



**Kodak**

# Prinergy Rules-Based Automation

Advanced Level  
Version 7.5

Activity Guide

Self-Study  
English



## Copyright

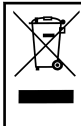
© Kodak, 2016. All rights reserved.

Some documentation is distributed in Portable Document Format (PDF). You may reproduce such documentation from the PDF file for internal use. Copies produced from the PDF file must be reproduced in whole.

## Trademarks

Kodak, Creo, Connect, Direct, Evo, InSite, Maxtone, Pandora, Powerpack, Preps, Prinerger, Publish, SQUAREspot, and Staccato are trademarks of Kodak.

## Equipment recycling



In the European Union, this symbol indicates that when the last user wishes to discard this product, it must be sent to appropriate facilities for recovery and recycling.

Contact your local Kodak representative or refer to <http://www.kodak.com/go/recycle/> for additional information on the collection and recovery programs available for this product.

## REACH

Please consult <http://www.kodak.com/go/REACH> for information about the presence of substances included on the candidate list according to article 59(1) of Regulation (EC) No. 1907/2006 (REACH).

<http://graphics.kodak.com/>

Internal 747-00021A-EN Rev. A

Revised 2016-05-08



# Contents

---

Learning description.....	1
Using this activity guide.....	3
Purpose of this guide.....	3
Guiding principles.....	3
Learning material.....	3
Your role as the student.....	4
Activity guide structure.....	4
Activity outline.....	5
Activity structure.....	5
Activity 1: Create a rule set with a generic exception handler.....	9
Overview.....	9
What you need to know.....	9
How to create a rule set with a generic exception handler.....	11
Get started.....	11
Create a generic exception handler.....	12
Complete and enable the rule set.....	13
Run the rule set.....	14
Activity review questions.....	15
Activity 2: Register the schema.....	17
Overview.....	17
What you need to know.....	17
How to add and register a schema.....	20
Register a new schema.....	20
Validate an XML file.....	21
Activity review questions.....	23
Activity 3: Associate a rule set with a job hot folder.....	25
Overview.....	25
How to associate a rule set with a job hot folder.....	26
Create a rule set activated by a job hot folder drop.....	26
Create a job hot folder and connect it to the enabled rule set.....	27
Activity review questions.....	29
Activity 4: Hot swap the rule set and create a job group.....	31
Overview.....	31
How to hot swap the rule set and create a job group.....	32
Add a Create Job Group action configured to the schema.....	32
Hot swap the rule set.....	34
Activity review questions.....	36
Activity 5: Set up an exception handler.....	37
Overview.....	37
How to add an exception handler to the rule set.....	38
Add an exception handler.....	38
Test the exception handler.....	39

Activity 6: Get the rule set to create a job.....	41
Overview.....	41
How to automatically create a job.....	42
Get the rule set to create a job.....	42
Activity 7: Get the rule set to add and refine input files.....	45
Overview.....	45
How to get the rule set to add and refine input files.....	46
Get the rule set to add the input files.....	46
Get the rule set to refine the input files.....	47
Activity 8: Get the rule set to import an imposition.....	49
Overview.....	49
How to get the rule set to import an imposition.....	50
Create a new import process template.....	50
Get the rule set to import an imposition.....	51
Activity 9: Get the rule set to perform APA.....	53
Overview.....	53
How to get the rule set to perform APA.....	54
Get the rule set to perform APA.....	54
Activity 10: Get the rule set to create final output.....	57
Overview.....	57
What you need to know.....	57
How to get the rule set to create final output.....	59
Get the rule set to create final output.....	59
Activity review questions.....	61
Activity 11: Use temporary variables to simplify event references.....	63
Overview.....	63
What you need to know.....	64
How to create, assign, and use temporary variables.....	66
Create temporary variables.....	66
Assign the values from the XML file to the temporary variables.....	69
Use the temporary variables as action parameters.....	70
Activity review questions.....	72
Activity 12: Create a table of values.....	73
Overview.....	73
How to create a table of values.....	74
Create an email action for output failure.....	74
Create an email action for output success, with a table of values for the To parameter.....	75
Set the Otherwise parameter.....	78
Activity review questions.....	80
Activity 13: Set up remote triggers.....	81
Overview.....	81
What you need to know.....	82

How to set up a remote trigger.....	83
Add a remote trigger in the receiver rule set.....	84
Update the XML file.....	85
Test receiver rule set.....	86
Create a remote sender event.....	87
Trigger the receiving rule set using the sending rule set.....	89
Activity review questions.....	90
<b>Activity 14: Get the rule set to populate custom fields.....</b>	<b>91</b>
Overview.....	91
What you need to know.....	92
How to get the rule set to populate custom fields.....	93
Create the required Prinerly custom fields.....	94
Modify a copy of the remote sender rule set to call the new remote receiver rule set.....	94
Modify a copy of the remote receiver rule set to use custom fields.....	95
Run the rule set.....	100
Activity review questions.....	101
<b>Activity 15: Locate and rename files in a folder.....</b>	<b>103</b>
Overview.....	103
Locate and rename files in a folder.....	104
Create a new rule set that triggers when the job status changes to In Prepress.....	104
Locate all PDF files in the UserDefinedFolder.....	104
Prepend the job name to each file if it is missing.....	105
Add the files to the job and test the rule set.....	105
Activity review questions.....	106
<b>Activity 16: Update a daily production record with the names of all pages approved that day.....</b>	<b>107</b>
Overview.....	107
What you need to know.....	107
Update a daily production record with the names of all pages approved that day.....	109
Create a new rule set and define the rule set variables.....	109
Configure the rule set to write each page approval to the production record.....	110
Create the initial CSV file on the disk.....	112
Test the rule set.....	112
Activity review questions.....	113
<b>Activity 17: Send an e-mail with a summary of a daily production record.....</b>	<b>115</b>
Overview.....	115
Send an e-mail with a summary of a daily production record.....	117
Create the skeleton for an infinitely repeating rule set.....	117
Use the production record name as the limiter to serialize access to the record.....	118
Register a schema for the Approval Record CSV file.....	119
Read the CSV file.....	119
Use temporary variables to summarize the contents of the CSV file.....	120
Send the report by e-mail.....	122
Delete the record.....	123
Recreate the CSV file with a header row.....	123
Add a standalone exception handler.....	124
Test the rule set.....	125

---

Activity review questions.....	126
Activity 18: Actions and events of interest.....	127
Overview.....	127
How to use the Transfer Files action.....	128
Configure the Transfer Files action.....	128
Create a rule set that uses Transfer Files.....	130
How to use the Exception event.....	131
Create a rule set with a generic Exception handler.....	132
Create a rule set with a specific Exception handler.....	133
Include the rule set name in the exception e-mail message.....	135
How to enable/disable hot folders.....	137
Enable/disable hot folders.....	137
How to trigger events when a rule set is enabled/disabled.....	140
Trigger events when a rule set is enabled/disabled.....	140
How to use other events and actions.....	142
Configure transfer files.....	142
Use the Publish File Done event.....	142
Use the Page Edit Done event.....	142
Appendix A: Assessment answers.....	145



# Learning description



## Target audience

- Prepress operators
- Prepress system administrators
- Prepress managers



## Time required

Approximately 4-6 hours to complete.



## Learner prerequisites

Learners should have a basic understanding of Rules-Based Automation and should be familiar with all the workflows covered in the RBA basic self-study guide.

**Important:** It is highly recommended to complete the RBA basic self-study guide *before* starting this advanced self-study guide. If you have not completed the basic RBA training, you will not gain the maximum benefit from this guide.

Learners should have a good understanding of all issues related to Kodak Prinerger Connect software workflows, and possess advanced knowledge of the requirements specific to their own prepress environment.

Learners should also have a basic understanding of software coding terms and basic programming skills.



## Learning objectives

To consolidate existing knowledge of Kodak Prinerger Rules-Based Automation software, to master more advanced processes, and apply this knowledge to your prepress environment.



# Using this activity guide

## Purpose of this guide

The purpose of this self-study guide is to provide realistic experience using advanced processes in Rules-Based Automation (RBA) in a Prinergy prepress environment.

## Guiding principles

This guide embodies the following adult learning principles:

- Conceptual materials provide a view of the larger picture.
- Hands-on practice enables learning by doing.
- Scenarios connect learning to real-world working environments.
- Activities offer opportunities to practice new skills in a safe environment.

## Learning material

This *Rules-Based Automation Self-Study Guide* and the associated training materials are located on Kodak's Partner Place web portal at <https://partnerplace.kodak.com/>. To access the *Rules-Based Automation Self-Study Guide*, follow these steps:

1. In the Partner Place login window, type your e-mail address and password, and click **Continue**.
2. In the Partner Place window, in the **Service & Support** menu, select **Search Knowledgeable Answers**.
3. In the **Search** box, type Prinergy 6.1 Rules-Based Automation Self-Study Guide and Files .

## Your role as the student

You will be an effective student if you follow these guidelines:

- If learning on-site, plan how you will balance the learning session with your regular work responsibilities.
- Read introductory information first.
- Prioritize what you want to learn.
- Complete all appropriate activities.
- Complete all appropriate review activities.
- Assemble your own sample job files..
- Think about workflow issues specific to your site's requirements, and discuss these with the service representative during on-site support.

We hope that you find the RBA training package useful, and we encourage you to provide any feedback by sending an e-mail message to [US&C-GCG-RBA-Feedback@kodak.com](mailto:US&C-GCG-RBA-Feedback@kodak.com). Your comments will help us improve the quality of this material.

## Activity guide structure

An *Activity outline* describes all activities that are included in this guide. View the outline to determine which activities best meet your needs and the requirements of your prepress environment.

Each activity includes the following standard components:

### **Who should complete this activity**

Suggests who should complete an activity based on their job role.

### **Why you should complete this activity**

Defines the importance of completing the activity.

### **Recommended reading**

Lists reading materials that are associated with an activity.

### **Time to complete this activity**

States an approximate time required to complete an activity.

### **Tasks**

Provides steps to follow when performing an activity.

## Activity outline

In the following activities, you will use XML to drive a complete, end-to-end prepress workflow. You will use a "control" job to support a hot folder that will take dropped XML and trigger a rule set that will consume the XML "intent" to provide parameter values to some of the actions within your rule set.

You will build the workflow progressively, using the RBA Debugger to validate each step of the rule set creation. You will have a generic Exception Handler logging to job history as appropriate.

At the end of the activities you will be able to drop XML into a job hot folder and have your rule set behave according to the intent, i.e., create the specified job in the intended group with all of the job elements as described by the intent.

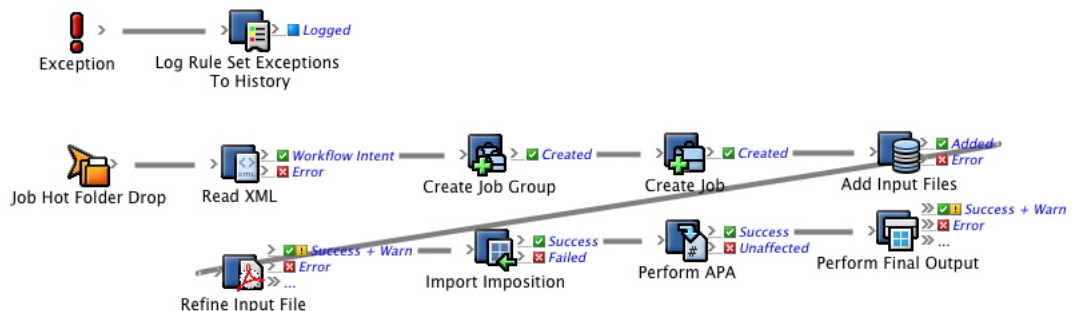
For the sake of this self-study guide, we have provided both a schema and an XML file with the intent. The XML file includes information about how to name the job, which server and job share to store the job on, which input files and imposition to use, and how to assign the pages using APA. If desired, it is possible to open and edit this XML file.

See the outline to view a brief description of each activity.

It is recommended to complete activities in sequential order, as each activity continues from the previous one.

**Important:** It is highly recommended to complete the RBA basic self-study guide *before* starting this advanced self-study guide.

The final rule set will look like this:



## Activity structure

The following table provides a brief description of all activities in this guide.

Activity name and description	Time to complete (minutes)
<p><b>Activity 1: Create a rule set with a generic exception handler</b></p> <p>Demonstrates how to capture exceptions that are thrown when a rule set is run.</p>	15-20
<p><b>Activity 2: Register the schema</b></p> <p>Demonstrates how to register a schema, to enable RBA to drive automatic processes when a customer makes an order via external software (for example, an eCommerce site).</p>	10-15
<p><b>Activity 3: Associate a rule set with a job hot folder</b></p> <p>Demonstrates how to create a rule set that is triggered by a job folder drop, which tells Prinerger to read the XML in the dropped file.</p>	10-15
<p><b>Activity 4: Hot swap the rule set and create a job group</b></p> <p>Demonstrates how to get the rule set to automatically create a job group, while working on a copy of the enabled rule set, and then hot swapping the rule set so that the updated copy is enabled.</p>	15-20
<p><b>Activity 5: Set up an exception handler</b></p> <p>Reviews and reinforces tasks such as creating an exception handler and hot swapping, while continuing to build the rule set.</p>	10-15
<p><b>Activity 6: Get the rule set to create a job</b></p> <p>Demonstrates how to get the rule set to automatically create a new job after the XML file is dropped into the hot folder.</p>	10-15
<p><b>Activity 7: Get the rule set to add and refine input files</b></p> <p>Demonstrates how to get the rule set to automatically add and refine the relevant input files.</p>	10-15

Activity name and description	Time to complete (minutes)
<p><b>Activity 8: Get the rule set to import an imposition</b></p> <p>Demonstrates how to get the rule set to import an imposition, using the information from the XML file.</p>	10-15
<p><b>Activity 9: Get the rule set to perform APA</b></p> <p>Demonstrates how to get the rule set to automatically perform APA (automatic page assignment).</p>	10-15
<p><b>Activity 10: Get the rule set to create final output</b></p> <p>Demonstrates how to get the rule set to automatically create the final output.</p>	10-15
<p><b>Activity 11: Create a table of values</b></p> <p>Demonstrates how to set conditions, so that RBA will perform a different action, depending on which condition has been fulfilled.</p>	25-30
<p><b>Activity 12: Set up remote triggers</b></p> <p>Demonstrates how to set up a remote trigger, in order to get one RBA rule set to be triggered via another "remote" rule set.</p>	30-45
<p><b>Activity 13: Get the rule set to populate custom fields</b></p> <p>Builds on the previous Remote Trigger activity and adds Dashboard-like functionality with the use of Prinergy custom fields. As the rule set runs, custom fields are populated to display information about the progress and output of the workflow.</p>	30-45
<p><b>Activity 14: Actions and events of interest</b></p> <p>Discusses some additional events and actions that you might find useful.</p>	30-45

Activity name and description	Time to complete (minutes)
<b>Activity 15: Locate and rename files in a folder</b> Demonstrates how to use the List Directory and Rename One File actions to identify and correct files that do not conform to a specific naming standard.	30-45
<b>Activity 16: Update a daily production record with the names of all pages approved that day</b> Demonstrates how to safely access a shared resource from multiple instances of a rule set.	25-30
<b>Activity 17: Send an e-mail with a summary of a daily production record</b> Demonstrates how to use the Read CSV action to read a CSV file, use variables to tabulate a production report, and properly implement a rule set that can infinitely repeat.	30-45



# Create a rule set with a generic exception handler

## Overview



### Why you should complete this activity

This activity demonstrates how to create a rule set that tracks exceptions and stores information about these errors in the log history.



### Target audience

Prepress operators with some RBA experience



### Time required

Approximately 15-20 minutes



### What you'll learn

After completing this activity, you will know how to perform the following tasks:

- Get started
- Create a generic exception handler
- Complete and enable the rule set
- Run the rule set



### Recommended Reading

- *Prinerger Connect Workflow 6.1 Rules-Based Automation User Guide*
- *Prinerger Powerpack Workflow 6.1 Rules-Based Automation User Guide*

## What you need to know

In this activity, you will create a new job and set up a generic exception handler to capture any exception errors in the log history.

### **Basic tasks**

Students should be familiar with Prinerger and should have completed the *RBA Self-Study Guide - Basic Level*. It is assumed that students can perform basic tasks such as group and job setup without guidance.

In many cases, simple and obvious instructions, such as when to click **OK** or when to double-click the red line separating the event and action have been skipped.

If you require more guidance, please refer to the *RBA Self-Study Guide - Basic Level*.



## How to create a rule set with a generic exception handler

### Scenario

You want a rule set to include a mechanism to capture exceptions that occur when the rule set is run. This way, if any action in the rule set throws an exception, you'll be able to check the log history to see what went wrong.

In this activity, we're going to deliberately create a rule set that contains errors, which will cause an exception to be thrown, in order to demonstrate how the generic exception handler works.

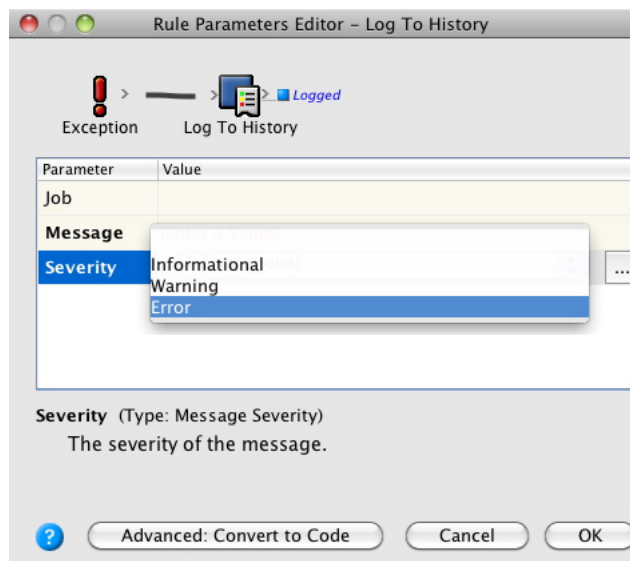
Follow these procedures:

### Task 1: Get started

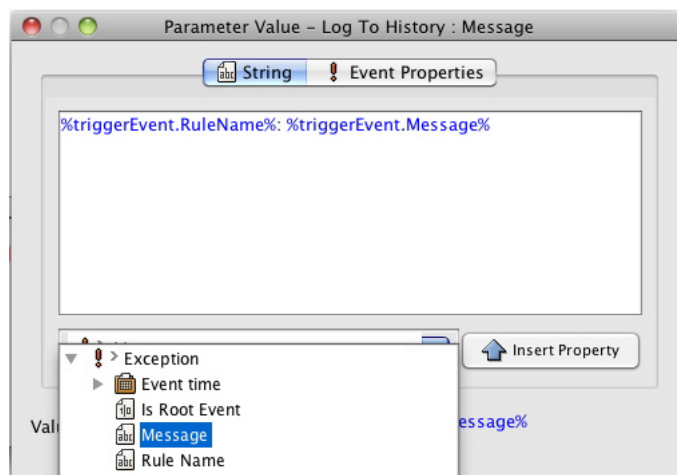
1. Locate the activity files on either the RBA Student Resources DVD or on Kodak's Partner Place web portal at [https://partnerplace.kodak.com/Service & Support/Search Knowledgebase Answers](https://partnerplace.kodak.com/Service%20&%20Support/Search%20Knowledgebase%20Answers).
2. Drag the `Rules-Based Automation Advanced Activity Files` folder directly to your desktop.  
If preferred, copy the folder to a local server that is accessible to your Prinergy system.
3. Start Prinergy and create a group named **<XX>\_RBA\_Self-Study**, (where <XX> represents your initials).
4. Create a job within that group called **<XX>\_RBA\_Job**.
5. Open **<XX>\_RBA\_Job** and from the **Tools** menu, open Rule Set Manager.
6. Create a Rule Set Manager group named **<XX>\_RBA\_Self-Study**, (where <XX> represents your initials).

