Overview

Usage scenario for processing JSON orders with web services

In this scenario, a printing company wants to process orders retrieved from a website for ordering books. Each order consists of 2 job tickets. One job ticket provides information (including the location of the print file) required to print the book. The other job ticket provides information required to print the cover for the book. The book and its cover go through different production processes, and the printing company must report when the whole order is completed. The website provides a REST web services interface.

The Web Services Enablement feature includes sample objects that show how this scenario works:

- RestfulWebServiceWF workflow
- RestfulWebServiceSample REST web service input device
- JSON orders and job tickets
- XPath expression
- XSLT style sheets
- RICOH ProcessDirector overrides files
- RestfulWebServiceSampleNotify REST web service notification

To examine the RestfulWebServiceWF workflow, see the related task topic about running a workflow that processes orders retrieved from REST web services.

Learning the requirements of the book-ordering website

The printing company learns the requirements to communicate with the REST web services for the book-ordering web site. RICOH ProcessDirector must call 3 web services:

- A call to one web service retrieves JSON information about each book order. If orders are waiting at the website, the web service returns one JSON order in response to each GET call. If no orders are waiting, the web service does not respond to the call.
  Contents of sample JSON order:
  `{"Order": {"orderId": "ORD1238875463", "customername": "Ricoh"}}`
- A call to another web service retrieves JSON information about the job tickets for the book and its cover.
  Sample JSON job ticket information:
  `{"JobTicket": [{ "itemnumber": "1182563839", "copies":"4","media" : "Letter Plain","file" : "http://localhost:15080/restapi/Brochure.pdf","type" : "Brochure","title" : "RPDBestSeller"}, { "itemnumber": "1934194376", "copies":"4","media" : "Letter Preprinted","file" : "http://localhost:15080/restapi/Cover.pdf","type" : "Cover" ,"title" : "RPDBestSeller" }]}`
- A call to a third web service sends the item number of each completed job ticket for an order. When both item numbers for an order have been sent, the web service creates a PDF file that lists the job numbers and item numbers. The web service places the PDF file in a directory.
  Contents of sample PDF file:
  `10000001.2 with item number 1182563839 has been processed10000001.1 with item number 1934194376 has been processed`

> **Note:**
> - The 3 sample web services simulate web services for a book-ordering website but are available locally as part of the Web Services Enablement feature. They return sample data in response to specific requests made by the RestfulWebServiceSample input device, the CallRESTService step in the RestfulWebServiceWF workflow, and the RestfulWebServiceSampleNotify notification.
> - The sample web services generate sample orders with different order IDs and sets of sample job tickets with different item numbers. The rest of the data in the samples is the same.

To make the calls to the web services, the RICOH ProcessDirector administrator learns:

- The values required to authenticate with the book-ordering website.
- The values required to request data from each web service.
- The format of the data provided in the response from each web service.

The administrator uses a browser plug-in to run manual tests that request responses from the REST web services. Examples of plug-ins include RESTClient for Mozilla Firefox and Boomerang for Google Chrome.

Preparing to process the JSON

The administrator reviews the contents of the JSON orders and job tickets returned by the REST web services and identifies information that RICOH ProcessDirector requires. Next, the administrator evaluates the RICOH ProcessDirector capabilities that call web services and process XML and JSON. The administrator decides on the RICOH ProcessDirector objects and sets of steps that process the orders and job tickets and extract

required information. The administrator then defines the XPath expressions and XSLT style sheets required by the objects and steps.

1. To process orders and job tickets in RICOH ProcessDirector, the administrator identifies JSON objects that supply required information. The administrator assigns the values of those objects to RICOH ProcessDirector job properties.
   - To call the web service that retrieves JSON information about the job tickets, the administrator must supply the order ID. The administrator also wants to track the customer name in RICOH ProcessDirector.
     Two members of the JSON order object provide these values: `orderId` and `customername`.
     This table shows the 2 JSON members and the names of the RICOH ProcessDirector job properties that the administrator uses to extract the information.

