it to the client.

  Data is sent in json structure

- **GET /ca/api/distribution/v1/status**
  
  You can use status to query the status of an ongoing job.

  The status of the job is stored for two days until the job is archived.

- **GET /ca/api/distribution/v1/batchStatus**
  
  You can use batchStatus to query the status of several ongoing jobs with the same batchId set.
The status of the job is stored for two days until the job is archived. After the job is archived, the data remains for 14 more days until it is deleted. After the data is deleted, it remains for six more days until it is purged.

- GET /ca/api/distribution/v1/printer/search

  This method returns a list of all printers. If a printer name is specified, the method returns a subset of printers.

Creating a job

A job is the object that is being sent to the submit method. It is defined as one array of input files, one array of targets, and an optional batchId.

This code shows the JSON format:

```json
{
    "input": [],
    "targets": [],
    "batchId": ""
}
```

The input array specifies all inputs, that is, files to be generated or existing files, to the job.

The target array specifies all targets to which the documents that are described in input are distributed. All input documents are applied to all targets.

The batchId is an optional parameter that groups several jobs. When specifying batchId, you can use the batchStatus endpoint. It is extremely important to use the batchId parameter when you perform a batch job that creates many jobs.

input

The input array can contain a number of different file objects that are used for different purposes. If you create the Json manually, then the first attribute in a file object must be "type" and must have one of these values:

- **generate**
  
  Generates a new document.

- **pid**
  
  Retrieves a document from IDM for a given pid.

- **xquery**
Retrieves the first document that is found from the given xquery.

- **list**
  Retrieves a list of documents that are found from the given xquery.

- **template**
  Retrieves a document from IDM given a document type and a template name.
  **Note:** Template is an IDM feature on how to specify a specific document and is not related to a Word layout.

- **assemble**
  Merge documents into one document.

- **data**
  A raw document that is sent from the client, base64-encoded.

- **priority**
  A list of file objects that is iterated until one document is found.

Each file type is described in detail below.

### generate (Java class GenerateFile)

Generate generates a document given a template layout and data.

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Value</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>&quot;generate&quot;</td>
<td>string</td>
<td>The file object type.</td>
</tr>
<tr>
<td>data</td>
<td>file object</td>
<td></td>
<td>A file object of type pid, xquery, template or data. The actual document must be an XML document. The normal scenario would be an XML produced by the client and sent in as a data object.</td>
</tr>
<tr>
<td>template</td>
<td>file object</td>
<td></td>
<td>A file object of type pid, xquery, template or data. The actual document must be a Word template. The normal scenario would be a template or xquery. It is discouraged to send the Word layout as a data object because unnecessary data is transferred to Document Output. Performance is affected.</td>
</tr>
</tbody>
</table>
This is an example of this input type in Json:

```
{
  "type": "generate",
  "data": {
    "type": "data",
    "filename": "OrderData_2342345.xml",
    "base64": "dGVzdA=="
  },
  "xquery": {
    "type": "xquery",
    "xquery": "/OutputManagementTesting[@RESOURCENAME = "/"OrderConfirmation.docx"]"
  },
  "filename": "OrderConfirmation_2342345.pdf",
```

Name of generated file. This value is used if you are sending the document as an attachment, saving it back to IDM, or referencing it in any other way. The extension of the name determines what type of document is produced. Currently .pdf, .docx, and .html are the only supported document types. Note that HTML documents cannot be stored back to IDM. HTML is mainly used for generating an email body.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filename</td>
<td>Name of generated file. This value is used if you are sending the document as an attachment, saving it back to IDM, or referencing it in any other way. The extension of the name determines what type of document is produced. Currently .pdf, .docx, and .html are the only supported document types. Note that HTML documents cannot be stored back to IDM. HTML is mainly used for generating an email body.</td>
</tr>
<tr>
<td>pdfA</td>
<td>Indicates whether to generate the pdf as a pdf/A. If the IDM_OUTPUTFORMAT custom property is specified in the document, then this property is ignored.</td>
</tr>
<tr>
<td>item</td>
<td>If there is an item target defined, then you can override the item defined in the item target with this value. If no item target is defined, then this value is ignored. Use this field if you must generate several documents in one job and save them to IDM with different attributes.</td>
</tr>
</tbody>
</table>
Document Output APIs

"item": {
  "type": "data",
  "base64": "sdfdsf",
  "filename": "MyItem.xml"
}
}

pid (Java Class ItemPidFile)

Retrieves a file from IDM given a pid. Note that a pid is only valid for one tenant or one on-premises installation. When you migrate data or clone a tenant, the pid changes.