## Task 2: Create a generic exception handler

1. Add a generic exception handler for the rule set, by dragging **Targeted events > Exception** to the work space. Place this above the rules that are already in the rule set.
2. Drag **Prinergy Actions > Log to History** to the resulting event.
3. Double-click the red line separating the event and action.
4. In the **Severity** field, select **Error**.



5. In the **Job** field, browse for and select **<XX>\_RBA\_Job**.
6. In the **Message** field, insert the properties **Rule Name** and **Message** to concatenate the rule name and exception error into one stored history message.



7. Rename the **Log to History** action as **Log Rule Set Exceptions to Job History** to describe the purpose of the action more clearly. This rule will be triggered if any other action in the rule set causes an exception, and it will log the rule name and the exception message to job history.

### Task 3: Complete and enable the rule set

1. Add an **Input File Added** trigger event to the rule set.
2. Add a **Refine Input File** action to the resulting event.
3. Configure the action to use **1st-Ref Normz** as a process template.  
**Note:** Note that RBA uses Auto-Assignment to populate the **Input Files** parameter.
4. Add a **Perform Loose Page Output** action to the **Success + Warn** generated event on the **Refine Input Files** action.
5. Select **Virtual Proof.Loose Page** in the **Process Template** parameter.  
**Note:** Note that RBA Auto-Assignment has populated the **Pages** parameter.  
**Note:** The RBA rules engine examines the relationship between a resulting event and the newly connected action to determine if it is appropriate to provide any of the action parameters with a default value. This would be a value found somewhere in the rule chain before the insertion point of the new action.
6. Open **Process Template Overrides** and configure the **Number of Copies** parameter to be: 1/0 (one divided by zero). This will force an exception.
7. Click **OK**.
8. Right-click the **Perform Loose Page Output** action and select **Show Exceptions**. This makes it available on the **>>** resulting event of the action.
9. Drag a **Sent Email** action to the **>>** resulting event of the **Perform Loose Page Output** action and attach it to **! > Exception**.
10. Configure the email action and include the rule name and message in the body parameter.  
**Note:** You can have multiple exception handlers in a rule set, for example, a generic one that will handle all action exceptions and specific ones (like the one you just created). This means that for any one exception, you can actually have two different workflows, in this case, one logging to history and the other sending an email.
11. Save the rule set as **<XX>\_RBA\_Debugger** and enable it in **<XX>\_RBA\_Job**.

#### Task 4: Run the rule set

1. Add the input file `Type_Brochure.PDF` to the job and witness the refining and proofing activities.
2. Add different input files and watch the processing. Remember to ensure that the **Process Selected Files Using Process Template** check box is left unchecked.
3. Display the **History** view to view the history log. Note the exceptions.



## Review what you know

### **Answer or consider the following:**

1. What is the benefit of adding a generic exception handler to a rule set?
  
2. What are the benefits of having a "specific" exception handler in a rule set as well as a "generic" handler?
  
3. What will be the results of concatenating the rule name and the exception error into one stored history message?

*Answers to review questions are located in the Appendix section of this guide.*





## Register the schema

---

### Overview



#### Why you should complete this activity

This activity introduces the XML Schema Manager and demonstrates how to register a new schema.



#### Target audience

Prepress operators with some RBA experience



#### Time required

Approximately 10-15 minutes



#### What you'll learn

After completing this activity, you will know how to perform the following tasks:

- Register a new schema
- Validate an XML file



#### Recommended Reading

- *Prinerly Connect Workflow 6.1 Rules-Based Automation User Guide*
- *Prinerly Powerpack Workflow 6.1 Rules-Based Automation User Guide*

### What you need to know

In this activity, you will add and register a new schema.

#### What is intent?

The term "intent" can be used to describe the XML file that is produced when a customer makes an order using external software, such as

eCommerce sites, MIS systems, or planning systems. Intent describes what to print and how to print it. This typically includes information about which input files, process templates, and imposition files to use. It may also include instructions about how to name the job and where to store it (which server and job share).

An example of an XML "intent" file:

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <RBASelfStudy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
3   <JobName>XX_Job</JobName>
4   <JobHomeShare>AraxiVolume_HOSTNAME_DRIVELETTER</JobHomeShare>
5   <JobHomeServer>HOSTNAME</JobHomeServer>
6   <JobGroup>XX_Group</JobGroup>
7   <ImpositionPath>\\HOSTNAME\AraxiVolume_HOSTNAME_DRIVELETTER\RESOURCEDIRECTORY\AustraliaTour.pjtf</ImpositionPath>
8   <APACCommand>ASSIGN="Aus Tours[#start]-[#end].p[#page].pdf" "*" [#start]+[#page]-1 1</APACCommand>
9   <InputFiles>
10     <string>\\HOSTNAME\AraxiVolume_HOSTNAME_DRIVELETTER\RESOURCEDIRECTORY\Aus Tours1-16.ps</string>
11     <string>\\HOSTNAME\AraxiVolume_HOSTNAME_DRIVELETTER\RESOURCEDIRECTORY\Aus Tours17-32.ps</string>
12     <string>\\HOSTNAME\AraxiVolume_HOSTNAME_DRIVELETTER\RESOURCEDIRECTORY\Aus Tours33-48.ps</string>
13   </InputFiles>
14 </RBASelfStudy>

```

## What is a schema?

A schema is a file that defines the attributes of the intent. This includes:

- The types of elements that are allowed in the document
- Their relationship to each other (for example, which elements contain other elements)
- The types of data that each element contains

An example of a schema:

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
3   <xs:annotation>
4     <xs:documentation> This is a sample schema for use with the Read XML Action in Rules-Based Automation
5       It shows how a simple XML file can be given to RBA for consumption by a rule set.
6
7       This schema describes information for creating and processing a simple Job in Prinergy. It gives:
8       JobName           - name of the Job that should be created
9       JobHomeShare      - Share on which to create the job folder
10      JobHomeServer     - server on which to create the job folder
11      JobGroup          - name of the group to be created in
12      ImpositionPath    - path to the Imposition to import
13      APACCommand       - path to the APA file to use for assigning pages
14      InputFiles        - array of paths to source data files for the job
15
16     </xs:documentation>
17   </xs:annotation>
18   <xs:element name="RBASelfStudy" nillable="true" type="RBASelfStudy" />
19   <xs:complexType name="RBASelfStudy">
20     <xs:sequence>
21       <xs:element minOccurs="1" maxOccurs="1" name="JobName" type="xs:string" />
22       <xs:element minOccurs="1" maxOccurs="1" name="JobHomeShare" type="xs:string" />
23       <xs:element minOccurs="1" maxOccurs="1" name="JobHomeServer" type="xs:string" />
24       <xs:element minOccurs="1" maxOccurs="1" name="JobGroup" type="xs:string" />
25       <xs:element minOccurs="1" maxOccurs="1" name="ImpositionPath" type="xs:string" />
26       <xs:element minOccurs="1" maxOccurs="1" name="APACCommand" type="xs:string" />
27       <xs:element minOccurs="1" maxOccurs="1" name="InputFiles" type="ArrayOfStrings" />
28     </xs:sequence>
29   </xs:complexType>
30   <xs:complexType name="ArrayOfStrings">
31     <xs:sequence>
32       <xs:element minOccurs="1" maxOccurs="unbounded" name="string" nillable="true" type="xs:string" />
33     </xs:sequence>
34   </xs:complexType>
35 </xs:schema>

```

### **Why does the schema need to be registered for RBA?**

Schemas enable RBA to interpret XML data submitted to a rule set. This makes it possible to integrate RBA with external systems, so that when an order is placed, processes run in Prinergy automatically.

For example, each time a customer places an order at an eCommerce site, the web site generates an XML file identifying the order details. When this XML file appears in a Prinergy hot folder, an RBA rule set uses the data in the file to automatically create a job, process the input file, and print the job.

Some eCommerce sites, for example, Kodak InSite Storefront, are already integrated with Prinergy and their schemas are automatically registered. However, other systems produce XML data that is not recognized by Prinergy. Their schemas need to be registered, so that Prinergy will be able to interpret their XML data.



## How to add and register a schema

### Scenario

Let's assume that you want RBA to work with an eCommerce site. You want to set up automatic processes, so that when a customer makes an order at the site, an XML file with the details of the order will be dropped into an RBA hot folder and RBA will then use this data to create a job, refine the input file, and print the job. Before you can start to work with the eCommerce site, you will need to register the site's schema.

In this activity, you will register a schema from the activity files that were provided with this guide. Once the schema is registered and validated, you will then test an XML file against that schema, to make sure that RBA will be able to interpret this file using the schema.

**Note:** Once a schema is registered, it's available to anyone on the system. Therefore, the registration activity only needs to be completed once for each new schema. So if multiple students are using this self-study guide, only the first one needs to do it.

Follow these procedures:

### Task 1: Register a new schema

1. From the **Tools** menu in Rule Set Manager, select **RBA XML Schema Manager**.
2. In the **Actions** section, select **Add a Group**.

**Kodak** RBA XML Schema Manager

**Actions:**

- Add a Schema
- Add a Group
- Edit/Delete Group

**Groups:**

- General
- InSiteCategories
- Kodak
- Regions
- TIS Training

**Add Group**

Name: RBA Self Study Guide

Parent Group: General

Description: Schema for Self Study Guide

Add Group

3. In the **Name** field, enter RBA Self Study Guide.
4. In the **Description** field, enter Schema for RBA Self Study Guide.
5. Click the **Add Group** button.

6. In the **Actions** section, select **Add a Schema**.

**Add Schema**

Name:  Group:

Description:

Schema File:

If the schema allows more than one element type at the root of an XML document, you must specify the Schema Root Element:

If the incoming XML file will not match the above schema, you can specify an XSLT transform file (optional):

7. From the **Group** list, select **RBA Self-Study Guide**.
8. Ensure that **General** is selected in the **Parent Group** list.
9. In the **Name** field, enter RBA Self-Study .
10. In the **Description** field, enter Schema to support RBA Self-study activities (advanced).
11. In the **Schema File** field, click **Browse** and locate the file `Self-study.XSD`. It is located in the `XML` folder within the activity files folder that you copied in Activity 1.
- Note:** Leave the **Schema root element** field empty. RBA can only consume/process one root element from a schema. If more than one were present, you would need to specify the one you want RBA to use. Also, leave the **Transform file** field empty. During the schema registration process, it is possible to apply an XSLT to modify the incoming file. Transform files are not covered in this document.
12. Click the **Add Schema** button.

## Task 2: Validate an XML file

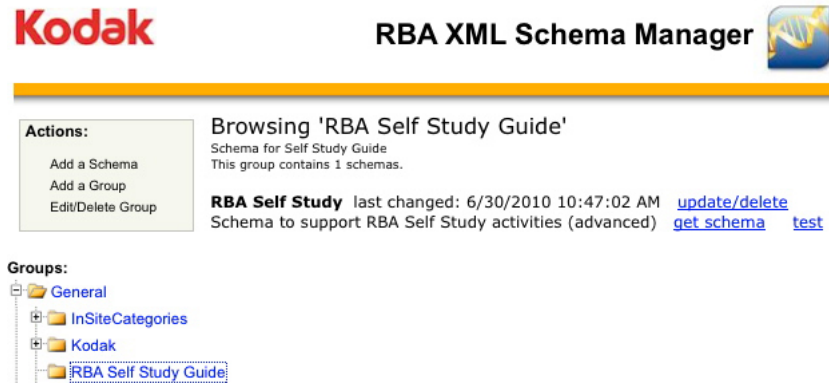
In the activity files that were provided, there is an XML file called `Self-study.XML`. We will assume that this was received from the eCommerce site. This XML file includes information about how to name the job, where to store it, which input files and imposition to use, and how to assign the pages using APA.

In this step, you will validate the XML file against the schema, to ensure that RBA will be able to interpret all the information.

1. In the **Groups** tree in Schema Manager, select **RBA Self-Study Guide**.

The schema that you added, **RBA Self-Study** appears under this group heading.

2. Click **Test**.



**Kodak** **RBA XML Schema Manager**

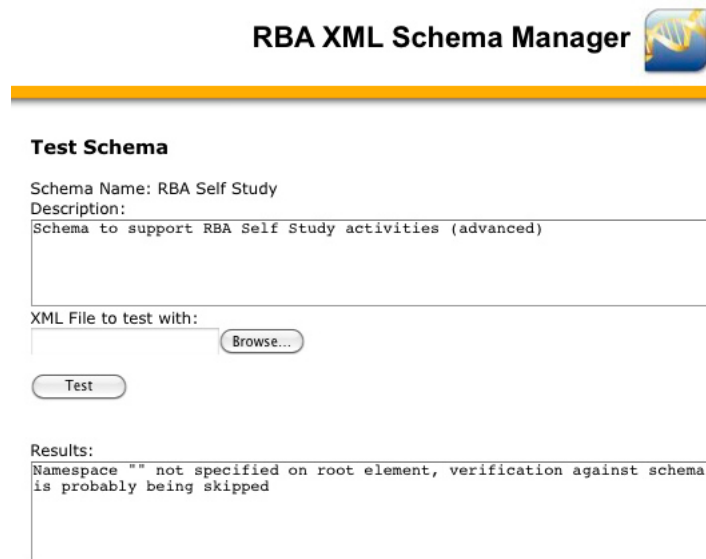
**Actions:**  
 Add a Schema  
 Add a Group  
 Edit/Delete Group

**Browsing 'RBA Self Study Guide'**  
 Schema for Self Study Guide  
 This group contains 1 schemas.

**RBA Self Study** last changed: 6/30/2010 10:47:02 AM [update/delete](#)  
 Schema to support RBA Self Study activities (advanced) [get schema](#) [test](#)

**Groups:**  
 General  
 InSiteCategories  
 Kodak  
 RBA Self Study Guide

3. Click the **Browse** button and locate the file `Self-Study.XML`. It is located in the `XML` folder within the activity files folder that you copied in Activity 1.
4. Click the **Test** button.



**RBA XML Schema Manager**

**Test Schema**

Schema Name: RBA Self Study  
 Description:  
 Schema to support RBA Self Study activities (advanced)

XML File to test with:

**Results:**  
 Namespace "" not specified on root element, verification against schema is probably being skipped

A message is displayed in the **Results** section: Namespace "" not specified on root element, verification against schema is probably being skipped. This means that a namespace was expected but not specified on the root element. This is okay. The XML is valid and will work with RBA.



## Review what you know

### **Answer or consider the following:**

1. What is the purpose of registering a schema in RBA Schema Manager?
2. Do you have to register a schema multiple times?
3. Would you need to update (re-register) a schema if it changes?
4. Can you validate XML for a specific schema before you use it in a rule set?

*Answers to review questions are located in the Appendix section of this guide.*





## Associate a rule set with a job hot folder

### Overview



#### Why you should complete this activity

This activity demonstrates how to create a rule set that is activated by a hot folder drop.



#### Target audience

Prepress operators with some RBA experience



#### Time required

Approximately 10 - 15 minutes



#### What you'll learn

After completing this activity, you will know how to perform the following tasks:

- Create a rule set activated by a job hot folder drop
- Create a job hot folder and connect it to the enabled rule set



#### Recommended Reading

- *Prinerly Connect Workflow 6.1 Rules-Based Automation User Guide*
- *Prinerly Powerpack Workflow 6.1 Rules-Based Automation User Guide*



## How to associate a rule set with a job hot folder

### Scenario

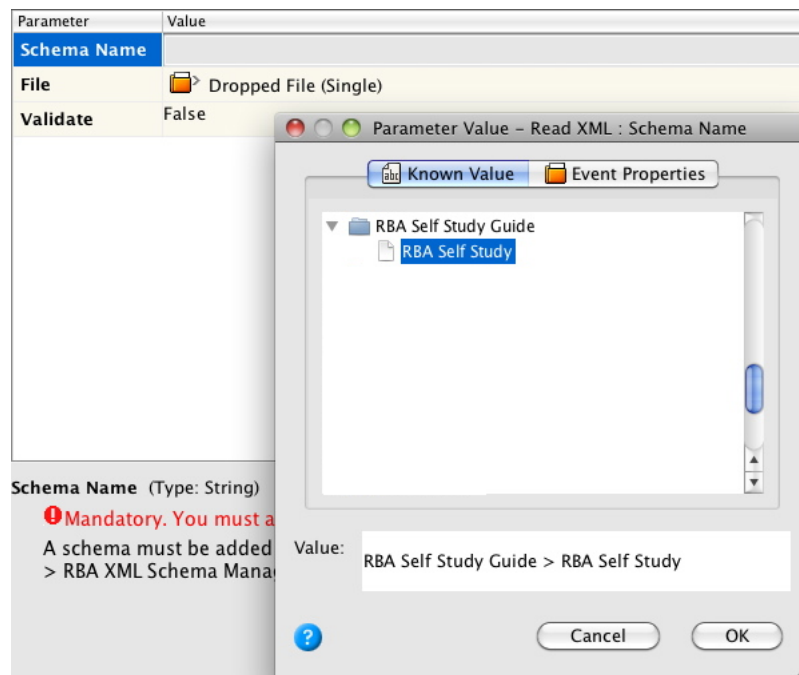
In this activity, you will create a rule set that is associated with a Prinerogy job hot folder. When an XML file is dropped into this hot folder, this will trigger the **Read XML** action. You will configure the **Read XML** action to use the schema that you registered in the previous activity. Then, you will create the hot folder and connect it to the rule set.

The objective is to start the automation process: so that if intent from an external source (such as an eCommerce site) is dropped into the job hot folder, Prinerogy will interpret the information within the document, according to the schema that you registered.

Follow these procedures:

### Task 1: Create a rule set activated by a job hot folder drop

1. Create a new job and name it **<XX>\_Job** (where <XX> represents your initials).
2. Create a new rule set for this job.
3. Create a rule set stub, with the trigger event **Job Hot Folder Drop** and the action **Read XML**.
4. Configure the **Read XML** action to use the **RBA Self Study** schema that you registered in the previous activity.



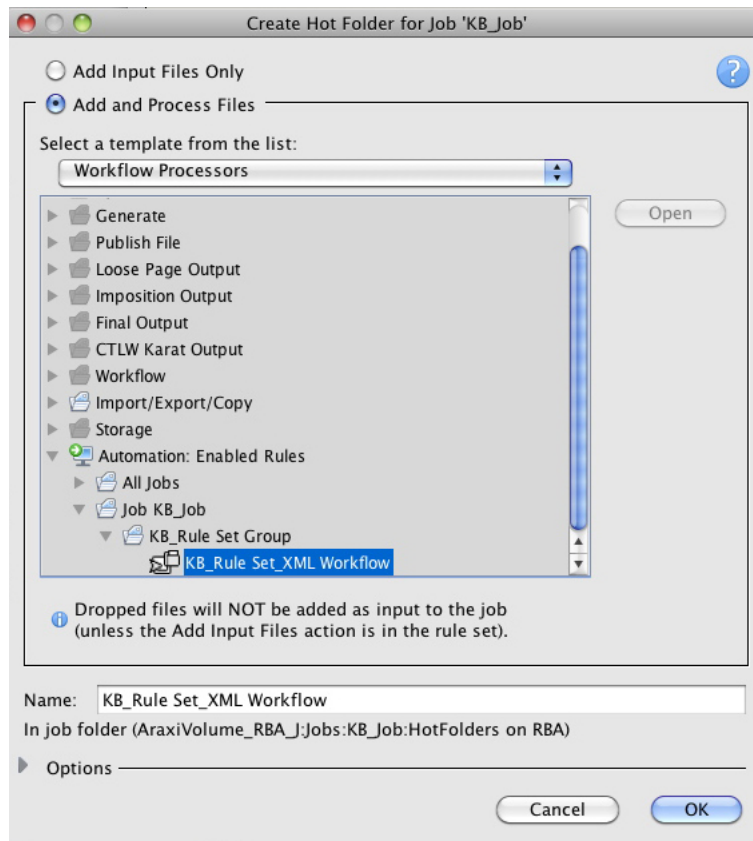
5. Rename the **Success** resulting event to **Workflow Intent**.