JSON member	Database name of job property	User interface name of job property
`orderId`	Job.Info.Attr3	Custom 3
`customername`	Job.CustomerName	Customer name

   - To create separate jobs for each job ticket, the administrator must use the item number. To download and print the PDF files for books and covers, the administrator must use the file, copies, and media information in the job ticket. The administrator also wants to track the book type and title in RICOH ProcessDirector.
     Six members of the JSON job ticket object provide these values: `file`, `itemnumber`, `type`, `title`, `copies`, and `media`.
     This table shows these 6 JSON members and the names of the RICOH ProcessDirector job properties.

JSON member	Database name of job property	User interface name of job property
`itemnumber`	Job.Info.Attr1	Custom 1
`file`	Job.Info.Attr2	Custom 2
`type`	Job.Info.Attr4	Custom 4
`title`	Job.Info.Attr5	Custom 5
`copies`	Job.Copies	Job copies requested
`media`	Job.Media	Media

2. Before designing the RICOH ProcessDirector process, the administrator examines the RICOH ProcessDirector objects and step template that call REST web services:
   - REST web service input device
   - REST web service notification
   - CallRESTService step template

   Each of them can make DELETE, GET, PATCH, POST, or PUT calls, and each can process JSON or XML.
   The input device can create a job from every response, or it can evaluate the response for a JSONPath or XPath expression. If the input device does an evaluation, it can create a job each time it finds the expression.
3. The administrator also examines the RICOH ProcessDirector step templates that process XML:
   - A step based on the ApplyXSLTransform step template can convert input in one format into output in another format. For example, the step can convert XML into this output:
     - A text file containing Ricoh ProcessDirector job or document properties.
     - A file with XML in another format for input to the CreateJobsFromXML step.
   - A step based on the CreateJobsFromXML step template can create one or more XML jobs by evaluating the contents of an XML input file. The step can submit the jobs as independent jobs or child jobs to a specified workflow.

   Because ApplyXSLTransform and CreateJobsFromXML do not process JSON, a step based on the ConvertJSONToXML step template can convert JSON to XML for input to those steps.
4. The administrator decides on the basic components of the RICOH ProcessDirector process.
   - To retrieve JSON orders from the book-ordering website, the administrator chooses a REST web service input device.
     Because each response contains a single order, the administrator sets up the input device to create a job from every response.
   - To retrieve JSON job tickets for each order, the administrator chooses 3 steps:
     - A ConvertJSONToXML step converts the JSON orders into XML for input to an ApplyXSLTransform step.
     - The ApplyXSLTransform step converts order ID and customer name values in the XML orders into RICOH ProcessDirector Custom 3 and Customer name job properties.
     - Using the value of the Custom 3 property, a CallRESTService step retrieves the JSON job tickets.
   - To create a job for each job ticket, the administrator chooses 2 steps:
     - A ConvertJSONToXML step converts the JSON job tickets into XML for input to a CreateJobsFromXML step.
     - The CreateJobsFromXML step creates a separate job for each job ticket by evaluating the response using an XPath expression.
   - To extract information from the job tickets and use it to download PDF print files, the administrator chooses 3 steps:
     - An ApplyXSLTransform step converts values in the XML job tickets into RICOH ProcessDirector job properties. For example, the URL of the file to download is converted to the value of the Custom 2 job property.
     - Using the value of the Custom 2 job property, a DownloadFile step downloads the PDF print files.
     - An AssignJobValues step sets the input data stream to PDF to tell the other steps in the workflow the type of file that they are processing.
   - When both PDF jobs in an order are printed, the administrator wants to notify the book-ordering website that the order is ready to ship. Because the website uses REST, the administrator chooses a REST web service notification.
5. For the CreateJobsFromXML step, the administrator needs an XML Path Language (XPath) expression that identifies the job tickets. To define the expression, the administrator examines the XML for the job tickets.

6. For each **ApplyXSLTransform** step, the administrator needs an XSLT style sheet to convert XML elements into job properties. The administrator uses an XSLT tool (such as Altova MapForce) to create 2 XSLT style sheets:
   - An order-to-overrides-file XSLT style sheet converts the order ID and customer name elements of the order into job properties.
   - A job-ticket-to-overrides-file XSLT style sheet converts the file, item number, type, title, copies, and media elements of the job ticket into job properties.

