Configuring

Creating an associated properties file

You can create an associated properties file to specify one or more properties that are associated with a job, but that are properties of other objects. When a step based on the StoreInRepository step template runs, these properties and their values are stored in a repository along with job and document data. For example, you can store the model of the printer requested for a job or the color of the media specified to print a job.

Note:
- You can also specify positional job properties in an associated properties file and store their values in a repository. You cannot select a positional job property as a value for the Job properties to store property on the StoreInRepository step. In a workflow, the values of positional properties can be different for different steps based on the same step template.

Positional job properties

To store values for a positional job property, you specify the property, the phase that the step is in, the internal name of the step with the property, and a property label. The syntax is:

Job_property[*Phase*][*Step_identifier*]:*Property_label*

For example, you have the Automated Verification feature, and you want to store the name of the barcode reader that the ReadBarcodeData step in the Insert phase uses to track the documents in a job through an inserter.

When you create the associated properties file, you type this line into a text editor:

Job.TrackAndTrace.BarcodeReader[Insert][ReadBarcodeData]:*Property_label*

The property label might be Job.BarcodeReader.

When the StoreInRepository step runs, RICOH ProcessDirector:

1. Gets the value of the Barcode reader job property (database name Job.TrackAndTrace.BarcodeReader) for the ReadBarcodeData step in the Insert phase.
   This value might be BarcodeReader1.
2. Stores the value of the Barcode reader property along with other information for the job and its documents in the repository.

To see whether a property on a step template is positional, click the ? icon and check the Usage notes in the help.

Properties associated with a job

To store the value for a property of another object, you must be able to create a chain of relationships to that property. The chain must start with a job property that specifies an object as a value. The next property in the chain must be a property of the object specified by the job property. The chain must end with the property whose values you want to store.

Although you can start the chain with any job property that specifies an object as a value, these job properties satisfy most needs.

Object	User interface name of job property	Database name of job property	Base product or feature
Barcode reader	Barcode reader	Job.TrackAndTrace.BarcodeReader	Automated Verification
Input device	None	Job.SourceInputDeviceName	Base product
Inserter	Inserter controller	Job.InserterSystem.ID	Inserter
Media	Media	Job.Media	Base product
Printer	Requested printer	Job.RequestedPrinter	Base product

The following line shows a simple version of the syntax for specifying associated properties:

Property_to_store@*Job_property*:*Property_label*

> **Note:**
> - If you chose **Any printer** on the **PrintJobs** step, you cannot use the Job.RequestedPrinter database name. Replace that name with Job.PreviousPrinter.

The system starts at the colon and reads the properties from right to left. The number of properties in the chain can vary. An @ symbol separates the properties. To the right of the colon is a property label. The property label is required.

These examples give the user interface names of properties with the database names in parentheses. Use the database names when you create the associated properties file.

- You want to store values for the **Printer model** printer property (database name **Printer.Model.Specific**). You can chain the **Requested printer** job property (database name **Job.RequestedPrinter**) directly to the printer property.
  When you create the associated properties file, you type this line into a text editor:
  `Printer.Model.Specific@Job.RequestedPrinter:`*`Property_label`*
  The property label might be `Job.PrinterModel`.
  When the **StoreInRepository** step runs, RICOH ProcessDirector:
  1. Gets the value of the **Requested printer** property.
     This value might be **Printer4**.
  2. Uses the **Printer.** portion of the **Printer.Model.Specific** property to identify the next object in the chain: a printer object.
  3. Gets the value of the **Printer model** property for **Printer4**.
     This value might be **Ricoh Pro C901**.
  4. Stores the value of the **Printer model** property along with other information for the job and its documents in the repository.

  > **Important:**
  > - The link between the job property and the object portion of the next property is critical. You must link the **Requested printer** job property to a printer property. The database name of a printer property starts with **Printer**. An example at the end of this topic shows how to link a job property to another property through an intermediate property.