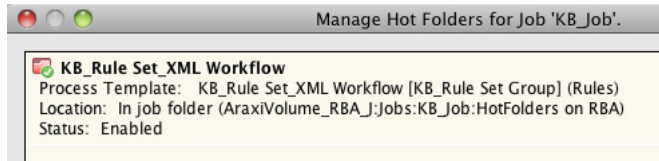
6. Save the rule set stub as **<XX>\_Rule Set\_XML\_Workflow** (where <XX> represents your initials) and enable it in the job.

## Task 2: Create a job hot folder and connect it to the enabled rule set

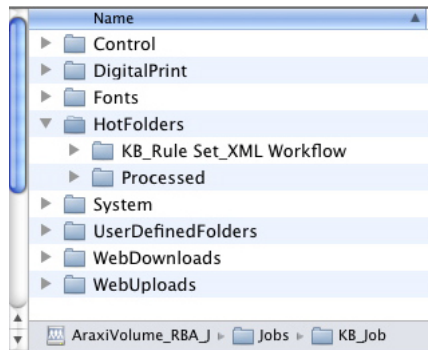
1. Create a job hot folder in the current job.
2. Select the **Add and Process Files** option.
3. Expand **Automation: Enabled Rules** and select your enabled rule set **<XX>\_Rule\_Set\_XML\_Workflow**.



4. In the Manage Hot Folders window, verify that your rule set is connected to the hot folder.



5. View the `Job` sub-folder with newly created hot holder. Note that the hot folder is named after the rule set.





## Review what you know

### **Answer or consider the following:**

1. What do you expect will happen when you drop an XML file in your job's hot folder?
  
2. What will happen if you drop an input file (instead of an XML file) in your job's hot folder?
  
3. What will happen if you drop multiple XML files in your job's hot folder?

*Answers to review questions are located in the Appendix section of this guide.*



## Hot swap the rule set and create a job group

### Overview



#### Why you should complete this activity

This activity demonstrates how to keep a rule set enabled while creating an updated version, by using hot swapping.



#### Target audience

Prepress operators with some RBA experience



#### Time required

Approximately 15-20 minutes



#### What you'll learn

After completing this activity, you will know how to perform the following tasks:

- Add a Create Job Group action configured to the schema
- Hot swap the rule set



#### Recommended Reading

- *Prinerly Connect Workflow 6.1 Rules-Based Automation User Guide*
- *Prinerly Powerpack Workflow 6.1 Rules-Based Automation User Guide*



## How to hot swap the rule set and create a job group

### Scenario

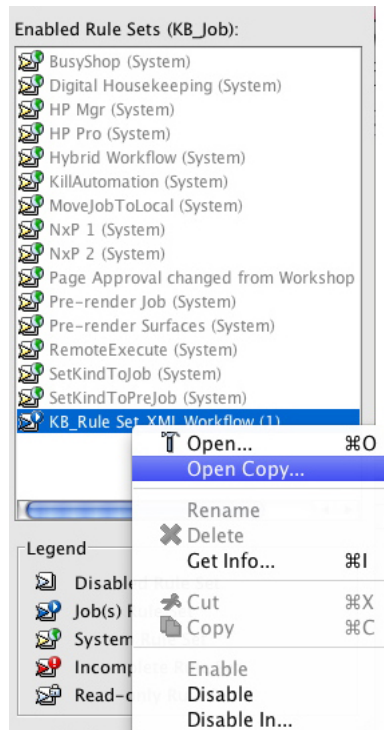
In this activity, you will add a **Create Job Group** action to a copy of the enabled rule set so that when an XML intent file has been read, RBA will automatically create a job group with a name specified in the XML.

By working with a copy of the rule set, you can keep the first rule set enabled until you're ready to hot swap it and enable the updated rule set instead. This way, you can keep the first rule set enabled so that it will continue to process any dropped XML files while you are making your changes, ensuring that RBA doesn't miss a trigger event that the rule set is listening for.

Follow these procedures:

### Task 1: Add a Create Job Group action configured to the schema

1. Right-click the rule set that you created in the previous activity and select **Open Copy** to open a copy of the enabled rule set.

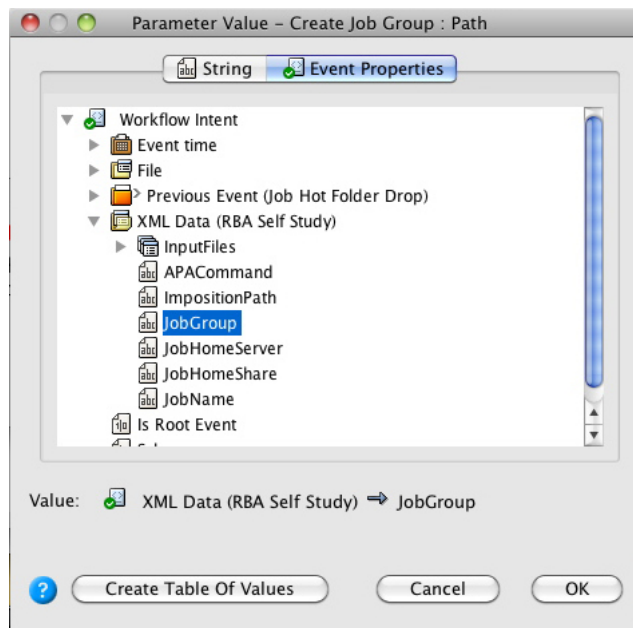




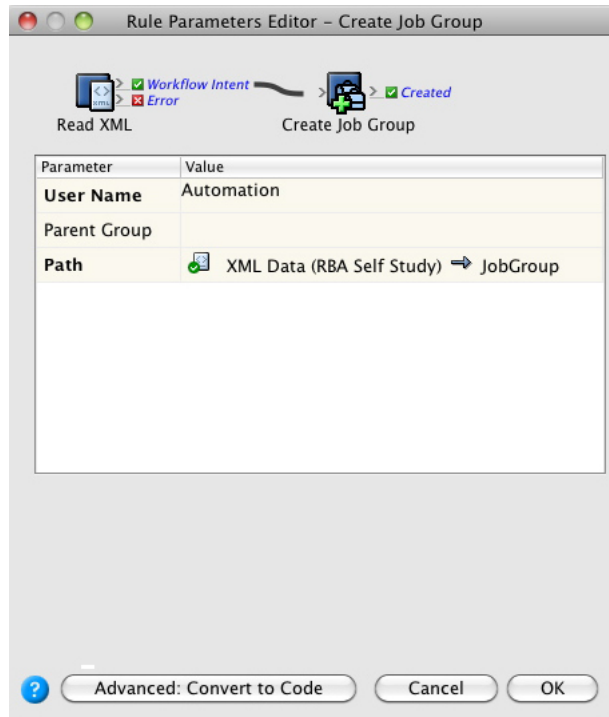
2. Add a **Create Job Group** action to the rule set, by dragging it to the success event of the **Read XML** action, which you renamed to **Workflow intent**.



3. Double-click the **Create Job Group** action, to configure it.
4. In the **Path** field, click the browse button and configure the group path to: **Workflow Intent > XML data (RBA self-study) > Job Group**.

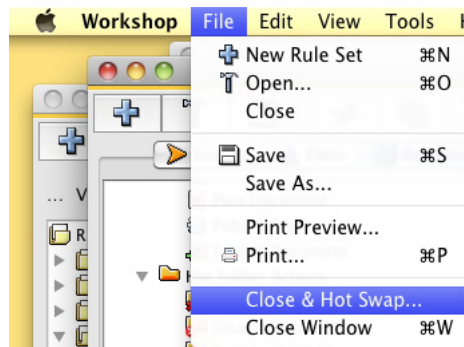


- Note how the name of the schema now appears in the **Path** field.



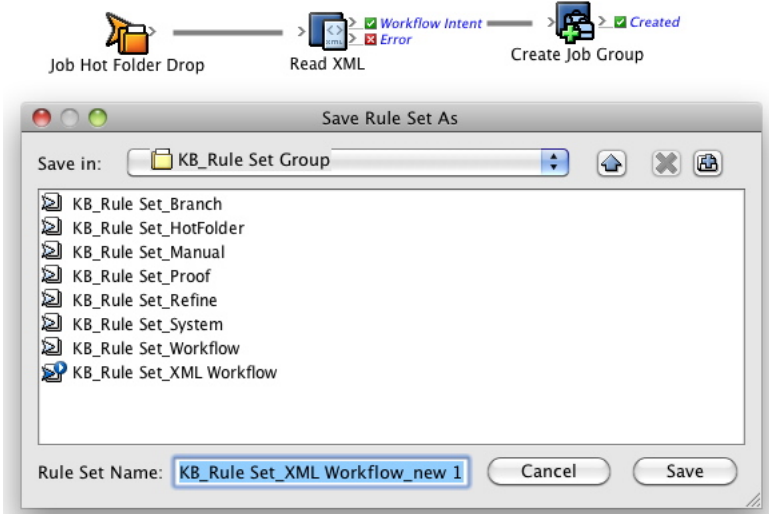
## Task 2: Hot swap the rule set

- From the **File** menu in Rule Builder, select **Close & Hot Swap**.

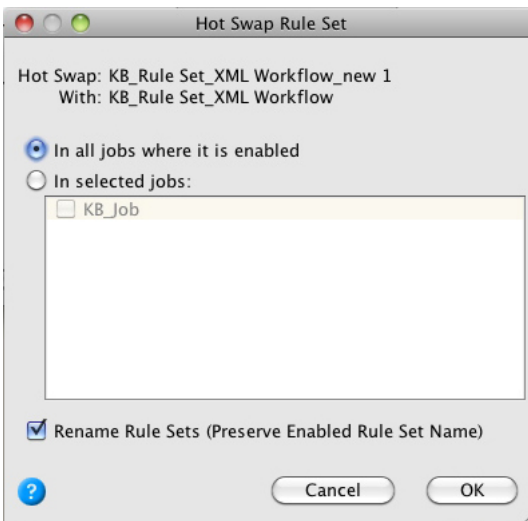


The Save As dialog box appears.

2. Select the default (temporary) rule set name and click **Save**.



3. Make sure that **Rename Rule Sets (Preserve Enabled Rule Set Name)** is selected and click **OK**.



Note that the new version is now enabled and the old version is saved as **<XX>\_rule set\_XML Workflow\_old 1**.



## Review what you know

### **Answer or consider the following:**

1. Why would you hot-swap an enabled rule set instead of disabling, editing, and re-enabling?
  
2. Under what circumstances should you preserve the enabled rule set name on hot-swap?

*Answers to review questions are located in the Appendix section of this guide.*

## Set up an exception handler

---

### Overview



#### Why you should complete this activity

This activity reviews the addition of an exception handler.



#### Target audience

Prepress operators with some RBA experience



#### Time required

Approximately 10-15 minutes



#### What you'll learn

After completing this activity, you will know how to perform the following tasks:

- Add an exception handler
- Test the exception handler



#### Recommended Reading

- *Prinerly Connect Workflow 6.1 Rules-Based Automation User Guide*
- *Prinerly Powerpack Workflow 6.1 Rules-Based Automation User Guide*



## How to add an exception handler to the rule set

### Scenario

The next item that needs to be added to your rule set is an exception handler.

This activity gives you an opportunity for some revision and reinforcement of tasks that you performed earlier in this activity guide, while continuing to build your rule set.

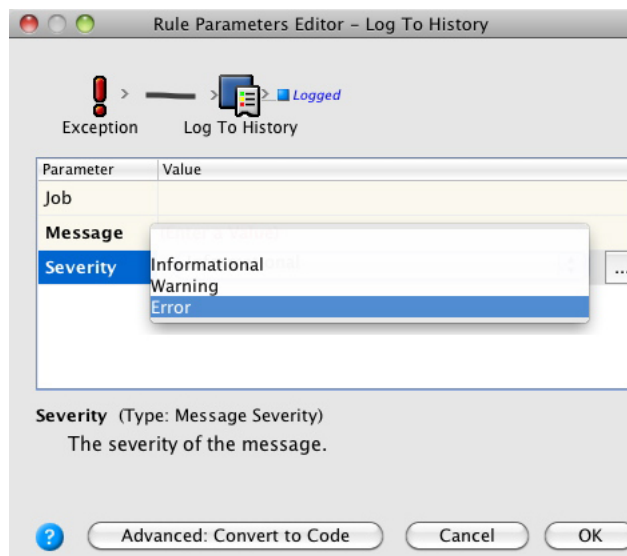
Follow these procedures:

### Task 1: Add an exception handler

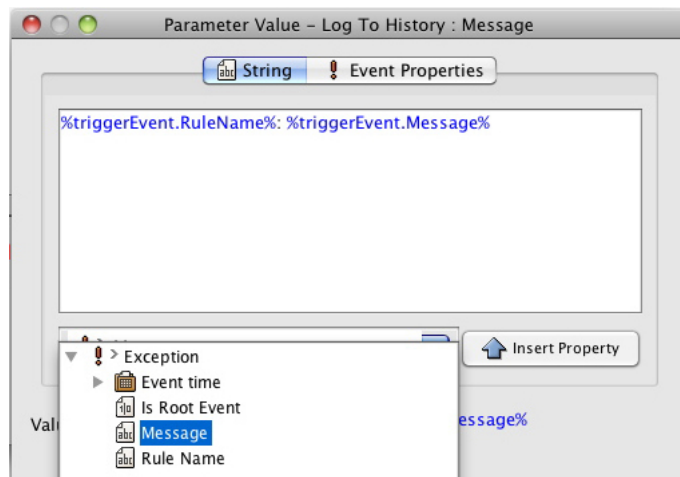
1. Open a copy of the enabled rule set and add an **Exception** trigger event (like you did in Activity 1).
2. Add a **Log To History** action.



3. Set the **Severity** parameter to **Error**.



4. Configure the **Message** parameter to capture the **Rule Name** and the exception **Message** text.



5. Hot swap the changes (like you did in Activity 4).

## Task 2: Test the exception handler

1. Test this portion of the rule set by dropping your XML file into the job hot folder that you set up in Activity 3.
2. Display the **History** view to view the history log and to check for exceptions.





## Get the rule set to create a job

---

### Overview



#### Why you should complete this activity

This activity demonstrates how to get the rule set to automatically create a new job after the XML file is dropped into the hot folder.



#### Target audience

Prepress operators with some RBA experience



#### Time required

Approximately 10-15 minutes



#### What you'll learn

After completing this activity, you will know how to perform the following tasks:

- Get the rule set to create a job



#### Recommended Reading

- *Prinerger Connect Workflow 6.1 Rules-Based Automation User Guide*
- *Prinerger Powerpack Workflow 6.1 Rules-Based Automation User Guide*



## How to automatically create a job

### Scenario

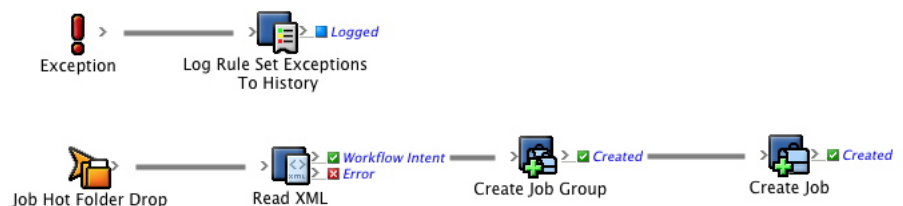
So far, the rule set has been configured so that if an XML file is dropped into the hot folder, RBA will read the XML and will automatically create a job group. It will also monitor the actions for any exceptions.

Now, you're going to get the rule set to automatically create a job in the job group.

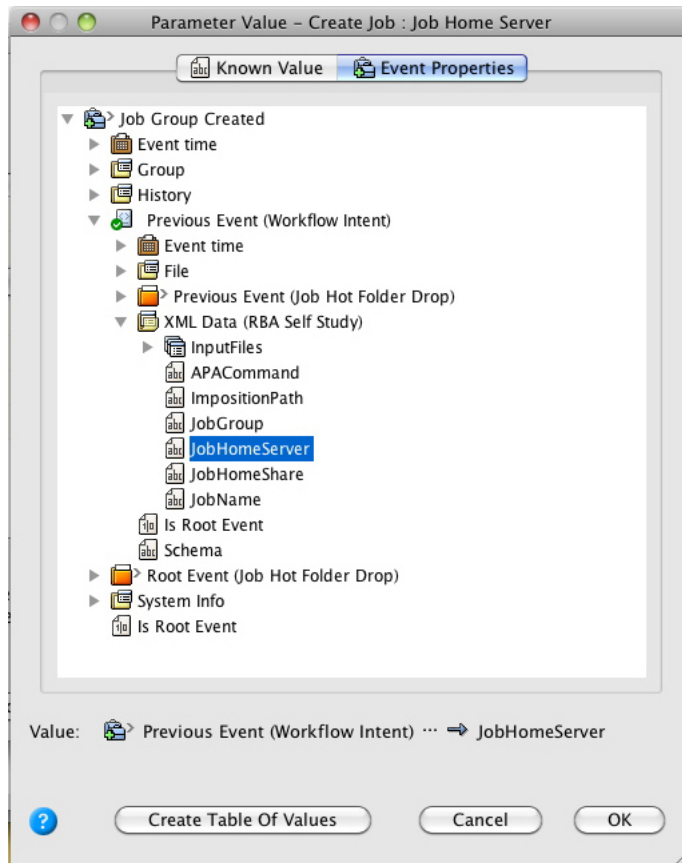
Follow these procedures:

### Task 1: Get the rule set to create a job

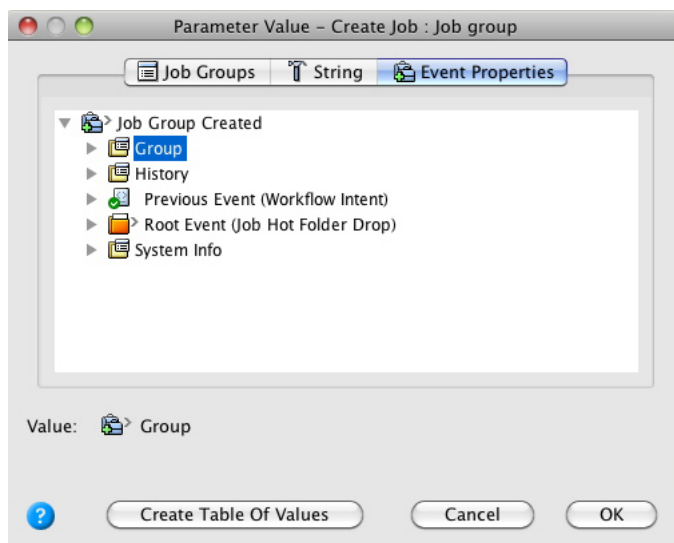
1. Open a copy of the enabled rule set.
2. Drag a **Create Job** action to the **Created** success event from the **Create Job Group** action.



3. In the **User Name** field, enter <XX>\_Automation (where <XX> represents your initials).
4. Configure the **Job Name** parameter to **Previous Event (Workflow Intent) > XML Data (RBA Self-study) > JobName**.  
This tells RBA to name the job as was specified in the XML file.
5. Configure the **Job Home Server** parameter to **Previous Event (Workflow Intent) > XML Data (RBA Self-study) > JobHomeServer**.  
This tells RBA to create the job on the server that was specified in the XML file.



6. Configure the **Job Home Share** parameter to **Previous Event (Workflow Intent) > XML Data (RBA Self-study) > JobHomeShare**.
7. Configure the **Job Group** parameter to **Job Group Created > Group**.



The Rule Parameters Editor should look like this:

The image displays a workflow diagram and a dialog box for configuring a job creation rule.

**Workflow Diagram:**

- Job Hot Folder Drop
- Read XML (Workflow Intent, Error)
- Create Job Group (Created)
- Create Job (Created) - highlighted in green

**Rule Parameters Editor - Create Job:**

Parameter	Value
User Name	KB_Automation
New Job Name	KB_XML_Job
Job Home Server	Previous Event (Workflow Intent) → XML Data (RBA Self Study) → JobHomeServer
Job Home Share	Previous Event (Workflow Intent) → XML Data (RBA Self Study) → JobHomeShare
Job group	Group
Template Job	
Create From Template Job Options	

Buttons: Advanced: Convert to Code, Cancel, OK

8. Save and hot swap the rule set with the one that is currently enabled.
9. Run the enabled rule set in the Debugger.

**Note:** For more information about the Debugger, see the *Debugger Self-study Guide*.

## Get the rule set to add and refine input files

---

### Overview



#### Why you should complete this activity

This activity demonstrates how to get the rule set to automatically add the relevant input files and refine them.



