Usage scenario for processing orders in an XML file

In this scenario, a printing company wants to process orders in XML files received from web-to-print sellers. Each XML file contains multiple orders, and each order can include both print and promotional items, such as coffee mugs and baseball caps.

In this scenario, the RICOH ProcessDirector administrator uses a third-party tool (such as Altova MapForce) to create Extensible Stylesheet Language Transformations (XSLT) style sheets. For examples of XML input files, XPath expressions, XSLT style sheets, and RICOH ProcessDirector overrides files that work with this scenario, see the related reference.

Reviewing and preparing to process the XML

The RICOH ProcessDirector administrator reviews the contents of the XML files and decides how to process them.

1. First the administrator identifies the XML elements that supply values for RICOH ProcessDirector job properties.
   In the example XML input file, two XML elements supply information required to track each item in an order. The `number` attribute of the `order` element and the `customername` element supply the order number and customer name.
   For print items, the `printfile` element supplies the URL of the PDF file to download and print.
   For promotional items, the `number` attribute of the `stock` element supplies the location of the item in the warehouse.
   For all items, the `quantity` element supplies the quantity ordered.
   This table shows these five XML elements and the names of the corresponding RICOH ProcessDirector job properties.

XML element	Database name of job property	User interface name of job property
`<order number>`	`Job.Info.Attr1`	Custom 1
`<customername>`	`Job.CustomerName`	Customer name
`<printfile>`	`Job.DownloadFile`	URL for download file
`<stock number>`	`Job.Info.Attr2`	Custom 2
`<quantity>`	`Job.Copies`	Job copies requested

2. The administrator breaks the process into five parts:
   - Create a separate XML job for each order.
   - Assign the order values (`order number` and `customername`) to job properties.
   - From each order, create a separate XML job for each item.
   - Assign the item values (`printfile`, `stock number`, and `quantity`) to job properties.
   - Process print items and stock items in separate workflows.
3. The administrator decides to define four workflows:
   - The **ExtractOrdersFromXML** workflow receives orders and makes separate jobs for each order.
   - The **SplitOrderIntoPrintAndStockJobs** workflow separates the items in each order into print and stock jobs.
   - The **ProcessPrintJobs** workflow processes print jobs.
   - The **ProcessStockJobs** workflow processes stock jobs.

   A step based on the **CreateJobsFromXML** step template creates jobs from elements in an XML file that match an XPath expression. The step submits the jobs to a workflow.
4. The administrator decides to use three **CreateJobsFromXML** steps, each with a different XML Path Language (XPath) expression:
   - The first step creates a separate XML job for each order.
     The Order Jobs expression identifies the XML elements that match orders: `/seller/order`
   - The second step creates a separate XML job for each print item.
     The Print Jobs expression identifies the XML elements that match print items: `/order/item/printfile/ancestor::item`
   - The third step creates a separate XML job for each promotional item.
     The Stock Jobs expression identifies the XML elements that match promotional items: `/order/item/stock/ancestor::item`

   The administrator assigns the **CreateJobsFromXML** steps to workflows:
   - The first step goes in the **ExtractOrdersFromXML** workflow. The step submits the XML jobs for each order to the **SplitOrderIntoPrintAndStockJobs** workflow.
   - The second and third steps go in the **SplitOrderIntoPrintAndStockJobs** workflow. The second step submits XML jobs for print items to the **ProcessPrintJobs** workflow. The third step submits XML jobs for promotional items to the **ProcessStockJobs** workflow.

   A step based on the **ApplyXSLTransform** step template can transform XML into a RICOH ProcessDirector overrides file that specifies the values of job properties.
5. The administrator decides to use three **ApplyXSLTransform** steps. Each step uses an XSLT style sheet. The administrator uses a third-party XSLT tool to create the XSLT style sheets.

- The first step uses the Order Jobs XSLT style sheet. It converts the `name` attribute of the `order` element and the `customername` element into the **Job.Info.Attr1** and **Job.CustomerName** job properties.
  The step goes in the **SplitOrderIntoPrintAndStockJobs** workflow before the two **CreateJobsFromXML** steps.
- The second step uses the Print Jobs XSLT style sheet. It converts the `printfile` element and the `quantity` element into the **Job.DownloadFile** and **Job.Copies** job properties.
  The step goes in the **ProcessPrintJobs** workflow.
- The third step uses the Stock Jobs XSLT style sheet. It converts the `number` attribute of the `stock` element and the `quantity` element into the **Job.Info.Attr2** and **Job.Copies** job properties.
  The step goes in the **ProcessStockJobs** workflow.