   Each **ApplyXSLTransform** step outputs the job properties in a text file. The value of each **Output file** property specifies an overrides file in the spool directory for the job.
   RICOH ProcessDirector uses the values in the overrides file to set the values of the properties for the job. Values from the orders are used to retrieve job tickets. Values from the job tickets are used to download and process print files.

Setting up the workflow

The administrator sets up a workflow with 2 branches.

1. The administrator creates a workflow named **ProcessBookOrders** and sets the properties of the **SetJobPropsFromTextFile** step.
2. The administrator adds 10 steps to the workflow in this order:
   - **DetectInputDataStream**
     This step sets the **Input data stream** property to **JSON**.
   - **ConvertJSONToXML**
     This step uses the default values for the JSON input file and XML output file.
   - **ApplyXSLTransform**
     This step uses the order-to-overrides-file XSLT style sheet to create a text file that sets values for the **Custom 3** and **Customer name** properties. The step outputs the text file as an overrides file in the spool directory for the job. RICOH ProcessDirector uses the overrides file to set the **Custom 3** and **Customer name** property values for the job.
   - **AssignJobValues**
     This step sets the value of the **Job name** property to `ORDER-`*`order ID`*.
   - **CallRESTService**
     This step makes a GET call to another web service for the book-ordering website. The step specifies order ID as the value of the **Request parameters** property:
     `orderId:${Job.Info.Attr3}`
     The web service returns JSON job ticket information.
   - **ConvertJSONToXML**
     This step uses the default values for the JSON input file and XML output file.
   - **CreateJobsFromXML**
     The administrator sets the value of the XPath expression to `//JobTicket` and the value of the workflow for new jobs to **ProcessBookOrders**. To keep the new jobs connected to the original job for the order, the administrator sets the value of the **Create child jobs** property to **Yes**. To specify order ID as the job name, the administrator sets the value of the **Name for new job property** property to `${Job.Info.Attr3}`.
     The XML jobs that enter the step go to the next step, while the new child jobs are sent down a separate branch of the workflow.
   - **RunHotFolderApplication**
     The administrator adds this step to communicate with the web service that creates PDF files and puts them in a hot folder. First, the web service polls a hot folder for order jobs. After the web service receives 2 requests per order from the **RestfulWebServiceSampleNotify** notification, the web service puts a PDF file in another hot folder. The **RunHotFolderApplication** step specifies the hot folder for order jobs as the value of the **Sending folder** property. The step specifies the hot folder with the PDF files as the value of the **Retrieval folder** property.

     > **Note:**
     > - This scenario includes a **RunHotFolderApplication** step because the sample **RestfulWebServiceWF** workflow uses this step to exchange data with the sample web service. Instead of returning a PDF file when an order is complete, the web service could do a different function. For example, the web service could send an email message to notify the packing department that the order is ready to pick up.

   - **AssignJobValues**
     This step sets the value of the **Input data stream** property to **PDF** so that the PDF files created by the web service can be viewed in the workflow.
   - **RetainCompletedJobs**
3. The administrator connects the **RetainCompletedJobs** step to the **RemoveJobs** step.
4. The administrator creates a second branch to process the child jobs for the books and their covers.
   - The administrator sets a rule on the connector between the **SetJobPropsFromTextFile** and **DetectInputDataStream** steps:
     - **Rule name**: `Parent`
     - **Order of execution**: `1`
     - **Property**: **Job number**
     - **Comparison**: **Unlike**
     - **Value**: `*.*`

     The rule sends JSON jobs that the input device retrieves to the **DetectInputDataStream** step.
   - The administrator adds an **ApplyXSLTransform2** step in the **Receive** phase and connects it to the **SetJobPropsFromTextFile** step.
     The step uses the job-ticket-to-overrides-file XSLT style sheet to create an overrides file that sets values for the **Custom 2**, **Custom 1**, **Custom 4**, **Custom 5**, **Job copies requested**, and **Media** properties. **Custom 2**, **Custom 1**, **Custom 4**, and **Custom 5** are file, item number, type, and title, respectively.
   - The administrator sets a rule on the connector between the **SetJobPropsFromTextFile** and **ApplyXSLTransform2** steps:
     - **Rule name**: `Child`
     - **Order of execution**: `2`

- Property: Job number
- Comparison: Like
- Value: `*.*`

The rule sends child jobs that the CreateJobsFromXML step creates to the ApplyXSLTransform2 step.