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Value</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>&quot;pid&quot;</td>
<td>string</td>
<td>The file object type</td>
</tr>
<tr>
<td>pid</td>
<td>Example: &quot;MDS_File-27-1-LATEST&quot;</td>
<td>string</td>
<td>The pid in IDM</td>
</tr>
</tbody>
</table>

This is an example of this input type in JSON:

```json
{
  "type": "pid",
  "pid": "MDS_File-27-1-LATEST"
}
```

xquery (Java class ItemXqueryFile)

Retrieves a file from IDM given a XQuery. If the XQuery returns more than one document, then the first document that is returned is used.

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Value</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>&quot;xquery&quot;</td>
<td>string</td>
<td>The file object type</td>
</tr>
<tr>
<td>xquery</td>
<td>Example: &quot;/MDS_File[@MDS_name=FileName]&quot;</td>
<td>string</td>
<td>The xquery.</td>
</tr>
</tbody>
</table>

This is an example of this input type in JSON:

```json
{
  "type": "xquery",
  "xquery": "/MDS_File[@MDS_name=FileName]"
}
```
list (Java class ItemListFile)

Retrieves a list of files from IDM given an xquery. Use this file type if several documents that have the same attributes must be emailed or printed.

Restrictions: An xquery can result in millions of documents returned. Document Output only includes documents to a total size of 10 MB.

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Value</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>&quot;xquery&quot;</td>
<td>string</td>
<td>The file object type.</td>
</tr>
<tr>
<td>xquery</td>
<td>Example: &quot;/MDS_File[@MDS_name=&quot;FileName&quot;]&quot;</td>
<td>string</td>
<td>The xquery.</td>
</tr>
<tr>
<td>merge</td>
<td>true/false</td>
<td>boolean</td>
<td>If set to true, all the files are assembled into one PDF. If set to true, you must also set the filename property.</td>
</tr>
<tr>
<td>filename</td>
<td>Example: &quot;MyAssembledFile.pdf&quot;</td>
<td>string</td>
<td>The name of the assembled output file. Ignored if merge=false</td>
</tr>
</tbody>
</table>

This is an example of this input type in JSON:

```
{
    "type": "list",
    "xquery": "/MDS_GenericDocument["MyAttribute="someValue"]",
    "merge":true,
    "filename": "MyAssembledFile.pdf"
}
```
template (Java class ItemTemplateFile)

Retrieves a file from IDM given a template entity name and template name. Using the template functionality when referencing a document in IDM is more convenient than writing an XQuery. Note that the template functionality in IDM is not directly related to layouts.

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Value</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>&quot;template&quot;</td>
<td>string</td>
<td>The file object type</td>
</tr>
<tr>
<td>templateEntityName</td>
<td>Example: MyDocumentType</td>
<td>string</td>
<td>Document type</td>
</tr>
<tr>
<td>templateName</td>
<td>Example: MyTemplateName</td>
<td>string</td>
<td>Template name</td>
</tr>
</tbody>
</table>

This is an example of this input type in JSON:

```json
{
    "type": "template",
    "templateEntityName": "MDS_File",
    "templateName": "MyTemplateName"
}
```

assemble (Java class AssembleFile)

The assemble file object merges several files into one file. In the array of files to be merged, you can merge any of the described file types: generate, pid, xquery, template, assemble, or data.

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Value</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>&quot;assemble&quot;</td>
<td>string</td>
<td>The file object type.</td>
</tr>
<tr>
<td>files</td>
<td>file object array</td>
<td>file object array</td>
<td>An array of file objects. The files are merged into one single pdf file, in the order they are specified. The assemble file type is not allowed.</td>
</tr>
<tr>
<td>filename</td>
<td>Example: &quot;MyAssembledFile.pdf&quot;</td>
<td>string</td>
<td>Name of the file to be generated. This value is used if you send the document as an attachment or save it back to IDM.</td>
</tr>
<tr>
<td>Attribute name</td>
<td>Value</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>-------</td>
<td>---------------</td>
<td>-------------</td>
</tr>
<tr>
<td>pdfA</td>
<td>Optional&lt;br&gt;True / False&lt;br&gt;Ignored if file extension differs from .pdf.&lt;br&gt;Default value is False.</td>
<td>boolean</td>
<td>Indicates whether to generate the pdf as a pdf/A.</td>
</tr>
<tr>
<td>item</td>
<td>Optional&lt;br&gt;File object containing a data file object</td>
<td>data file object</td>
<td>If there is an item target defined, then you can override the item defined in the item target with this value. If no item target is defined, then this value is ignored. For an example, see the generate object.</td>
</tr>
</tbody>
</table>