6. The administrator decides to use a **DownloadFile** step to download PDF files from a specified URL.
   Because the printing company uses a proxy server to communicate with external websites, the administrator sets up RICOH ProcessDirector to use the proxy server.

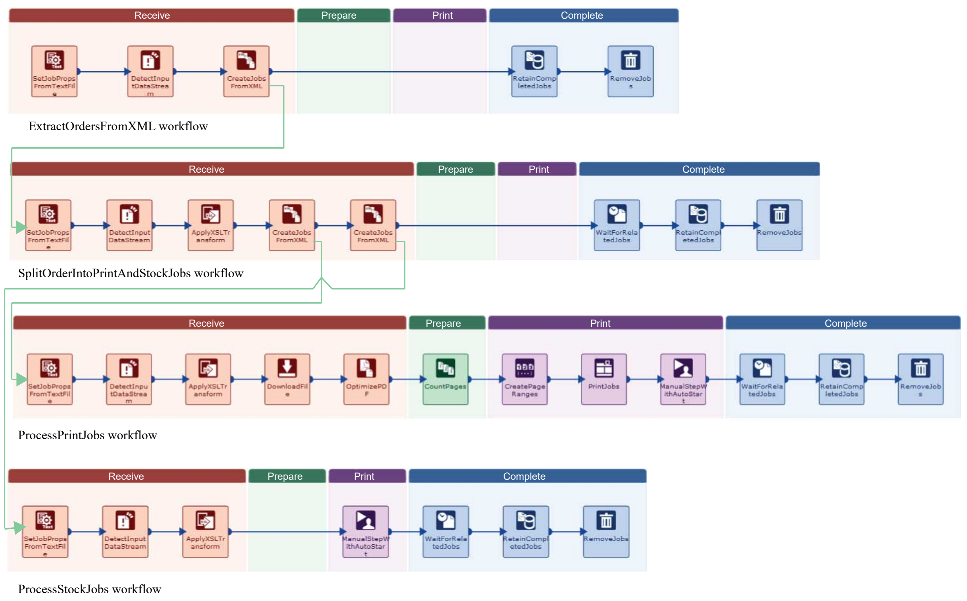
Setting up the input device and workflows

The administrator sets up a hot folder input device to receive XML files and four workflows to process them.

1. The **ExtractOrdersFromXML** workflow creates a job for each order in the XML input file.
   After the **SetJobPropsFromTextFile** and **DetectInputDataStream** steps, the administrator adds the **CreateJobsFromXML** step. The administrator sets the value of the XPath expression to `/seller/order` and the value of the workflow for new jobs to **SplitOrderIntoPrintAndStockJobs**. The administrator specifies that new jobs are not created as child jobs. Each order is an independent job.
2. The **SplitOrderIntoPrintAndStockJobs** workflow receives each order job, creates XML files for print and promotional items, and submits them as child jobs to the appropriate workflow.
   After the **SetJobPropsFromTextFile** and **DetectInputDataStream** steps, the administrator adds an **ApplyXSLTransform** step. The step uses the Order Jobs XSLT style sheet.
   After the **ApplyXSLTransform** step, the administrator adds two **CreateJobsFromXML** steps:
   - For the first step, the administrator sets the value of the XPath expression to `/order/item/printfile/ancestor::item` and the value of the workflow for new jobs to **ProcessPrintJobs**.
   - For the second step, the administrator sets the value of the XPath expression to `/order/item/stock/ancestor::item` and the value of the workflow for new jobs to **ProcessStockJobs**.
   - For both steps, the administrator specifies that new jobs are created as child jobs. The administrator also specifies that the parent jobs continue to the next step when no child jobs are created. For example, child jobs are not created in the first **CreateJobsFromXML** step if an order job does not have any print items.