#### Target audience

Prepress operators with some RBA experience



#### Time required

Approximately 10-15 minutes



#### What you'll learn

After completing this activity, you will know how to perform the following tasks:

- Get the rule set to add the input files
- Get the rule set to refine the input files



#### Recommended Reading

- *Prinerly Connect Workflow 6.1 Rules-Based Automation User Guide*
- *Prinerly Powerpack Workflow 6.1 Rules-Based Automation User Guide*



## How to get the rule set to add and refine input files

### Scenario

So far, the rule set has been configured so that if an XML file is dropped into the hot folder, RBA will read the XML and will automatically create a job group and job, while monitoring the actions for any exceptions.

Now, you're going to get the rule set to automatically add input files to the job and refine them.

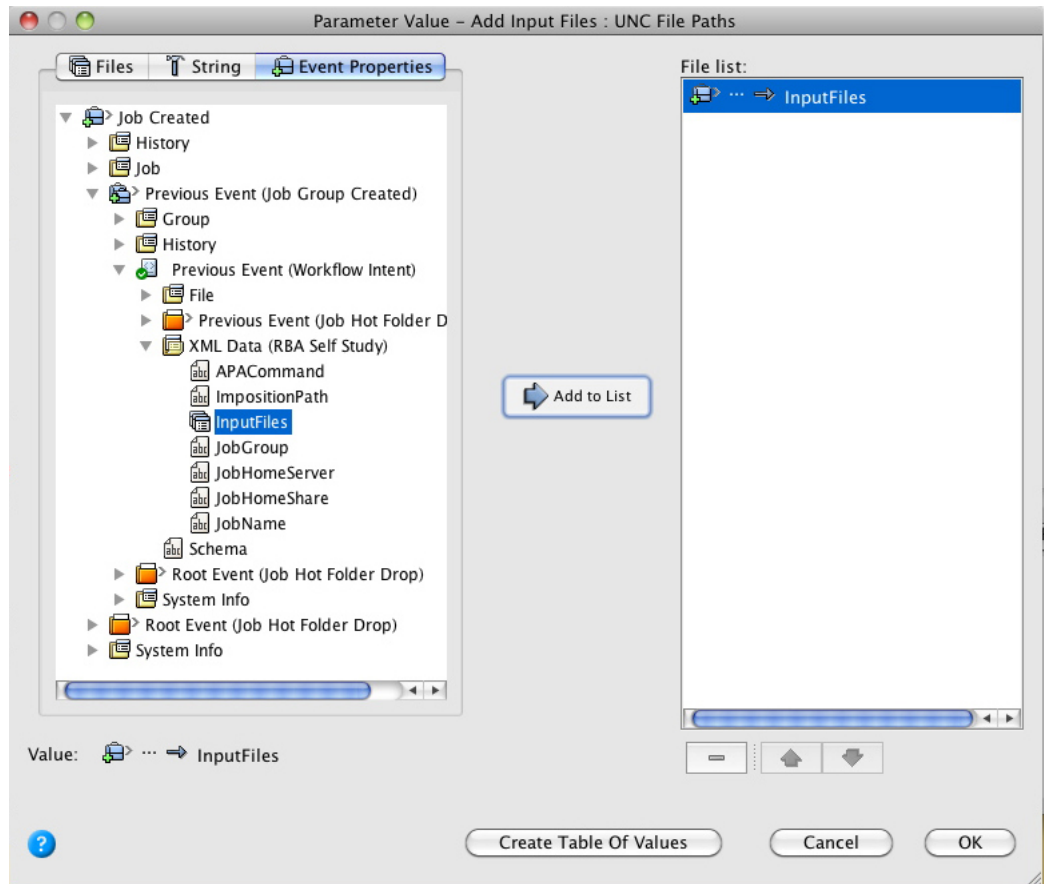
Follow these procedures:

### Task 1: Get the rule set to add the input files

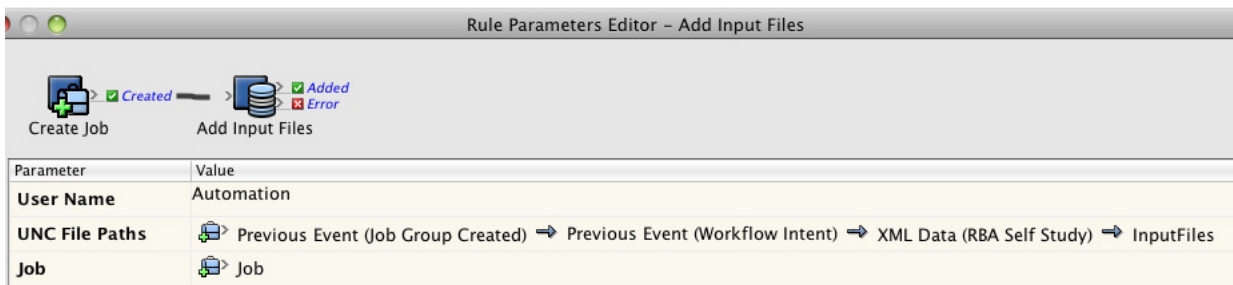
1. Open a copy of the enabled rule set.
2. Drag an **Add Input Files** action to the **Created** success event from the **Create Job** action.



3. Configure the **UNC File Paths** parameter to **Previous Event (Job Created) > Previous Event (Job Group Created) > Previous Event (Workflow Intent) > XML Data (RBA Self-study) > Input Files**. This tells RBA to look at the XML file to see which input files to use.

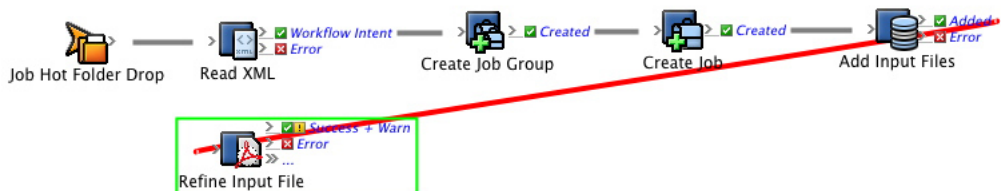


The Rule Parameters Editor should look like this:

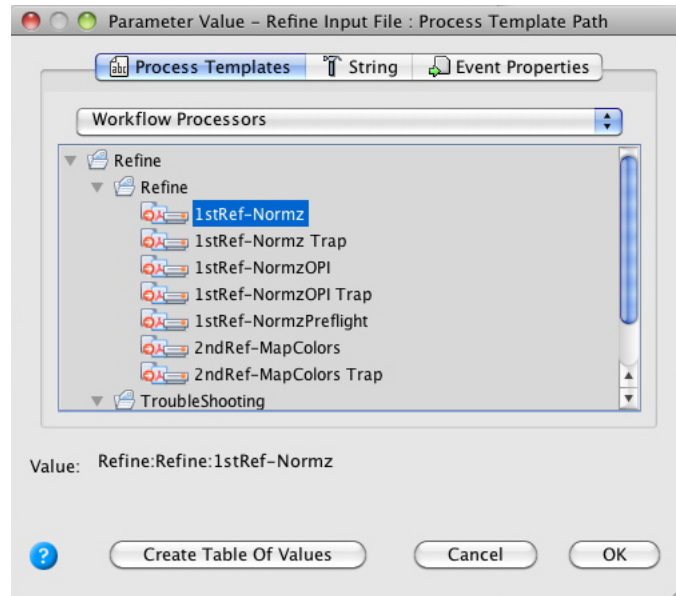


### Task 2: Get the rule set to refine the input files

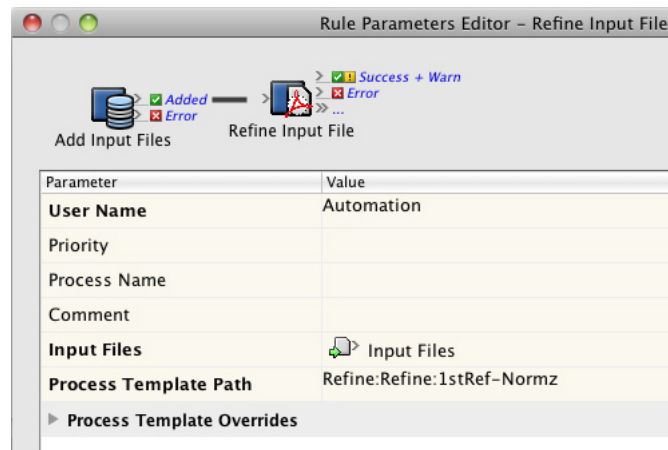
1. Drag a **Refine Input File** action to the **Added** success event from the **Add Input Files** action.



2. Verify that the **Input Files** parameter is configured as **Input File Added > Input Files**.  
This should occur automatically.
3. Configure the **Process Template Path** parameter to use the **1stRef-Normz** process template.



The Rule Parameters Editor should look like this:



4. Save and hot swap the rule set with the one that is currently enabled.
5. Run the enabled rule set in the Debugger.



## Get the rule set to import an imposition

### Overview



#### Why you should complete this activity

This activity demonstrates how to get the rule set to import an imposition, using the information from the XML file.



#### Target audience

Prepress operators with some RBA experience



#### Time required

Approximately 10-15 minutes



#### What you'll learn

After completing this activity, you will know how to perform the following tasks:

- Create a new import process template
- Get the rule set to import an imposition



#### Recommended Reading

- *Prinerly Connect Workflow 6.1 Rules-Based Automation User Guide*
- *Prinerly Powerpack Workflow 6.1 Rules-Based Automation User Guide*



## How to get the rule set to import an imposition

### Scenario

The rule set has now been configured so that if an XML file is dropped into the hot folder, RBA will read the XML and automatically create a job group and job, while monitoring the actions for any exceptions. It will then automatically add input files to the job and refine them.

Now, you're going to get the rule set to import an imposition, using the information within the XML file.

Follow these procedures:

### Task 1: Create a new import process template

1. Open the **ImportRBAImposition** import process template from the Process Template Editor.
2. Select **File > Save As** and save the process template as **<XX> ImportRBAImposition\_DeleteExisting** (where <XX> represents your initials).

**Tip:** Open the **Import** section.

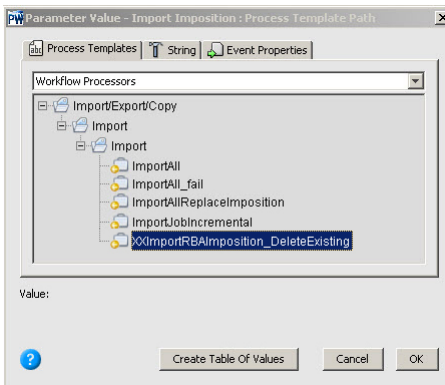
3. From the **If Page Set Already Exists** drop down list, select **Delete existing Page set and Imposition**.  
The **Delete Existing Page set and Imposition** option will replace the existing page set and imposition with the page set and imposition being imported. Making this adjustment will ensure that a new page set and imposition are not created and imported each time your RBA rule set is run. If you do not make this adjustment, you will find that your rule set triggers multiple VPS output and email notifications.
4. Save the process template.

## Task 2: Get the rule set to import an imposition

1. Open a copy of the enabled rule set.
2. Drag an **Import Imposition** action to the **Success & Warn** event from the **Add Input Files** action.

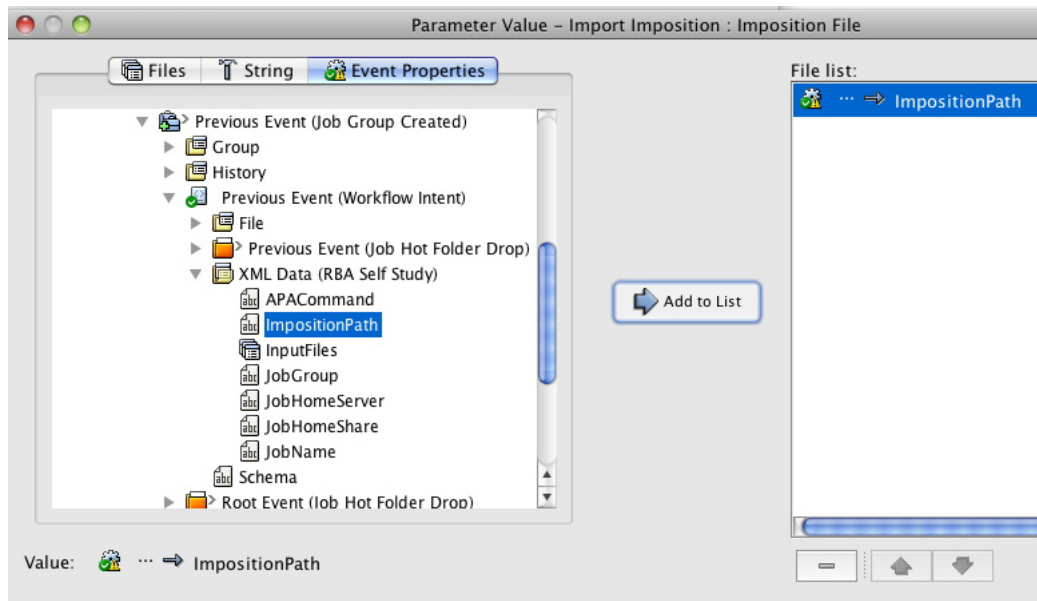


3. Configure the **Process Template Path** parameter to use the **<XX>ImportRBAImposition\_DeleteExisting** process template.



4. Configure the **Imposition File** parameter to **Previous Event (Input File Added) > Previous Event (Job Created) > Previous Event (Job Group Created) > Previous Event (Workflow Intent) > XML Data (RBA Self-study) > ImpositionPath**.

This tells RBA to look at the XML file to see which imposition to import and from where to import it.



5. Save and hot swap the rule set with the one that is currently enabled.
6. Run the enabled rule set in the Debugger.

## Get the rule set to perform APA

---

### Overview



#### Why you should complete this activity

This activity demonstrates how to get the rule set to automatically perform APA.



#### Target audience

Prepress operators with some RBA experience



#### Time required

Approximately 10-15 minutes



#### What you'll learn

After completing this activity, you will know how to perform the following tasks:

- Get the rule set to perform APA



#### Recommended Reading

- *Prinerger Connect Workflow 6.1 Rules-Based Automation User Guide*
- *Prinerger Powerpack Workflow 6.1 Rules-Based Automation User Guide*



## How to get the rule set to perform APA

### Scenario

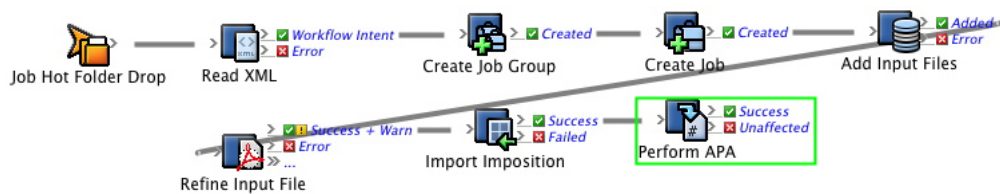
The rule set has been configured so that if an XML file is dropped into the hot folder, RBA will read the XML and automatically create a job group and job, while monitoring the actions for any exceptions. It will then automatically add input files to the job, refine the input files, and import an imposition.

Next, you need to add a step that will perform automatic page assignment on the imposition that was imported.

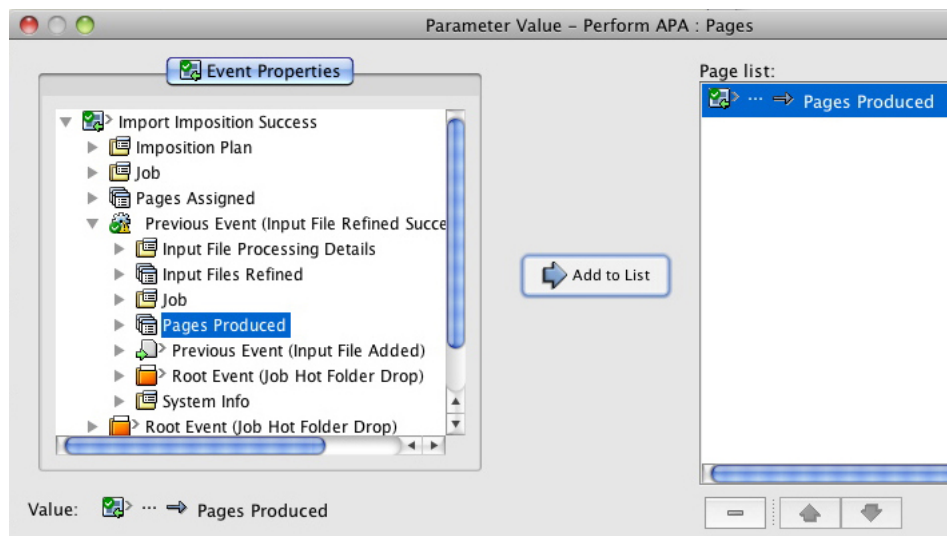
Follow these procedures:

### Task 1: Get the rule set to perform APA

1. Open a copy of the enabled rule set.
2. Drag a **Perform APA** action to the **Success** event from the **Import Imposition** action.

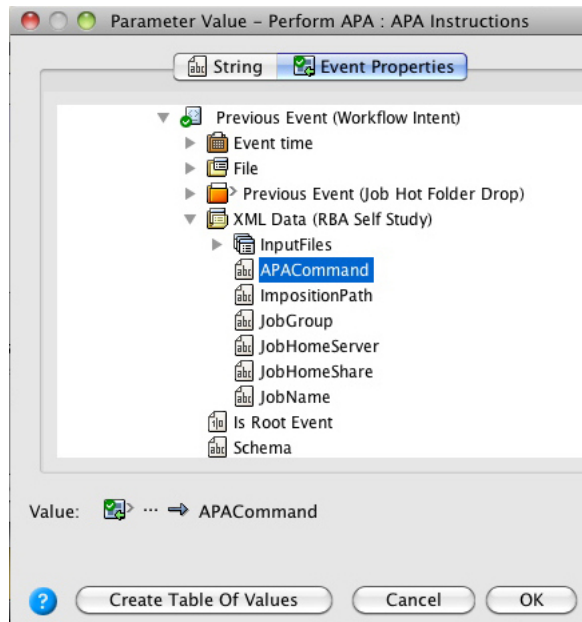


3. Configure the **Pages** parameter to **Import Imposition Success > Previous Event (Input File Refined Success and Warn) > Pages Produced**.  
This tells RBA to use the pages that were produced in the previous refine action.

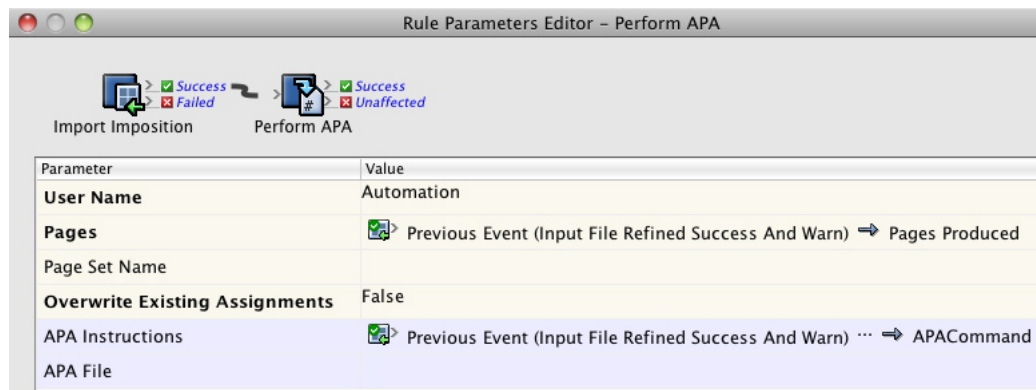


4. Configure the **APA Instructions** parameter to **Previous Event (Input File Refined Success and Warn) > Previous Event (Input File Added) > Previous Event (Job Created) > Previous Event (Job Group Created) > Previous Event (Workflow Intent) > XML Data (RBA Self-study) > APA Command**.