- Because the focus of this scenario is web services, the printing portion is simple. A production process could send the books and their covers through different steps in different workflows or branches to different printers.

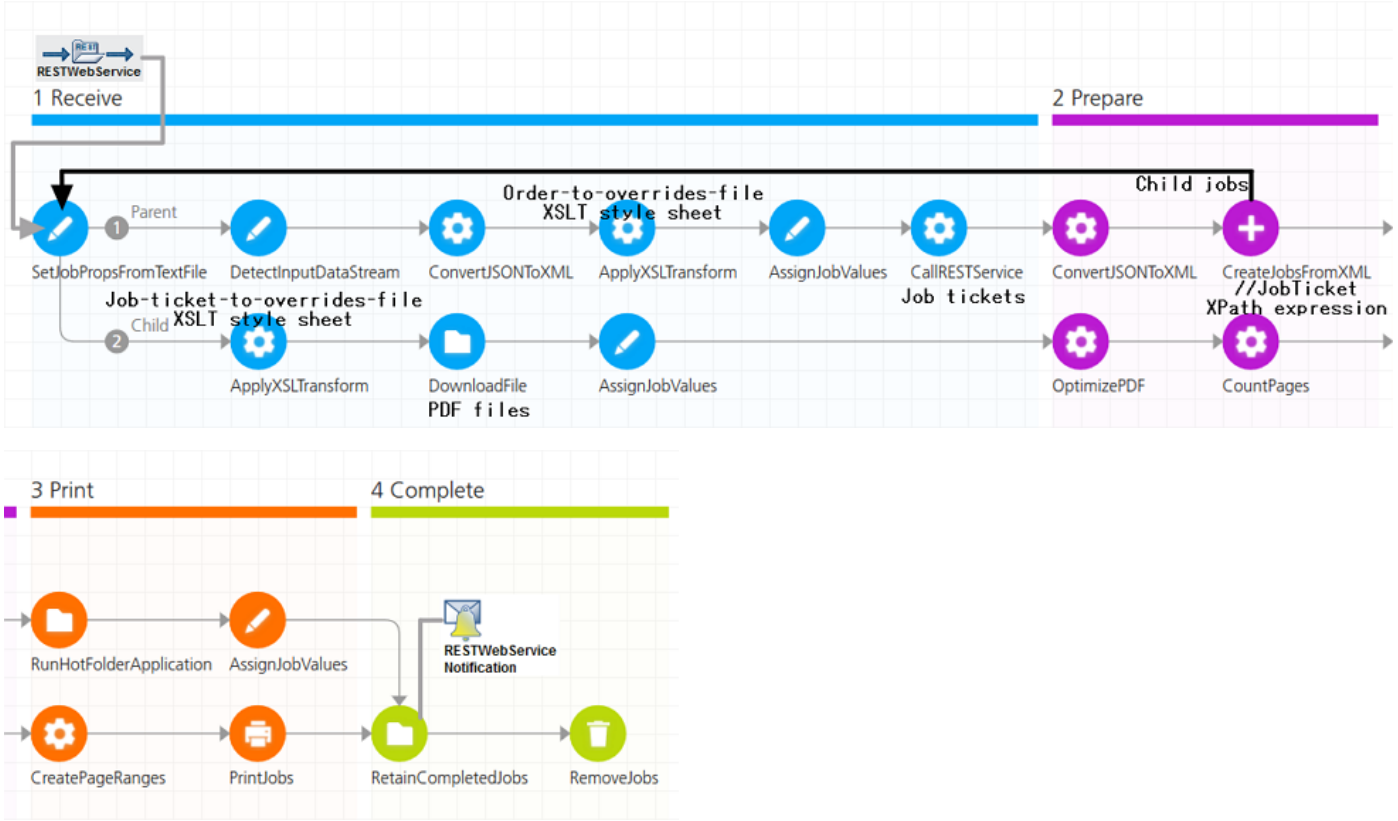
5. The administrator adds 6 steps to the workflow in this order:
   - DownloadFile
     This step downloads PDF files for the book and its cover from a website. The step sets the value of the URL for download file property to Custom 2, which contains the file download information from the job ticket.
   - AssignJobValues
     This step sets the value of 3 job properties:
     - Requested printer
     - Input data stream
     - Job name