This is an example of this input type in JSON:

```json
{
    "type": "assemble",
    "files": [
        {
            "type": "generate",
            "data": {
                "type": "data",
                "filename": "mydata.xml",
                "base64": "dGVzdA=="
            },
            "pid": {
                "type": "pid",
                "pid": "MDS_File-27-1-LATEST"
            },
            "filename": "MyGeneratedFile.pdf",
        },
        {
            "type": "xquery",
            "xquery": "/MDS_File[@MDS_name="FileName"]"
        }
    ]
}
```
data (Java class DataFile)

A complete file sent from the client that is base64-encoded. Use this option for documents or data that is not stored in IDM. We recommend that you store a file in IDM if possible and then retrieve it with one of these file types: pid, xquery, or template.

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Value</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>&quot;data&quot;</td>
<td>string</td>
<td>The file object type.</td>
</tr>
<tr>
<td>base64</td>
<td></td>
<td>string</td>
<td>Base 64 encoded document.</td>
</tr>
<tr>
<td>filename</td>
<td>Example:&quot;Invoice_234525.xml&quot;</td>
<td>string</td>
<td>The name of the file or data.</td>
</tr>
<tr>
<td>item</td>
<td>Optional file object containing a data file object</td>
<td>data file object</td>
<td>If there is an item target defined, then you can override the item defined in the item target with this value. If no item target is defined, then this value is ignored.</td>
</tr>
</tbody>
</table>

This is an example of this input type in Json:

```
{
  "type": "data",
  "base64": "dGVzdA==",
  "filename": "MyData.xml",
}
```

priority (Java class ItemPriorityFile)

A list of file objects that should be searched in order. Valid file objects are pid, xquery, or template. You can use this file type if you must first search for one document and, if that is not found, then try with the next document.
<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Value</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>&quot;priority&quot;</td>
<td>string</td>
<td>The file object type.</td>
</tr>
<tr>
<td>priority</td>
<td>Example: File object array</td>
<td>An array of file objects. Valid file object types are pid, xquery, and template. The priority list is iterated in order and when a document is found, the iteration is stopped.</td>
<td></td>
</tr>
</tbody>
</table>

This is an example of this input type in JSON:

```
{
    "type": "priority",
    {
        "type": "xquery",
        "xquery": "/CustomerModifiedDocumentType[@MDS_name="Invoice.docx"]"
    },
    {
        "type": "xquery",
        "xquery": "/DefaultDocumentType[@MDS_name="Invoice.docx"]"
    }
}
```

target

The target array specifies the targets to which the job that is being created will be submitted. If you create the JSON manually, then it is mandatory that the first attribute in a target object is “type” and has one of these values:

- email
  Sends all documents through email.
- print
  Prints all documents.
- item
Store generated and assembled files back to IDM.
Each target type is described in detail below.

email (Java class EmailTarget)
The email target sends an email with all the files from input as attachments or, if the total size of all documents exceeds 25 MB, as links.

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Value</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>&quot;email&quot;</td>
<td>string</td>
<td>The target object type.</td>
</tr>
<tr>
<td>to</td>
<td>Example: <a href="mailto:jane.doe@company.com">jane.doe@company.com</a></td>
<td>string</td>
<td>A semicolon-separated list of email addresses to send email to.</td>
</tr>
<tr>
<td>cc</td>
<td>Optional Example: <a href="mailto:jane.doe2@company.com">jane.doe2@company.com</a></td>
<td>string</td>
<td>A semicolon-separated list of email addresses to send email to as cc.</td>
</tr>
<tr>
<td>bcc</td>
<td>Optional Example: <a href="mailto:jane.doe3@company.com">jane.doe3@company.com</a></td>
<td>string</td>
<td>A semicolon-separated list of email addresses to send email to as bcc.</td>
</tr>
<tr>
<td>from</td>
<td>Optional Example: <a href="mailto:john.doe@company.com">john.doe@company.com</a></td>
<td>string</td>
<td>Override the sender email. On an IDM installation level (on-premises), you can configure whether it is allowed to change the from email address. If that configuration is done, then this attribute is ignored.</td>
</tr>
<tr>
<td>fromDisplayName</td>
<td>Optional Example: John Doe</td>
<td>String</td>
<td>Override the sender display name. This attribute is ignored if the from attribute is not specified or if IDM has been configured not allowing to change the sender. If fromDisplayName is omitted and the from attribute is specified, then the from attribute is copied to fromDisplayName.</td>
</tr>
<tr>
<td>subject</td>
<td>Example: &quot;Invoice&quot;</td>
<td>string</td>
<td>Subject of the email.</td>
</tr>
<tr>
<td>Attribute name</td>
<td>Value</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>-------</td>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>body</td>
<td>Optional Example: &quot;Please pay at once!&quot;</td>
<td>string</td>
<td>The body of the email.</td>
</tr>
<tr>
<td>attachments</td>
<td>Optional</td>
<td>array of File objects</td>
<td>An array of additional attachments that shall only be included in this email. These are allowed File objects: • pid • xquery • template • data</td>
</tr>
<tr>
<td>bodyFileName</td>
<td>Optional Example: MyGeneratedFile.html</td>
<td>string</td>
<td>To set the email body from a generated document, use this property to specify a generated file. The file extension must be .html. Read more about how to generate and send email here. Available from 12.0.24.</td>
</tr>
<tr>
<td>Include</td>
<td>Optional Example: [&quot;MyGeneratedFile.pdf&quot;, &quot;anotherfile.pdf&quot;]</td>
<td>array of string</td>
<td>List of file names to be included to the target. In this case, the files are attached to the email. If include is omitted or null, then all generated files are attached to the email. If some of the generated documents should not be included to the target, then the include list must only contain the names of the generated documents that should be attached. If no files are to be attached, you must provide an empty list. HTML documents are never included as an attachment. Available from 12.0.24.</td>
</tr>
</tbody>
</table>