This tells RBA to look at the XML file for instructions about how to assign the pages.



The Rule Parameters Editor should look like this:



5. Save and hot swap the rule set with the one that is currently enabled.
6. Run the enabled rule set in the Debugger.





## Get the rule set to create final output

---

### Overview



#### Why you should complete this activity

This activity demonstrates how to get the rule set to create final output.



#### Target audience

Prepress operators with some RBA experience



#### Time required

Approximately 10-15 minutes



#### What you'll learn

After completing this activity, you will know how to perform the following tasks:

- Get the rule set to create final output



#### Recommended Reading

- *Prinerger Connect Workflow 6.1 Rules-Based Automation User Guide*
- *Prinerger Powerpack Workflow 6.1 Rules-Based Automation User Guide*

### What you need to know

#### Parameter mutual exclusivity

In some actions, typically output actions, some parameters can be marked as mutually exclusive and will appear with a light blue background shading. This means that at runtime, only one of those parameters can have a value.

At rule set creation time, it is possible and valid to specify a value for each parameter if the rule set trigger event is a **Manual Start** event.

### **Assignment hints**

The RBA rules engine examines the relationship between a resulting event and a newly connected action to determine if it is appropriate to provide any of the action parameters with a default value. This would be a value found somewhere in the rule chain before the insertion point of the new action. You may need to modify this default value.



## How to get the rule set to create final output

### Scenario

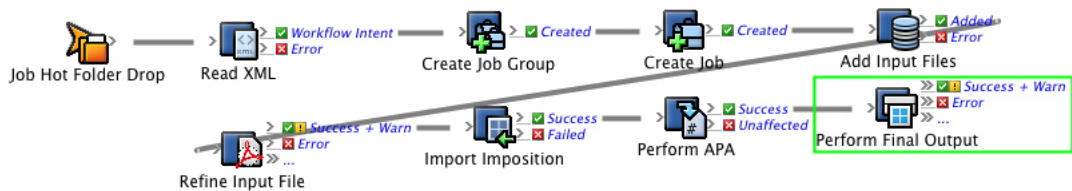
The rule set has been configured so that if an XML file is dropped into the hot folder, RBA will read the XML and automatically create a job group and job, while monitoring the actions for any exceptions. It will then automatically add input files to the job, refine the input files, import an imposition, and perform automatic page assignment.

Next, you need to add a step that will automatically create the final output.

Follow these procedures:

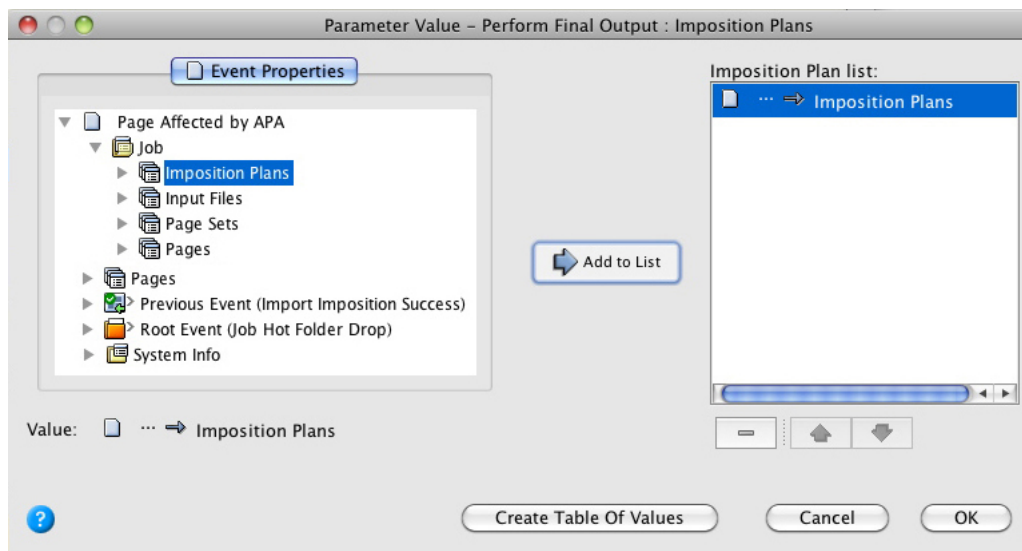
### Task 1: Get the rule set to create final output

1. Open a copy of the enabled rule set.
2. Drag a **Perform Final Output** action to the **Success** event from the **Perform APA** action.



3. Configure the **Imposition Plans** parameter to **Page Affected by APA > Job > Imposition Plans**.

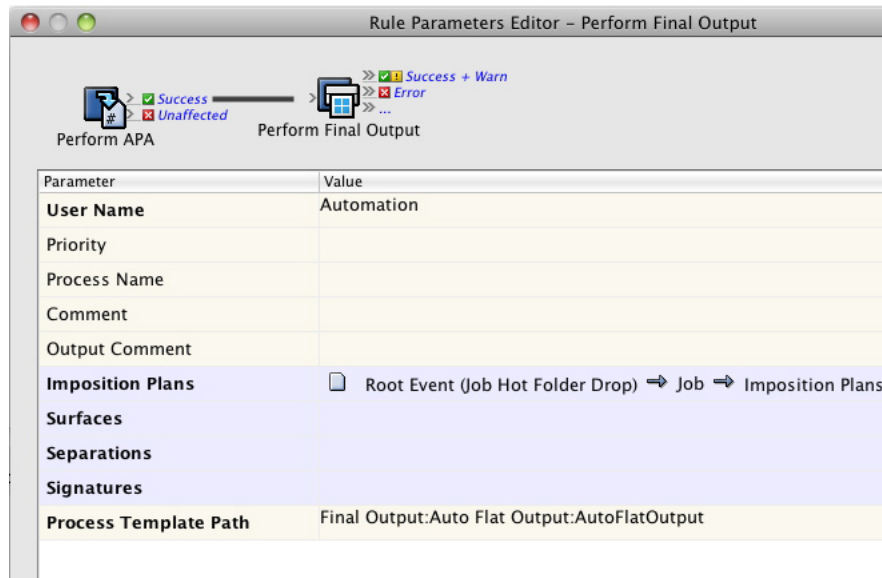
This tells RBA to output all the impositions in the job.



**Note:** If you are unable to edit the **Imposition Plans** parameter, check if one of the other parameters (**Surfaces**, **Separations**, or **Signatures**) has been populated. These four parameters are mutually exclusive, so only one of them can be given a value. If any of these parameters have been automatically populated, delete the value and the **Imposition Plans** parameter will become available again.

4. Configure the **Process Template Path** parameter to **AutoFlatOutput**.

The Rule Parameters Editor should look like this:



5. Save and hot swap the rule set with the one that is currently enabled.
6. Run the enabled rule set in the Debugger.



## Review what you know

### Answer or consider the following:

1. Why are the four parameters **Imposition Plans, Surfaces, Separations, and Signatures** mutually exclusive in the **Perform Final Output** action?
2. Under what circumstances is it necessary to provide a parameter value for only one of these imposition parameters?
3. Under what circumstances is it valid to provide a parameter value for all of these imposition parameters?
4. Name two ways to identify a set of mutually exclusive action parameters.
5. Would this rule set need to change for different XML files?

*Answers to review questions are located in the Appendix section of this guide.*



# Use temporary variables to simplify event references

## Overview



### Why you should complete this activity

This activity demonstrates how to use temporary variables to simplify complex event reference chains that refer back to previous events. By assigning values to temporary variables they can be directly referenced later in the rule chain.



### Target audience

Prepress operators with some RBA experience



### Time required

Approximately 25-30 minutes



### What you'll learn

After completing this activity, you will know how to perform the following tasks:

- Create temporary variables
- Assign the values from the XML file to the temporary variables
- Use the temporary variables as action parameters



### Recommended Reading

- *Prinergy Connect Workflow 6.1 Rules-Based Automation User Guide*
- *Prinergy Powerpack Workflow 6.1 Rules-Based Automation User Guide*

## What you need to know

In this activity, you will create and use temporary variables. For more information about variables, see the Variables chapter in the *Prinergy Connect Workflow 6.1 Rules-Based Automation User Guide*.

### What is a variable?

Variables allow for the storage and later retrieval of data within rule sets.

### Types of variables

There are three types of variables available:

- Global variables—shared by and accessible from all rule sets
- Rule set variables—shared by and accessible from all instances of a single rule set
- Temporary variables—exist within a rule set instance and therefore are accessible only from that instance

### What are the variable data types?

Type	Description
Boolean	A true or false value
Date and Time	A date and time representing an instant in time
Double	A 64-bit floating point number
File	A file or directory, along with an indication of exactly what type of file system object this represents
GUID	A globally unique identifier
Integer	A 32-bit integer
String	A character string
Time Span	An interval of time
Web File	A file or directory, along with an indication of exactly what type of file system object this represents.
Boolean list	A list of booleans
Date and Time list	A list of date and time objects
Double list	A list of Doubles



Type	Description
File List	A list of Files
Files (Web)	A list of Web Files
GUID list	A list of GUIDs
Integer list	A list of Integers
String list	A list of strings
Time Span list	A list of time spans

### What are the possible uses of each variable type?

Variable type	Possible uses
Global	<ul style="list-style-type: none"><li>• Isolating external references</li><li>• Persisting processing or execution state</li></ul>
Rule Set	<ul style="list-style-type: none"><li>• Isolating external references</li><li>• Persisting processing or execution state</li></ul>
Temporary	<ul style="list-style-type: none"><li>• Simplifying complex event property references</li><li>• Persisting processing or execution state</li></ul>



## How to create, assign, and use temporary variables

### Scenario

The XML workflow rule set that you have just created uses a number of values out of the XML file as parameters to various actions in the rule chain. To use the values within an action, the parameters must be retrieved from the **XML Data** object in the **XML Parsed OK** event that resulted from the **Read XML** action. For actions farther down the rule chain such as the **Perform APA** action, the value for the APA instructions parameter requires navigating through several previous event references to get back to the **XML Parsed OK** event, resulting in a long and cumbersome event property reference such as:

APA Instructions → Previous Event (Input File Refined Success And Warn) → Previous Event (Input File Added) → Previous Event (Job Created) → Previous Event (XML Parsed OK) → XML Data (Xml Activity) → APACommand

Temporary variables can be used to simplify these property references. Instead of referencing the values through the previous event chain, the values can be assigned to temporary variables immediately after the **Read XML** action completes and then later in the rule chain the values can be referenced directly.

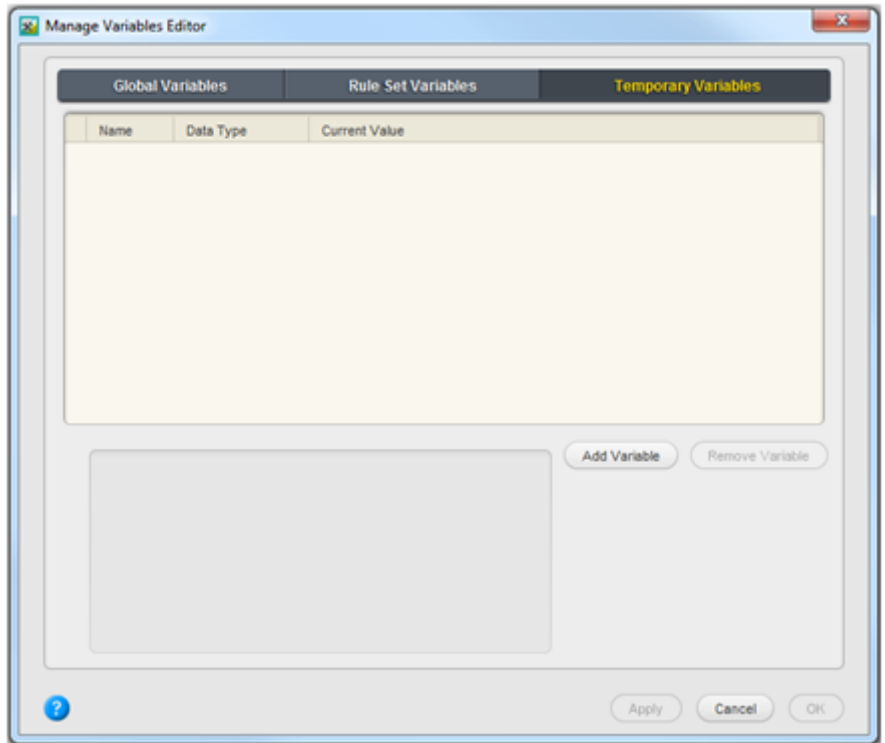
In this activity, we're going to create, assign, and use temporary variables.

Follow these procedures:

### Task 1: Create temporary variables

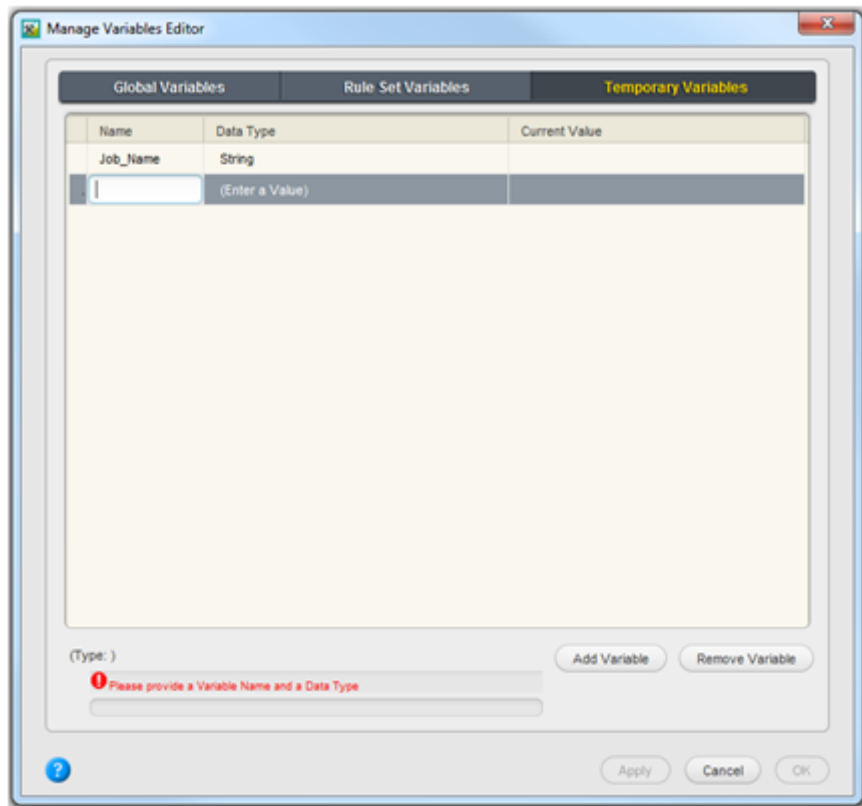
1. Open a copy of the enabled rule set.
2. In Rule Builder, from the **Edit** menu, select **Manage Variables**.
3. In the message that appears that prompts you to save the rule set, click **Yes** and type a new name for this copy of the rule set.

4. In the Manage Variables Editor window that appears, click the **Temporary Variables** tab.



5. Click the **Add Variable** button.
6. In the new row that appears, in the **Name** box, type `Job_Name`.
7. In the **Data Type** list, select **String**.
8. To confirm the current values and create the next variable, click the **Add Variable** button.

The Manage Variables Editor window, should look as follows:



9. Repeat steps 1-7 to create the following additional variables:

Name	Data type
Job_Home_Server	String
Job_Home_Share	String
Job_Group	String
Imposition_Path	File
APA_Command	String

**Note:** The observant reader will note that we are creating a variable for each element of interest in the XML file except for the **Input Files** list. No variables is being created because currently **File** or **String** lists variable cannot be assigned to File [ ] action parameters due to the issue PRINERGY-33919.

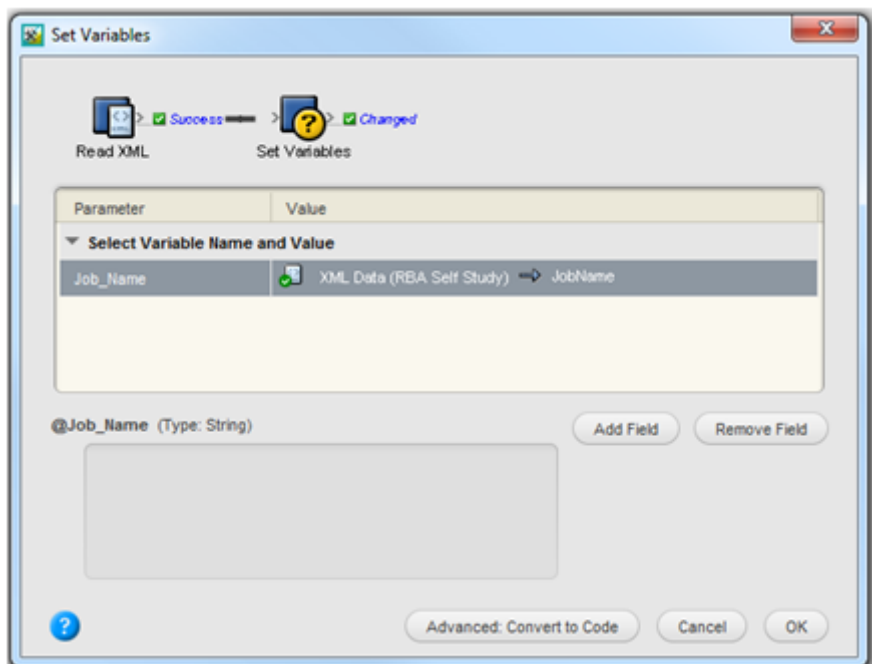
10. When you complete to define the required variables, click the **OK** button to confirm the creation of the variables and to close the Manage Variable Editor window.

## Task 2: Assign the values from the XML file to the temporary variables

1. In Rule Builder, on the **Action** tab, under **System Actions**, locate the **Set Variables** action.
2. Drag the **Set Variables** action onto the link between the **Read XML** and the **Create Job Group** actions. It may be necessary to rearrange the rule graph layout at this point.



3. To open the parameter editor and define the action parameters, double-click the **Set Variables** action.
4. (Optional) Resize the **Parameter** column.
5. In the **Parameter** column, click **%triggerEvent%**. Then in the list that appears, expand the **Temporary Variables** group and select the **Job\_Name - String** variable.
6. In the **Value** column, at the same row, click the **Edit Value** button (...) to access the Parameter Value dialog box.
7. Click the **Event Properties** tab.
8. Expand the **XML Data (RBA Self Study)** node.
9. Select the **JobName** element and click **OK**.  
The **Job\_Name** variable should now be configured to receive the value of the **JobName** element from the **XML Data** object.