   After the second **CreateJobsFromXML** step, the administrator adds a step based on the **WaitForRelatedJobs** step template. This step holds the parent jobs until all the child jobs for print and promotional items have been processed.
3. The **ProcessPrintJobs** workflow receives each job for a print item, and prints the items.
   After the **SetJobPropsFromTextFile** and **DetectInputDataStream** steps, the administrator adds an **ApplyXSLTransform** step. The step uses the Print Jobs XSLT style sheet.
   After the **ApplyXSLTransform** step, the administrator adds a step based on the **DownloadFile** step template to download the PDF file specified by the value of the **Job.DownloadFile** property. The administrator sets the value of the **Path to downloaded file** property to `${getFileName(print,pdf,write)}` so that the PDF file is downloaded to the spool directory for the job.
   Following the **DownloadFile** step, the administrator adds the steps that the printing company uses to process PDF jobs.
   After the last step that processes the print jobs, the administrator adds a step based on the **ManualStepWithAutoStart** step template. This step gives a shipping clerk time to add the print job to the order in the shipping department.
   After the **ManualStepWithAutoStart** step, the administrator adds a step based on the **WaitForRelatedJobs** step template. This step holds each job for a print item in an order until all the jobs in the order have been processed.
4. The **ProcessStockJobs** workflow receives each job for a promotional item, and accounts for the time required to retrieve a promotional item from the warehouse.
   After the **SetJobPropsFromTextFile** and **DetectInputDataStream** steps, the administrator adds an **ApplyXSLTransform** step. The step uses the Stock Jobs XSLT style sheet.
   After the **ApplyXSLTransform** step, the administrator adds a step based on the **ManualStepWithAutoStart** step template. This step gives a shipping clerk time to get the promotional item from the warehouse and add it to the order in the shipping department.
   After the **ManualStepWithAutoStart** step, the administrator adds a step based on the **WaitForRelatedJobs** step template. This step holds each job for a promotional item in an order until all the jobs in the order have been processed.

This figure shows the four workflows. The green lines show the workflow that each **CreateJobsFromXML** step submits XML jobs to.



Processing jobs through the workflows

When the administrator enables the workflows and submits an XML file to the input device, RICOH ProcessDirector processes the job through the four workflows.

The **ExtractOrdersFromXML** workflow receives the XML job from the input device and creates an XML file for each `order` element that matches the Order Jobs XPath expression. The workflow submits each XML file as a job to the **SplitOrderIntoPrintAndStockJobs** workflow.

The **SplitOrderIntoPrintAndStockJobs** workflow does this processing:

- It receives each order job.
- It uses the Order Jobs XSLT style sheet to convert `order number` and `customername` elements and values into job properties and values.
- It records the job properties and values in an overrides file.
- It creates an XML file for each `item` element that matches the Print Jobs XPath expression. The workflow submits each XML file as a child job to the **ProcessPrintJobs** workflow. A copy of the overrides file is submitted with each child job.
  It creates an XML file for each `item` element that matches the Stock Jobs XPath expression. The workflow submits each XML file as a child job to the **ProcessStockJobs** workflow. A copy of the overrides file is submitted with each child job.
- It holds the parent job for each order until all the child jobs have been processed.

The **ProcessPrintJobs** workflow does this processing:

- It receives each job for a print item.
- It uses the Print Jobs XSLT style sheet to convert the `printfile` and `quantity` elements and values into job properties and values.
- It records the job properties and values by overwriting the contents of the overrides file submitted with the job.
- It uses the value of the **Job.DownloadFile** property to download the PDF file for each print item.
- It prints the PDF file.
- It waits while a shipping clerk checks the values of the **Custom 1** (order number) and **Customer name** properties for the job. The clerk then adds the print item to the order in the shipping department.
  To move the job to the next step in the workflow, the clerk uses the **Manual complete** action.
- It holds each job for a print item until order processing is complete.

The **ProcessStockJobs** workflow does this processing:

- It receives each job for a promotional item.
- It uses the Stock Jobs XSLT style sheet to convert `stock number` and `quantity` elements and values into job properties and values.
- It records the job properties and values by overwriting the contents of the overrides file submitted with the job.
- It waits while a shipping clerk does these tasks:
  - Finds the stock number of the item by checking the value of the **Custom 2** property.
  - Gets the required number of promotional items from the warehouse.
  - Checks the values of the **Custom 1** (order number) and **Customer name** properties for the job.
  - Adds the promotional items to the order in the shipping department.

  To move the job to the next step in the workflow, the clerk uses the **Manual complete** action.
- It holds each job for a promotional item until order processing is complete.

When all the jobs for the print and promotional items in an order have reached the **WaitForRelatedJobs** step, order processing is complete. RICOH ProcessDirector moves the parent job for the order and all the child jobs for the items to the steps based on the **RetainCompletedJobs** step template.

Parent topic: Jobs with XML files