This is an example of this target type in JSon:

```json
{
    "type": "email",
    ...
}
```
"to": "jane.doe@company.com",
"cc": "john.smith@company.com",
"bcc": "secret.user@anothercompany.com",
"from": "donald.duck@anothercompany.com",
"subject": "Please read",
"body": "Please read carefully",
"attachments": [
{
  "type": "xqyery",
  "pid": "/OutputManagementTesting[@RESOURCENAME = "law-enforcement-guidelines-us.pdf"]"
}
]
}

print (Java class PrintTarget)

Prints documents to the specified printer. The ID of the printer is the Google Cloud Print ID of the printer.

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Value</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>&quot;print&quot;</td>
<td>string</td>
<td>The target object type.</td>
</tr>
<tr>
<td>printerId</td>
<td>Example: &quot;sdsdf324sdf234&quot;</td>
<td>String</td>
<td>The printer ID that belongs to the printer solution configured.</td>
</tr>
<tr>
<td>noOfCopies</td>
<td>Optional Example: 2</td>
<td>int</td>
<td>Number of copies to be printed. Default value is 1.</td>
</tr>
<tr>
<td>include</td>
<td>Optional Example:</td>
<td>array of</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[&quot;MyGeneratedFile.pdf&quot;,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;anotherfile.pdf&quot;]</td>
<td>string</td>
<td>List of file names to be included to the target. In this case, the files are printed. If include is omitted or null, then all generated files are printed. If some of the generated documents should not be included to the target, then the include list must only contain the names of</td>
</tr>
</tbody>
</table>
This is an example of this target type in JSON:

```json
{
    "type": "print",
    "printerId": "sdsdf324sdf234",
    "noOfCopies": 2
}
```

**Item (Java class ItemTarget)**

Stores documents in IDM. Files of type `pid`, `xquery`, and `template` are ignored and are not stored back to IDM.

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Value</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>&quot;item&quot;</td>
<td>string</td>
<td>The target object type.</td>
</tr>
<tr>
<td>itemdatafile</td>
<td>DataFile object</td>
<td>Data file object containing a base64-encoded item xml.</td>
<td></td>
</tr>
<tr>
<td>include</td>
<td>array of string</td>
<td>List of file names to be included to the target. In this case, the files are stored in IDM. If <code>include</code> is omitted or null, then all generated files are stored. If some of the generated documents should not be included to the target, then the include list must only contain the names of the generated documents that should be added to IDM. Note that HTML files cannot be stored in IDM. If any HTML document has been generated, then use this attribute to specify only those files that should be stored in IDM.</td>
<td></td>
</tr>
</tbody>
</table>
This is an example of this target type in JSON:

```json
{
    "type": "item",
    "itemdatafile": {
        "type": "data",
        "base64": "sdfdsf",
        "filename": "MyItem.xml"
    }
}
```

**Submitting a Job**

When a job has been created, you can submit it to the submit REST endpoint. After the job is submitted, a validation of the input data is performed and the job is put in the queue.

You can use the IDM Java SDK to submit the job. If you are not using the IDM Java SDK, POST the Job JSON to the submit endpoint.

This code shows a successful response from the /submit endpoint:

```json
{
    "success": true,
    "jobId": "sadf3245dsfg435"
}
```

This code shows the response if something goes wrong during the submission:

```json
{
    "success": false,
    "errorMessage": "Something went wrong"
}
```