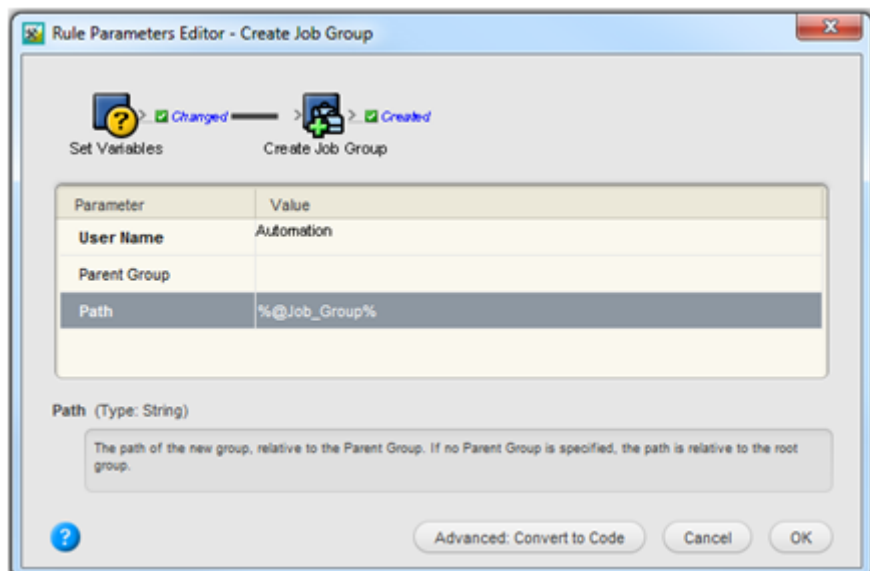
10. To add another new variable, click the **Add Field** button.
11. Repeat steps 5-10 for each of the following variables:

Variable	XML data element
Job_Group	JobGroup
Job_Home_Server	JobHomeServer
Job_Home_Share	JobHomeShare
Imposition_Path	ImpositionPath
APA_Command	APACommand

12. To close the Set Variables window, click **OK**.  
The line separating the event and the action should be grey, indicating that the action parameters were properly configured.

### Task 3: Use the temporary variables as action parameters

1. In the rule chain, double-click the **Create Job Group** action.
2. In Rule Parameters Editor, under **Parameter**, select the **Path** parameter
3. In the **Value** column, at the same row, click the **Edit Value** button (...) to access the Parameter Value dialog box.
4. Click the **Variables** tab.
5. Expand the **Temporary group** node, and from the list that appears, select the **Job\_Group** variable.
6. To close the editor and insert the variable reference as the **Path** parameter, click **OK**.  
The reference that appears is **%@Job\_Group%**.



7. To confirm the change and close the Rule Parameter Editor, click **OK**.
8. Repeat steps 1-7 for each of the following variables:

Action	Parameter	Variable
Create Job	New Job Name	Job_Name
	Job Home Server	Job_Home_Server
	Job Home Share	Job_Home_Share
Import Imposition	Imposition File	Imposition_Path
Perform APA	APA Instructions	APA_Command

9. Close and hot swap the rule set with the one that is currently enabled
10. Destroy the previous test job and resubmit the XML file to the hot folder.  
The rule set should still function exactly the same as before except now the values retrieved are passing through the temporary variables before being passed to the actions as parameters. By using direct references to temporary variables, the extremely long and cumbersome relative event references to event properties are avoided.



## Review what you know

### Answer or consider the following:

1. What would happen if rule set variables were used instead of temporary variables to store the elements from the **XML Data** object?
  
2. What is another use for a temporary variable?

*Answers to review questions are located in the Appendix section of this guide.*



## Create a table of values

---

### Overview



#### Why you should complete this activity

This activity demonstrates how to set conditions, so that RBA will perform a different action, depending on which condition has been fulfilled.



#### Target audience

Prepress operators with some RBA experience



#### Time required

Approximately 25-30 minutes



#### What you'll learn

After completing this activity, you will know how to perform the following tasks:

- Create an email action for output failure
- Create an email action for output success, with a table of values for the To parameter
- Set the Otherwise parameter



#### Recommended Reading

- *Prinerly Connect Workflow 6.1 Rules-Based Automation User Guide*
- *Prinerly Powerpack Workflow 6.1 Rules-Based Automation User Guide*



## How to create a table of values

### Scenario

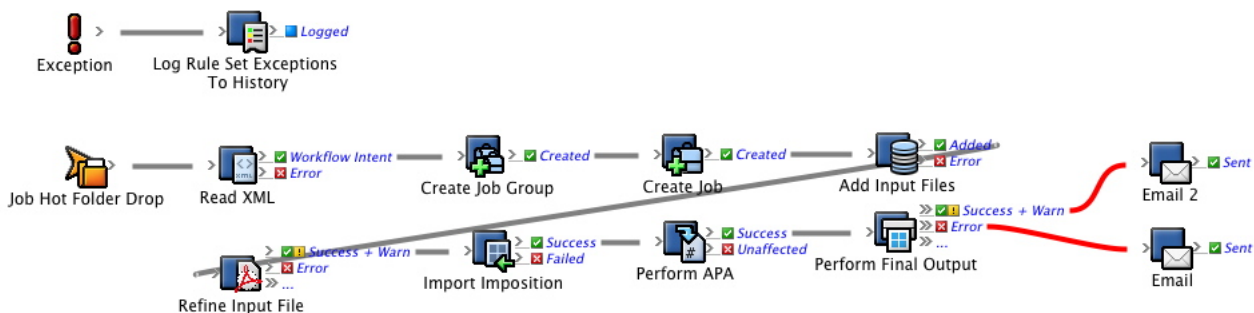
Now that the rule set has been configured to perform a workflow whenever an XML file is dropped into the hot folder, you want RBA to send an email message describing the success or failure of the workflow.

However, in your company, there are different team leaders, each of whom will need to be informed about different types of work. You want RBA to send an email to different people under different circumstances. So you need to set up conditions controlling when the email needs to be sent to Bob, and when it should be sent to Joe, etc. You will do this by creating a table of values for the **To** parameter on the email action.

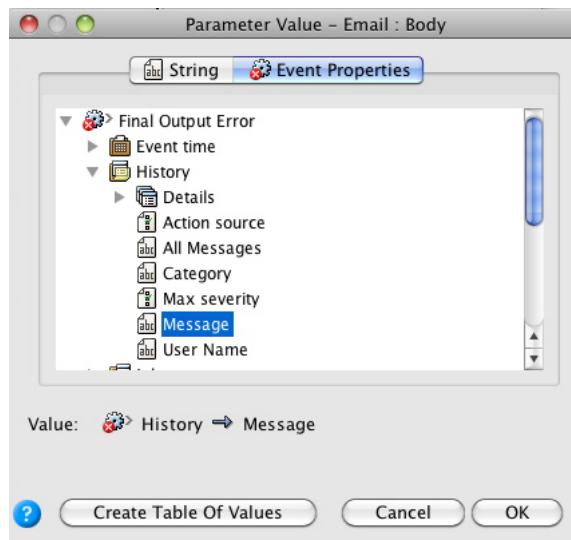
Follow these procedures:

### Task 1: Create an email action for output failure

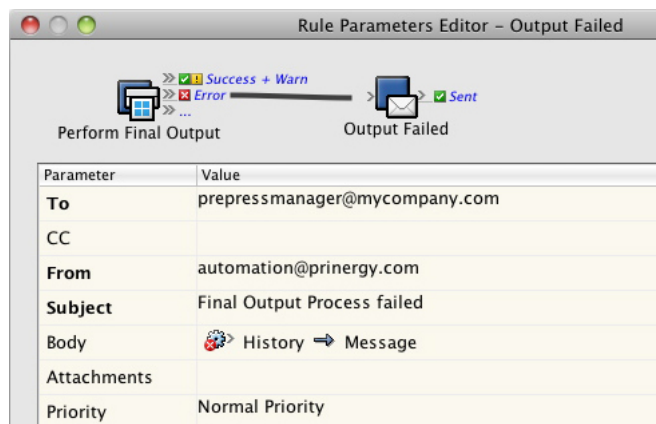
1. Open a copy of the enabled rule set.
2. Drag an **Email** action to both the **Success + Warn** and the **Error** generated events of the **Perform Final Output** action.



3. Rename the **Email** action on the **Error** event so that it has a meaningful name, like **Output Failed**.
4. Configure the **To** parameter, by typing an appropriate email address, such as `prepressmanager@mycompany.com`.
5. Configure the **Subject** parameter by typing `Final output process failed`.
6. Configure the **Body** parameter to **History > Message**. This tells RBA to get the reason for output failure from the Process History message and to copy it into the body of the email.



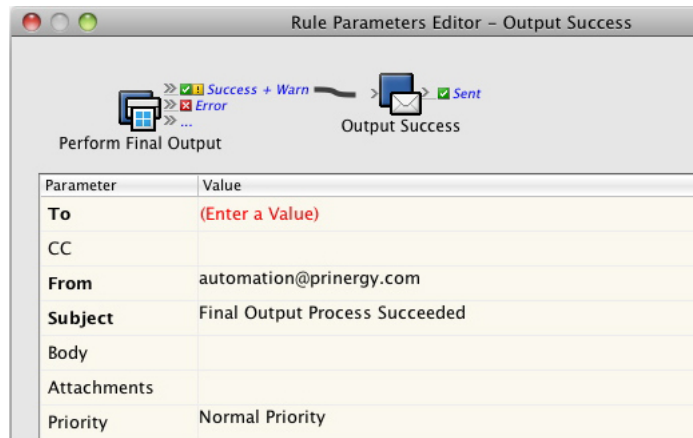
The Rule Parameters Editor should look like this:



Task 2: Create an email action for output success, with a table of values for the To parameter

1. Rename the second **Email** action, on the **Success + Warn** event, to **Output Success**.

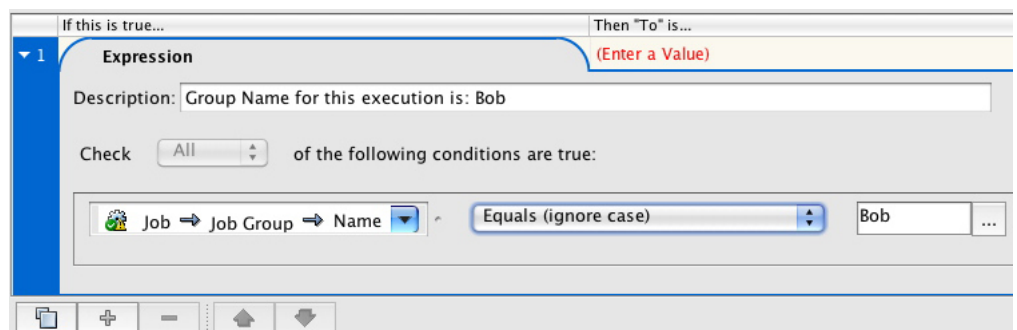
- Configure the **Subject** parameter by entering `Final output process successful`.



- In the **To** parameter configuration dialog, click the **Create Table Of Values** button. The Table of values configuration dialog appears.

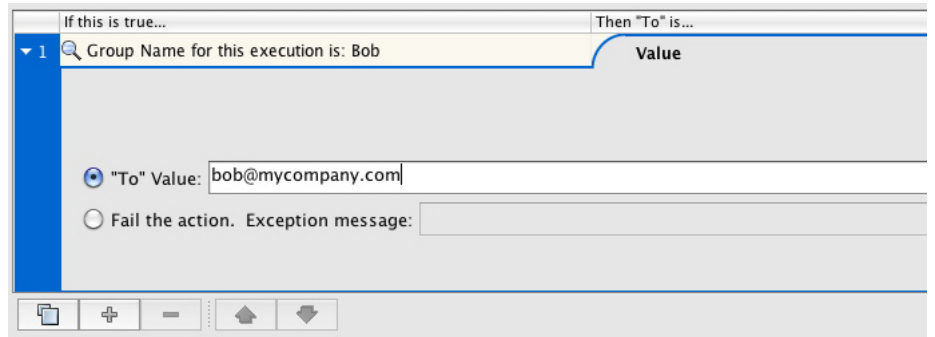
	If this is true...	Then "To" is...
1	(Enter an Expression)	(Enter a Value)
	Otherwise (if none of the above):	(Enter a Value)

- On the **If this is true** tab, double-click **Enter an Expression** to expand the configuration for Case 1.  
**Note:** Alternatively, you can single-click the right-pointing arrow near the number 1.
- In the **Description** parameter, enter a meaningful description, such as `Group name for this execution is Bob`.
- Create a criterion that compares the job group name to "Bob":
  - In the property list, select **Job > Job Group > Name**.
  - For the operator, select **Contains (ignore case)**.
  - For the value, enter `Bob`.



- On the **Then "To" is** tab, select **Enter a Value**.

- Enter an appropriate email address, such as bob@mycompany.com.

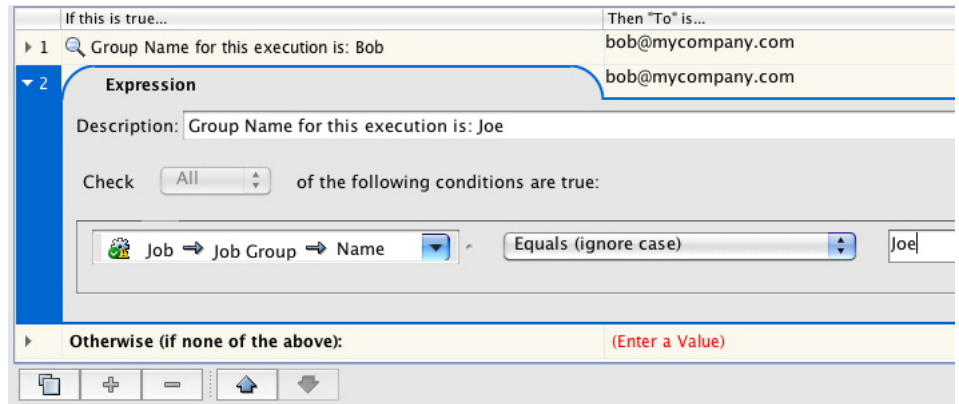


- Locate and click the **Duplicate Entry** button



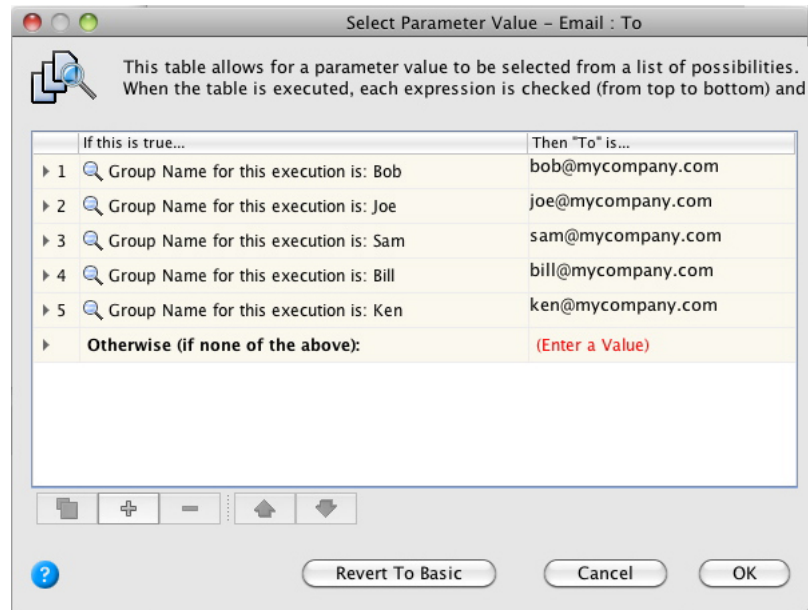
at the lower left corner of the dialog box.

- Modify the configuration of Case 2, by changing the **Description** parameter and the criterion, so that Joe appears instead of Bob.



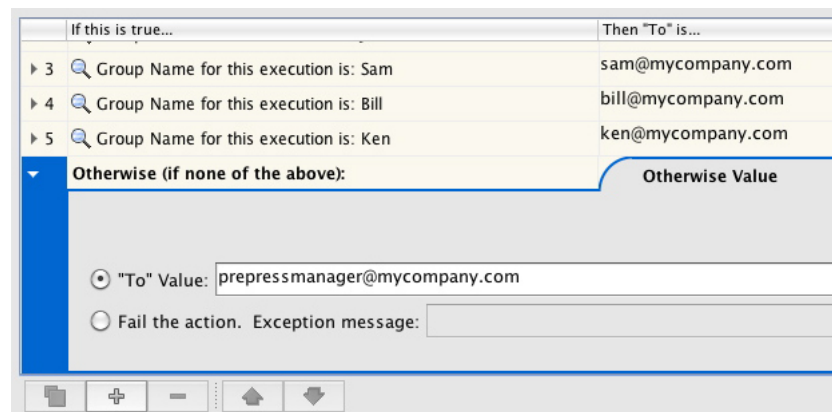
- On the **Then "To" is** tab, select **Enter a Value** and change bob@mycompany.com to joe@mycompany.com.

- Repeat the steps to add some more cases: Sam, Bill, and Ken.

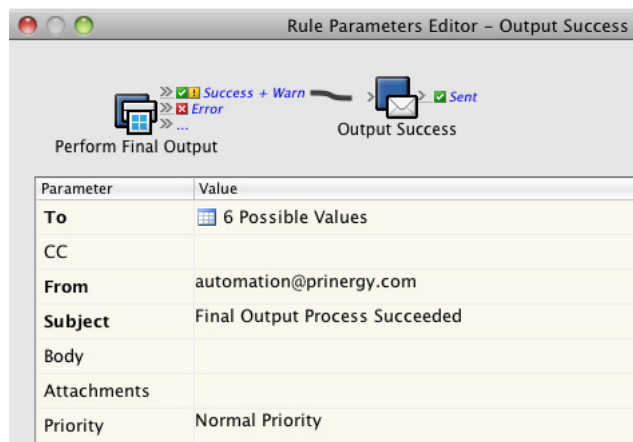


### Task 3: Set the Otherwise parameter

- Scroll down and expand the **Otherwise (if none of the above)** case. This is to accommodate the case where the group name is not one of the expected values.
- Configure the **To** parameter by entering `prepressmanager@mycompany.com`



The Rule Parameters Editor should look like this:



3. Save and hot swap the rule set with the one that is currently enabled.
4. Run the enabled rule set in the Debugger.



## Review what you know

### **Answer or consider the following:**

1. Where would you use a table of values?
2. Can a table of values be used more than once in an action?
3. At what point would a table of values make more sense than a series of branches?

*Answers to review questions are located in the Appendix section of this guide.*



## Set up remote triggers

---

### Overview



#### Why you should complete this activity

This activity demonstrates how to set up a remote trigger, in order to get one RBA rule set to be triggered via another "remote" rule set.



#### Target audience

Prepress operators with some RBA experience



#### Time required

Approximately 30-45 minutes



#### What you'll learn

After completing this activity, you will know how to perform the following tasks:

- Add a remote trigger in the receiver rule set
- Update the XML file
- Test receiver rule set
- Create a remote sender event
- Trigger the receiving rule set using the sending rule set



#### Recommended Reading

- *Prinerly Connect Workflow 6.1 Rules-Based Automation User Guide*
- *Prinerly Powerpack Workflow 6.1 Rules-Based Automation User Guide*

## What you need to know

### Remote trigger

A rule set with a remote trigger is one that can be triggered by another rule set in the same or different job, or from a DOS command line using `raiseevent.exe`.

For example, you can create a 'sender' rule set that is configured to call a 'receiver' rule set in another job. The sender rule set provides a path to an XML file that the receiver will consume.

Circumstances under which it would make sense to use a remote trigger include:

- Interaction with a third-party, non-Prinergy system via the `RaiseEvent.exe` utility that ships with every Prinergy system. This is usually an MIS or Insite StoreFront system.
- Hub and spoke systems where the hub instructs the spoke (via a remote trigger) to start a workflow. This is common where spoke sites output plates only and the prepress preparation work is done on the hub.
- In advanced cases, a remote trigger can be used as a subroutine-type call within a rule set. This is a very advanced use and would not be common in the field.



## How to set up a remote trigger

### Scenario

In this activity, you will take the existing XML-processing rule set and change the way it gets its intent. In the existing rule set, when an XML file is dropped into a job hot folder, RBA then processes that intent in various actions along a chain.

In this new scenario, RBA will get that same intent via a different method. The basic structure and performance of the rule set remains the same, but the way it gets the XML file is different.

In one job, you will create a 'sender' rule set that will be configured to call a 'receiver' rule set in another job. The sender will provide a path to the XML file that the receiver will consume via its **Send Remote Trigger** action.

So, when you run the sender rule set, it will call the receiver and pass it the appropriate values (via parameter values and strings). The receiver will then act on those event values. The receiver rule set is exactly the same as the one you created in activities 1-12, except that instead of a hot folder drop trigger event, it will start with a remote trigger event.