     The value of Job name is `${Job.Info.Attr5}-${Job.Info.Attr4}`: the book title, a hyphen, and the type of print job (book or cover).
   - OptimizePDF
   - CountPages
   - CreatePageRanges
   - PrintJobs

> **Note:**
> - Because the focus of this scenario is web services, the printing portion of the workflow has only 4 steps. A production process could send the books and their covers through many more steps in separate branches (or separate workflows) to different printers.

6. The administrator completes the workflow by connecting the PrintJobs step to the RetainCompletedJobs step.

This figure shows the input device, workflow, and notification. The black line indicates that the CreateJobsFromXML step submits the child jobs to the ProcessBookOrders workflow.



Setting up the input device

On the General tab of the REST web service input device, the administrator sets the value of the child workflow property to ProcessBookOrders. The input device submits JSON orders retrieved from the web service for the book-ordering website to that workflow. The administrator sets the polling interval to 30 seconds to retrieve orders quickly when several of them are waiting at the website. Other properties are set to match the environment of the printing company.

On the Request tab, the administrator specifies the values required to retrieve JSON orders from the REST web service. Because each response contains one JSON order, the administrator sets the value of the Create jobs from response property to Always. The network at the printing company does not require a proxy server.

> **Note:**
> - To use a proxy server, set the **Use proxy server** property to **Yes**. Proxy server properties are set on the **Proxy server** tab of the System Settings page.

On the Authentication tab, the administrator specifies the values required to authenticate with the REST web service.

> **Note:**
> - The supplied **RestfulWebServiceSample** input device shows how to set up a REST web service input device for this scenario. The input device calls a web service that does not require authentication. For information about authenticating with a REST web service, see the related task topic.

Setting up the notification

On the Request tab of the REST web service notification, the administrator specifies the values required by the REST web service that receives the notification. The network at the printing company does not require a proxy server.

> **Note:**
> - To use a proxy server, set the **Use proxy server** property to **Yes**.

On the Authentication tab, the administrator specifies the values required to authenticate with the REST web service.

> **Note:**
> - The supplied **RestfulWebServiceSampleNotify** notification shows how to set up a REST web service notification for this scenario. The notification calls a web service that does not require authentication. For information about authenticating with a REST web service, see the related task topic.

On the Event tab, the administrator specifies the notification event information:

- **Event type**: Job
- **Property**: Current job state
- **Action**: Changes to
- **Value**: Retained

On the Conditions tab, the administrator specifies the notification conditions information:

- **Apply any or all of the following conditions** = All
- **Property 1**: Workflow
- **Comparison 1**: =
- **Value 1**: ProcessBookOrders
- **Property 2**: Job number
- **Comparison 2**: like
- **Value 2**: `*.*`

The summary of the conditions statement is:

```
Job.JobType = 'RestfulWebServiceWF' AND Job.ID like '*.*'
```

When all the child jobs for an order reach the **RetainCompletedJobs** step, the notification calls the web service. The web service creates a PDF file containing the item number of each job ticket in the order. The web service puts the file in a hot folder monitored by the **RunHotFolderApplication** step. The PDF file appears in the jobs table. Employees at the printing company can view the PDF file to confirm that the book and cover for the order have been printed.

Processing jobs through the workflow

The **RestfulWebServiceWF** workflow supplied with the Web Services Enablement feature is similar to the **ProcessBookOrders** workflow. To see how jobs are processed through the **ProcessBookOrders** workflow, run the **RestfulWebServiceWF** workflow. For more information, see the related task topic about running a workflow that processes orders retrieved from REST web services.

Parent topic: Web Services Enablement