- You have the Automated Verification feature, and you want to store values for the **Barcode Format** property (database name **BarcodeReader.BarcodeFormat**). You can chain the **Barcode reader** job property (database name **Job.TrackAndTrace.BarcodeReader**) directly to the **Barcode Format** property. Because the **Barcode reader** property is positional, you need to specify the Phase and Step identifier.
  You have two different steps that read barcodes, and the barcode readers in the two steps use a different barcode format. You want the barcode reader that the **ReadBarcodeData** step in the Insert phase uses.
  When you create the associated properties file, you type this line into a text editor:
  `BarcodeReader.BarcodeFormat@Job.TrackAndTrace.BarcodeReader[Insert][ReadBarcodeData]:`*`Property_label`*
  The property label might be `Job.BarcodeFormat`.
  When the **StoreInRepository** step runs, RICOH ProcessDirector:
  1. Gets the value of the **Barcode reader** property for the **ReadBarcodeData** step in the **Insert** phase.
     This value might be **BarcodeReader2**.
  2. Uses the **BarcodeReader.** portion of the **BarcodeReader.BarcodeFormat** property to identify the next object in the chain: a barcode reader object.
  3. Gets the value of the **Barcode Format** property for **BarcodeReader2**.
     This value might be **BarcodeFormat2**.
  4. Stores the value of the **Barcode Format** property along with other information for the job and its documents in the repository.

To create an associated properties file:

1. With a text editor, create a new file.
2. Type a line for the first property whose values you want to store.
   Use this syntax:
   `Property_to_store@Intermediate_property@Job_property[Phase][Step_identifier]:Property_label`
   where:
   - *Property_to_store* is the database name of the property you want to store.
   - *Intermediate_property* is the database name of an intermediate property, if needed, that links the job property to the property you want to store by identifying an intermediate object, such as Media. If you need to specify two intermediate properties, separate them with an @ symbol.
     You can link many job properties directly to properties you want to store without an *Intermediate_property*.
   - *Job_property*[*Phase*][*Step_identifier*] has these parts:
     - *Job_property* is the database name of the job property that identifies an object such as a printer.
       If you are storing a positional job property, which you cannot select as a value for the **Job properties to store** property on the **StoreInRepository** step, *Job_property* is the database name of the positional job property. You do not need to specify any additional properties.
     - If the property is positional, *Phase* is the name of the phase that the step is in, and *Step_identifier* is the internal name of the step with the property.
       If the property is not positional, do not type a [*Phase*] or [*Step_identifier*].
   - *Property_label* is the name that appears on the Properties tab when you click **Show details** on the Results table of the Archive tab. We recommend the format Job.*MyProperty*. The property label for each property in the associated properties file must be unique.

   For example, you might type:
   `Job.TrackAndTrace.BarcodeReader[Insert][ReadBarcodeData]:Job.BarcodeReader`

3. If you want to store values for a second property, type a line break, and then repeat the previous step for the second property.

   For example, you might type:
   `Printer.Model.Specific@Job.RequestedPrinter:Job.PrinterModel`
4. Save the text file.
   For example, you might name the file `associatedproperties.txt`.
5. Send the associated properties file to the RICOH ProcessDirector server in a directory that the RICOH ProcessDirector system user has access to.

The file is now available to use as the value of the **Associated properties file** property for a **StoreInRepository** step in a workflow.

When the **StoreInRepository** step runs, RICOH ProcessDirector stores (with each document and job) the value of each stored property specified in the associated properties file.

- If the value is null for any stored property on a line of the associated properties file, RICOH ProcessDirector stores a null value for the property.
- If a property allows multiple selections, RICOH ProcessDirector stores the multiple selections separated by a vertical bar (|). For example, the value of a stored property might be **BarcodeReader1|BarcodeReader2**.
- If multiple values are selected for both a job property and the associated object property, RICOH ProcessDirector adds an underscore and the name of the job property value to the property label. RICOH ProcessDirector stores each job property value separately. For example:
  `Job.BarcodeFormat_BarcodeReader1 Job.BarcodeFormat_BarcodeReader2`
  `BarcodeFormat1|BarcodeFormat2 BarcodeFormat3`

Users cannot search a repository for these properties. After you search on the Archive tab for job or document properties, RICOH ProcessDirector displays the values of associated properties on the Properties tab when you click **Show details** on the Results table.