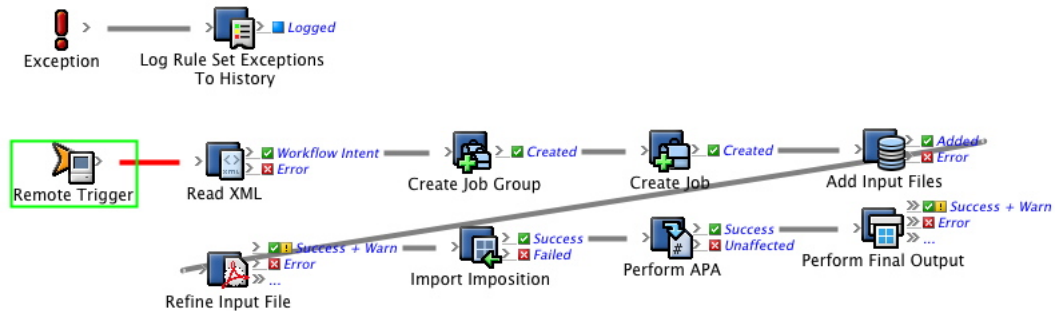
In this activity, you will work with four different jobs:

- **<XX>\_RBA\_Job** - This is where the XML file `Self-Study.xml` is stored.
- **<XX> Remote\_Job** - This is the resulting job of the XML workflow.
- **<XX> Hub\_Job** - This is where **<XX>\_Rule\_Set\_XML\_Remote\_Sender** is enabled.
- **<XX> Spoke\_Job** - This is where **<XX>\_Rule\_Set\_XML\_Remote\_Receiver** is enabled.

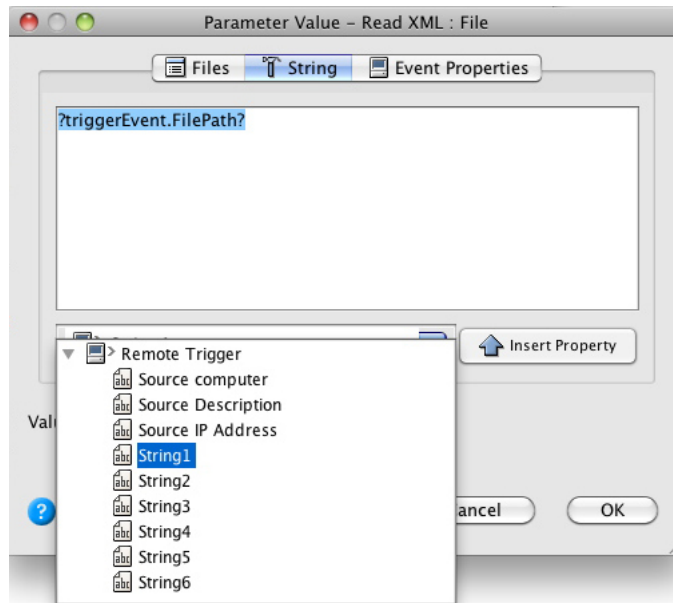
**Important:** Please note that RBA requires precision in the names of jobs and rule sets. If there are spelling mistakes in any of these names, for example, if an underscore is missing, the rule set will not work. Please pay close attention to the spelling of the jobs and rule sets in this activity.

### Task 1: Add a remote trigger in the receiver rule set

1. Disable **<XX>\_Rule\_Set\_XML\_Workflow** and make a copy of it called **<XX>\_Rule\_Set\_Remote\_Receiver** (where <XX> represents your initials).
2. Create two new jobs in **XX\_Rule\_Set\_Group**:
  - **<XX>\_Hub\_Job** - this is where **<XX>\_Rule\_Set\_Remote\_Sender** will be enabled
  - **<XX>\_Spoke\_Job** - this is where **<XX>\_Rule\_Set\_Remote\_Receiver** will be enabled
3. Enable **<XX>\_Rule\_Set\_Remote\_Receiver** in **<XX>\_RBA\_Job**.
4. Open a copy of the rule set for editing.
5. From the **Events** tab in Rule Set Builder, drag the **Remote Trigger** event over the top of the **Job Hot Folder Drop** event on the canvas.



6. Re-configure the **Read XML** action. Change the value of the **File** parameter to now be the **String1** event property from the **Remote Trigger** event  
 This tells RBA to get the XML file from the resulting event of the remote trigger (String1). We are going to populate **String1** in the sender rule set, in the next task.



7. Enable **<XX>\_Rule\_Set\_Remote\_Receiver** in **<XX>\_Spoke\_Job**.

## Task 2: Update the XML file

1. Open the XML file `Self-Study.XML` in a text editor.
2. Modify the XML file so that it points to the correct location of your jobs.

Before:

```
<?xml version="1.0" encoding="utf-8"?>
<RBASelfStudy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <JobName>XX_Remote_Job</JobName>
  <JobHomeShare>AraxiVolume_HOSTNAME_DRIVELETTER</JobHomeShare>
  <JobHomeServer>HOSTNAME</JobHomeServer>
  <JobGroup>XX_Group</JobGroup>
  <ImpositionPath>\\HOSTNAME\AraxiVolume_HOSTNAME_DRIVELETTER\RESOURCEDIRECTORY\AustraliaTour.pjtf</ImpositionPath>
  <APACCommand>ASSIGN="Aus Tours[#start]-[#end].p[#page].pdf" "*" [#start]+[#page]-1 1</APACCommand>
  <InputFiles>
    <string>\\HOSTNAME\AraxiVolume_HOSTNAME_DRIVELETTER\RESOURCEDIRECTORY\Aus Tours1-16.ps</string>
    <string>\\HOSTNAME\AraxiVolume_HOSTNAME_DRIVELETTER\RESOURCEDIRECTORY\Aus Tours17-32.ps</string>
    <string>\\HOSTNAME\AraxiVolume_HOSTNAME_DRIVELETTER\RESOURCEDIRECTORY\Aus Tours33-48.ps</string>
  </InputFiles>
</RBASelfStudy>
```

After:

```
<?xml version="1.0" encoding="utf-8"?>
<RBASelfStudy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <JobName>KB_Remote_Job</JobName>
  <JobHomeShare>AraxiVolume_RBA_J</JobHomeShare>
  <JobHomeServer>RBA</JobHomeServer>
  <JobGroup>KB_Group</JobGroup>
  <ImpositionPath>\\RBA\AraxiVolume_RBA_J\Jobs\KB_RBA_J\UserDefinedFolders\Input\Imposition Plans\AustraliaTour.pjtf</ImpositionPath>
  <APACCommand>ASSIGN="Aus Tours[#start]-[#end].p[#page].pdf" "*" [#start]+[#page]-1 1</APACCommand>
  <InputFiles>
    <string>\\RBA\AraxiVolume_RBA_J\Jobs\KB_RBA_J\UserDefinedFolders\Input\Input Files\Aus Tours1-16.ps</string>
    <string>\\RBA\AraxiVolume_RBA_J\Jobs\KB_RBA_J\UserDefinedFolders\Input\Input Files\Aus Tours17-32.ps</string>
    <string>\\RBA\AraxiVolume_RBA_J\Jobs\KB_RBA_J\UserDefinedFolders\Input\Input Files\Aus Tours33-48.ps</string>
  </InputFiles>
</RBASelfStudy>
```

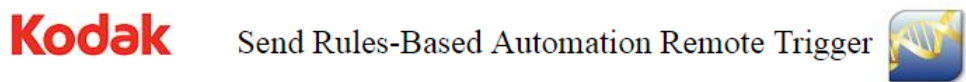
**Note:** Make sure that you use accurate spelling and that the paths are correct. Note that in the screenshot, the initials KB are used as an example, but in your case, use your own initials.

### Task 3: Test receiver rule set

1. In a Web browser, navigate to the Send Rules-Based Automation Remote Trigger page at `http://<primary server name or IP>:61235/ExtAutomation/RaiseExternalEventImp/SendRemoteTrigger.htm`

**Note:** Before you use the above URL, you must replace *<primary server name or IP>* with the actual name of your primary server.

2. Enter the parameters required to activate the receiver rule set:
  - **Rule Set Path:** `<XX>_RBA_Self-Study/<XX>_Rule_Set_Remote_Receiver`
  - **Environment:** `<XX>_Spoke_Job`
  - **String 1:** Enter the full path to the XML (intent) file that you have saved on your system.



Enter the parameters for a rule set with a Remote Trigger root event then click **Send Remote Trigger** button.

The servers response will be displayed in the **Status & Response** area.

Rule Set Path:

Environment:

String 1:

String 2:

String 3:

String 4:

String 5:

String 6:

Files:

Status & Response:

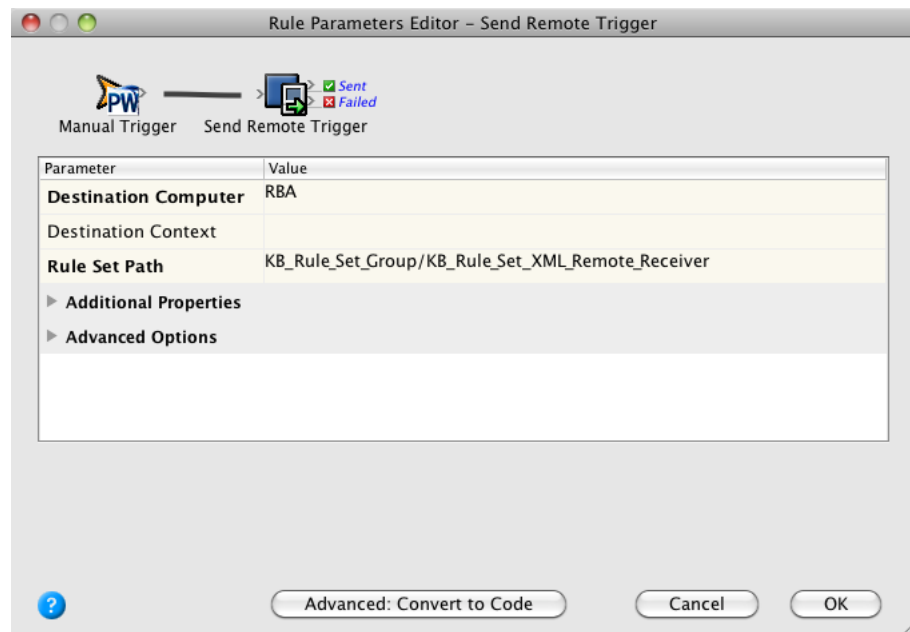
3. Activate the receiver rule set by clicking the **Send Remote Trigger** button.  
The **Status & Response** pane should show a status code of 200 indicating the send completed OK. If so, watch the process in Job Finder. Look for evidence of your group being created, followed by the job. See if you find VPS files in the `Proofs` sub-folder under the `XX_Remote_Job`.
4. Destroy or rename the newly created job in preparation for the next task. The job will be created again in the next task and if this instance is not destroyed or renamed first it will conflict with the next task.

#### Task 4: Create a remote sender event

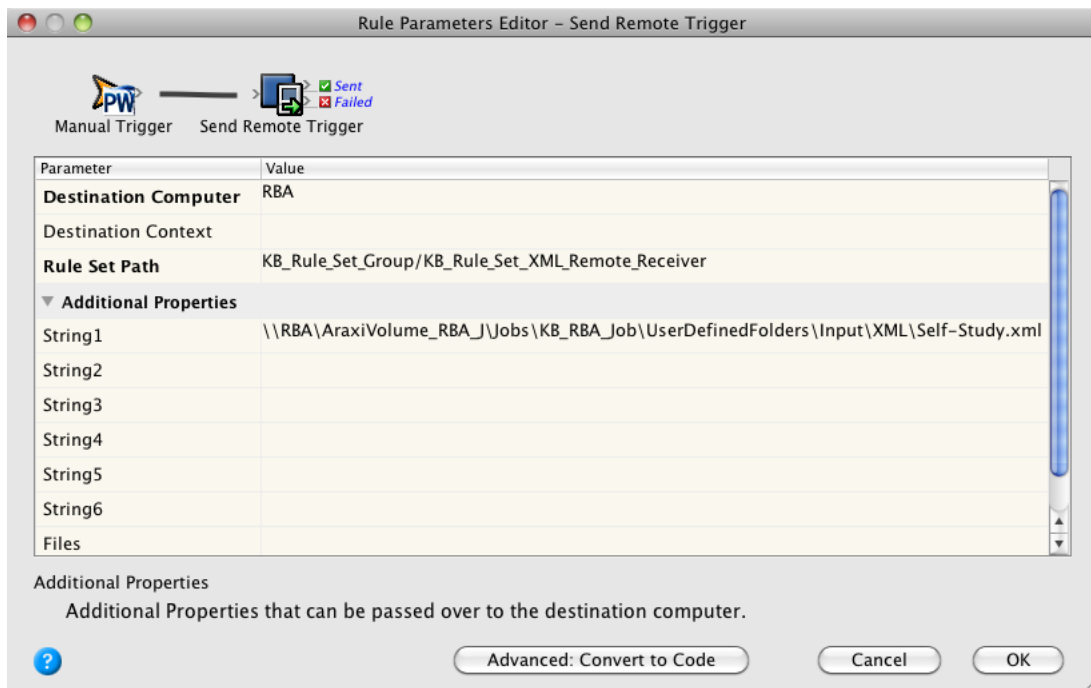
1. Create a new rule set called **<XX>\_Rule\_Set\_XML\_Remote\_Sender** (where <XX> represents your initials) in **<XX>\_Hub\_Job**.
2. Add an exception handler rule, like you did in the previous rule set.
3. Add a **Manual Trigger** action.
4. Add a **Send Remote Trigger** action to the resulting event of the **Manual Trigger** action.



5. Configure the **Rule Set Path** parameter to your **<XX>\_Rule\_Set\_Group > <XX>\_Rule\_Set\_XML\_Remote\_Receiver**.  
This path must be accurate for the trigger to work.

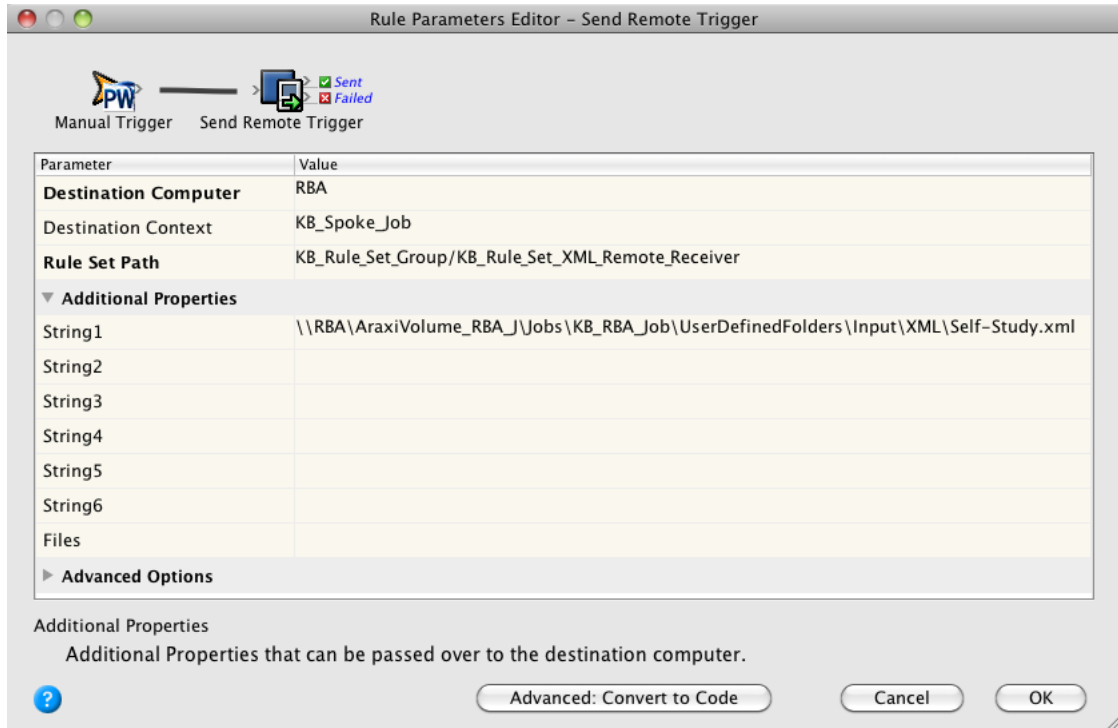


- Configure the **String1** parameter by entering the full path to the XML (intent) file that you have saved on your system.





- Configure the **Destination Context** parameter by entering your spoke job name, **<XX>\_Spoke\_Job**.



- Enable **<XX>\_Rule\_Set\_XML\_Remote\_Sender** in **<XX>\_Hub\_Job**.

### Task 5: Trigger the receiving rule set using the sending rule set

- Trigger **<XX>\_Rule\_Set\_XML\_Remote\_Sender**. Select the **Job** icon in the lower left corner of Workshop, right-click and select the **Enabled Rules** menu option.
- Run both the enabled rule sets in the Debugger.
- Run the activity by manual trigger in **XX\_Hub\_Job** and watch the process in Job Finder. Look for evidence of your group being created, followed by the job. See if you find VPS files in the `Proofs` sub-folder under the `XX_Remote_Job` folder.



## Review what you know

### **Answer or consider the following:**

1. Where would you use a pair of rule sets that relied on remote triggers?
2. How many string arguments can you provide to a remote trigger?
3. Is it possible to send a remote trigger to a rule set from outside of Prinerger?

*Answers to review questions are located in the Appendix section of this guide.*

## Get the rule set to populate custom fields

---

### Overview



#### Why you should complete this activity

This activity builds on the previous Remote Trigger activity and adds Dashboard-like functionality with the use of Prinerger custom fields. As the rule set runs, custom fields are populated to display information about the progress and output of the workflow.



#### Target audience

Prepress operators with some RBA experience



#### Time required

Approximately 30-45 minutes



#### What you'll learn

After completing this activity, you will know how to perform the following tasks:

- Create the required Prinerger custom fields
- Modify a copy of the remote sender rule set to call the new remote receiver rule set
- Modify a copy of the remote receiver rule set to use custom fields
- Run the rule set



#### Recommended Reading

- *Prinerger Connect Workflow 6.1 Rules-Based Automation User Guide*
- *Prinerger Powerpack Workflow 6.1 Rules-Based Automation User Guide*

## What you need to know

### **Custom fields**

Prinergy custom fields are stored in arrays.

This means that each custom field you define at whatever level and for whatever type can/will have a unique value across each of your jobs. For example, if you create a job-level string-type custom field and you made it visible, you could set two different values in two different jobs.



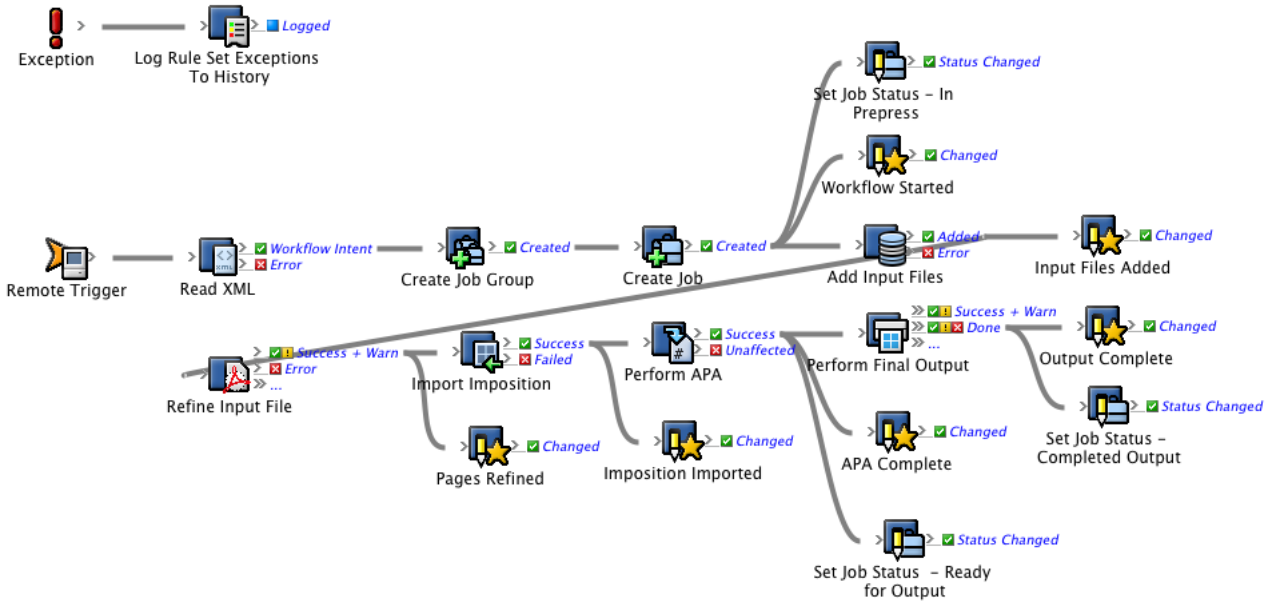
# How to get the rule set to populate custom fields

## Scenario

You want the rule set to populate custom fields to display information about the progress and output of the workflow.