Examples

Storing the model of the printer requested to print a job

- File contents:
  `Printer.Model.Specific@Job.RequestedPrinter:Job.PrinterModel`
- Value of **Printer model** property:
  `Ricoh Pro C901`
- Information stored in the repository for a specific job and each document in the job:
  `Job.PrinterModelRicoh Pro C901`
- Information displayed in the properties notebook for the results of a search:
  Job.PrinterModel: Ricoh Pro C901

> **Note:**
> - When processing an associated properties file, RICOH ProcessDirector does not store or display the value of the *Job_property_identifying_object* property or any *Property_identifying_intermediate_object* property.
> - If you chose **Any printer** on the **PrintJobs** step, you cannot use the Job.RequestedPrinter database name. Replace that name with Job.PreviousPrinter.

Storing the barcode format used by a barcode reader

- File contents:
  `BarcodeReader.BarcodeFormat@Job.TrackAndTrace.BarcodeReader[Insert]`
  `[ReadBarcodeData]:Job.BarcodeFormat`
- Barcode readers selected on the **ReadBarcodeData** step:
  - **BarcodeReader1**
  - **BarcodeReader2**
- Barcode formats selected for **BarcodeReader1**:
  - **BarcodeFormat1**
  - **BarcodeFormat2**
- Barcode format selected for **BarcodeReader2**: **BarcodeFormat3**.
- Information stored in the repository for a specific job and each document in the job:
  `Job.BarcodeFormat_BarcodeReader1 Job.BarcodeFormat_BarcodeReader2`
  `BarcodeFormat1|BarcodeFormat2 BarcodeFormat3`
- Information displayed in the properties notebook for the results of a search:
  Job.BarcodeFormat_BarcodeReader1: BarcodeFormat1|BarcodeFormat2
  Job.BarcodeFormat_BarcodeReader2: BarcodeFormat3

Storing the reprint method used by an inserter

- File contents:
  `InserterSystem.ReprintMethod@Job.InserterSystem.ID:Job.InserterReprintMethod`
- Value of **Reprint method** property (database name **InserterSystem.ReprintMethod**):
  `Open loop`
- Information stored in the repository for a specific job and each document in the job:

```
Job.InserterReprintMethod
Open loop
```

- Information displayed in the properties notebook for the results of a search:
  Job.InserterReprintMethod: Open loop

Storing the folder location of the input device that received a job

- File contents:
  `InputDevice.FolderLocation@Job.SourceInputDeviceName:Job.InputDeviceFolder`
- Value of **Folder location** property (database name **InputDevice.FolderLocation**):
  `/aiw/aiw1/System/hf/defaultPDF`
- Information stored in the repository for a specific job and each document in the job:
  `Job.InputDeviceFolder`
  `/aiw/aiw1/System/hf/defaultPDF`
- Information displayed in the properties notebook for the results of a search:
  Job.InputDeviceFolder: /aiw/aiw1/System/hf/defaultPDF

Storing the color of the media requested to print a job

- File contents:
  `MediaType.Color@Media.MediaTypeID@Job.Media:Job.MediaColor`
- Value of **Media color** property (database name **MediaType.Color**):
  `Ricoh Pro C901`
- Information stored in the repository for a specific job and each document in the job:
  `Job.MediaColor`
  `Blue`
- Information displayed in the properties notebook for the results of a search:
  Job.MediaColor: Blue

> **Note:**
> - You can replace the **Media color** property with other media type properties such as **Media weight** (database name **MediaType.Weight**) and **Media details** (database name **MediaType.Details**).
>   To store the values of a media size property, such as **Media height** (database name **MediaSize.Height**), use this line:`MediaSize.Height@Media.MediaSizeID@Job.Media:Job.MediaHeight`

Parent topic: Setting up the Archive feature