In this activity, the custom fields are used in a read-only capacity, but these fields can be used to provide information to a rule set as well.

This is what the completed rule set will look like.



Follow these procedures to complete this scenario.

### Task 1: Create the required Prinergy custom fields

1. From the **Tools > Custom Fields Manager** menu in Workshop, create the required Prinergy custom fields (as shown in the screenshot).

Name	Default Value	Type	Show in Workshop
SSG_AddInput	<input type="checkbox"/>	Yes/No	<input checked="" type="checkbox"/>
SSG_APA	<input type="checkbox"/>	Yes/No	<input checked="" type="checkbox"/>
SSG_Imposition	<input type="checkbox"/>	Yes/No	<input checked="" type="checkbox"/>
SSG_ImpositionName		Text	<input checked="" type="checkbox"/>
SSG_Output	<input type="checkbox"/>	Yes/No	<input checked="" type="checkbox"/>
SSG_PagesRefined		Integer	<input checked="" type="checkbox"/>
SSG_Refine	<input type="checkbox"/>	Yes/No	<input checked="" type="checkbox"/>
SSG_SigToOutput		Text	<input checked="" type="checkbox"/>
SSG_Start	<input type="checkbox"/>	Yes/No	<input checked="" type="checkbox"/>

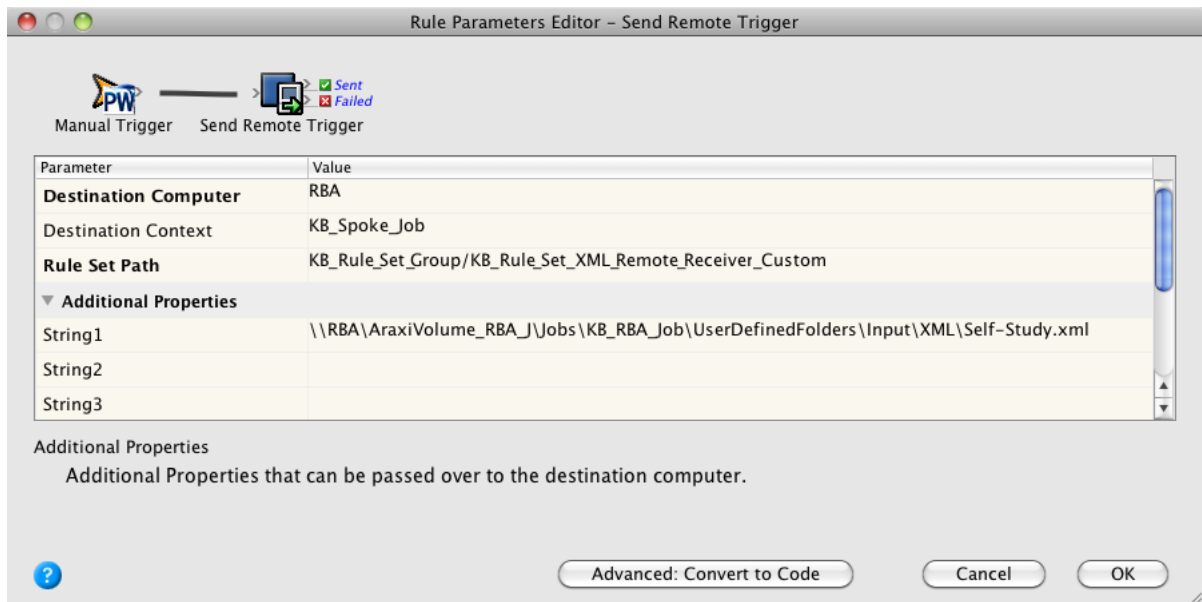
Ensure that these fields are created with the specified types and at the appropriate level (job).

2. Verify that the check boxes in the **Show in Workshop** column are selected.  
This will make these custom fields visible in Workshop.

### Task 2: Modify a copy of the remote sender rule set to call the new remote receiver rule set

1. Create a copy of the rule set **<XX>\_Rule\_Set\_XML\_Remote\_Sender** and name it **<XX>\_Rule\_Set\_XML\_Remote\_Sender\_Custom**.

2. Modify this rule set so that it calls the rule set **<XX>\_Rule\_Set\_XML\_Remote\_Receiver\_Custom**.



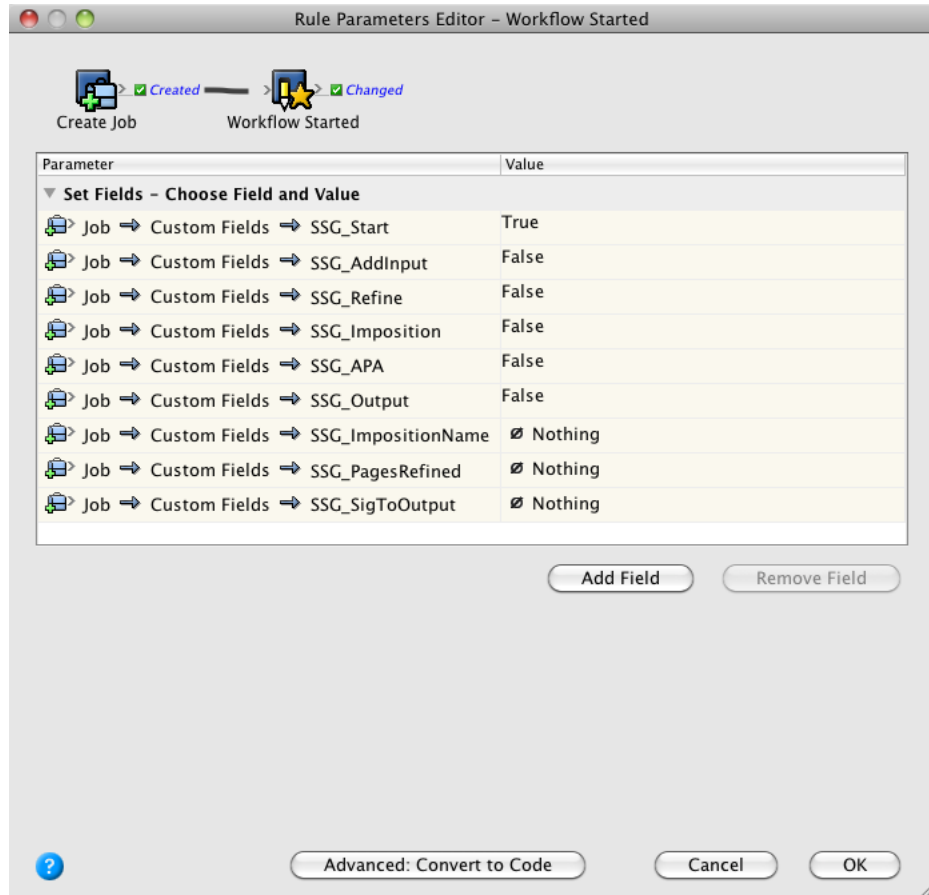
3. Save and enable your new sender rule set **<XX>\_Rule\_Set\_XML\_Remote\_Sender\_Custom** in the hub job, named **<XX>\_Hub\_Job**.

**Task 3: Modify a copy of the remote receiver rule set to use custom fields**

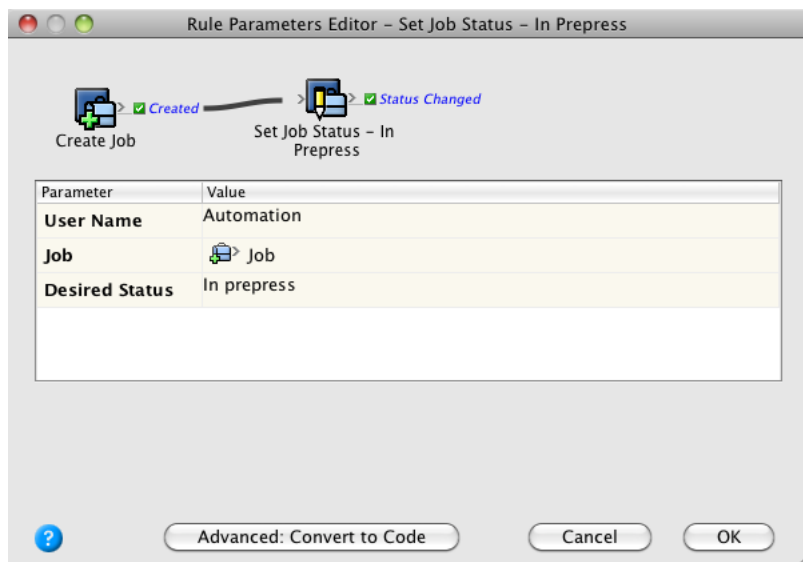
Use the complete rule set screen capture as a guide for the **Set Custom Field** and **Set Job Status** actions in this section.

1. Create a copy of the rule set **<XX>\_Rule\_Set\_XML\_Remote\_Receiver** and name it **<XX>\_Rule\_Set\_XML\_Remote\_Receiver\_Custom**.

2. Add a **Set Custom Field** action to the **Created** resulting event of the **Create Job** action. Use this to initialize all the custom fields that this rule set uses.

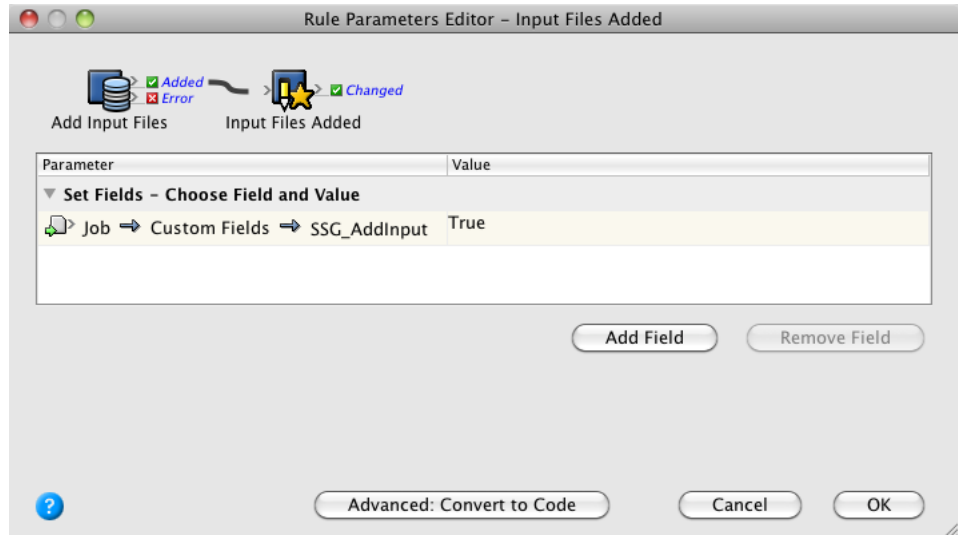


3. Add a **Set Job Status** action to the **Created** resulting event of the **Create Job** action to alter the job status.

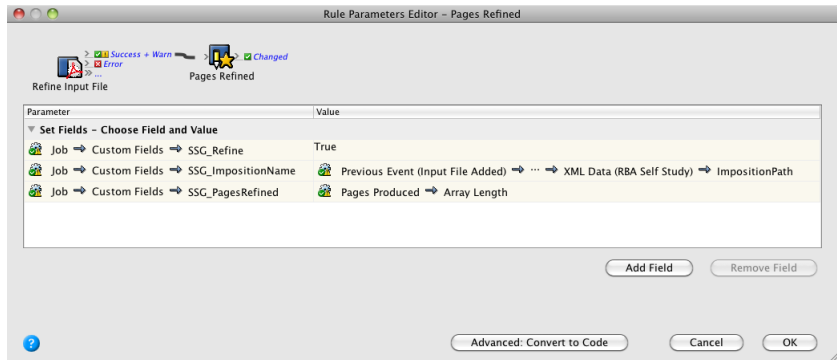




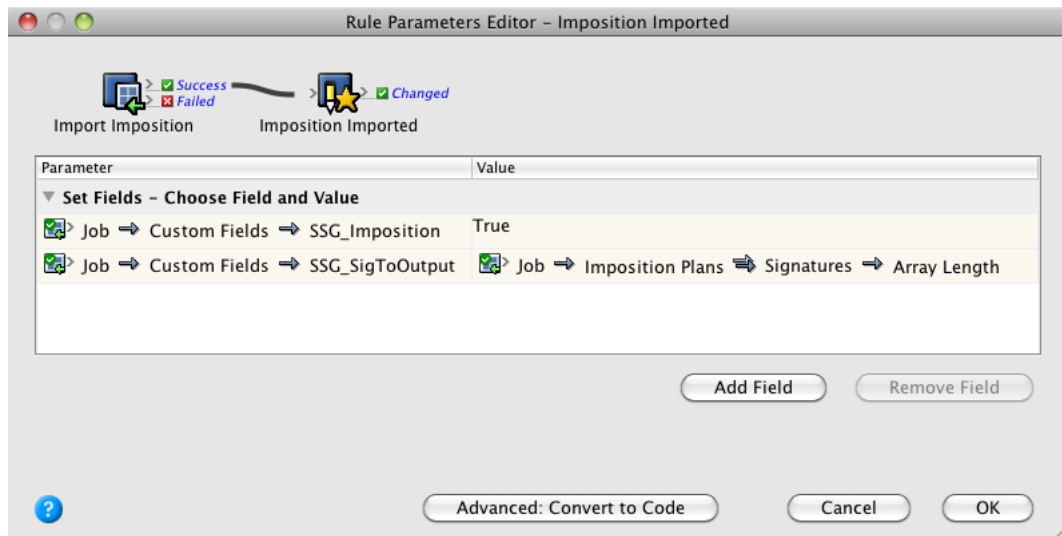
4. Add a **Set Custom Field** action to the **Added** resulting event of the **Add Input Files** action. Configure as shown.



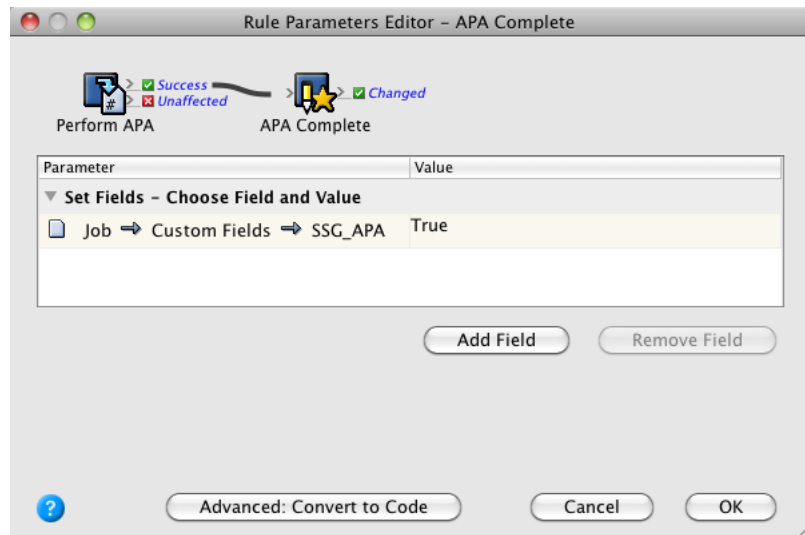
5. Add a **Set Custom Field** action to the **Success + Warn** resulting event of the **Refine Input File** action. Configure as shown.



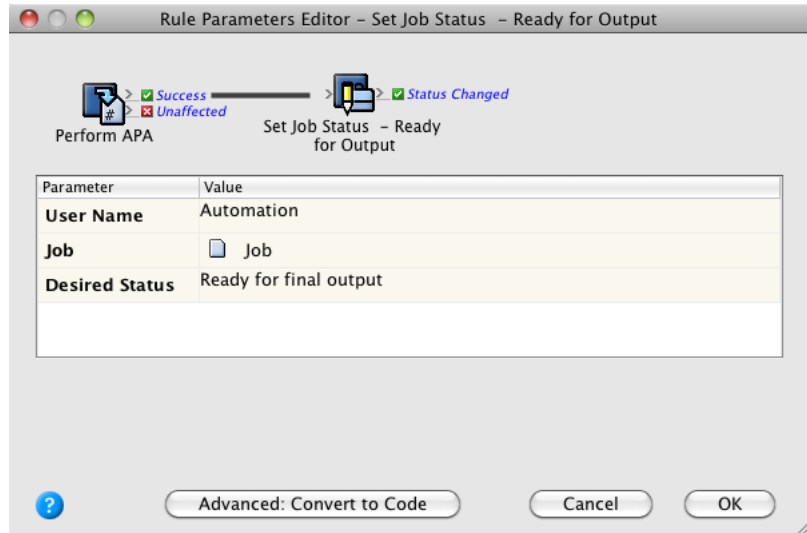
6. Add a **Set Custom Field** action to the **Success** resulting event of the **Import Imposition** action. Configure as shown.



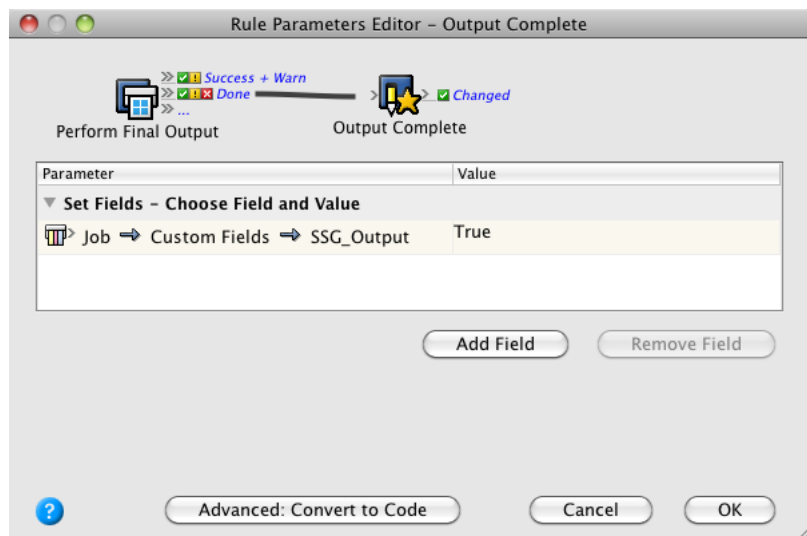
7. Add a **Set Custom Field** action to the **Success** resulting event of the **Perform APA** action. Configure as shown.



8. Add a **Set Job Status** action to the **Success** resulting event of the **Perform APA** action. Set the desired status to **Ready for final output**.



9. Add a **Set Custom Field** action to the **Done** resulting event of the **Perform Final Output** action. Configure as shown.



10. Save and enable the new receiver rule set in the spoke job, named **<XX>\_Spoke\_Job**.

## Task 4: Run the rule set

1. In Job Finder, use the **View > Visible Columns** option to show all of the SSG custom fields that you created, and arrange them into a workflow order as shown below.

Name	Job Status	SSG_Start	SSG_AddInput	SSG_Refine	SSG_PagesRefined	SSG_Imposition	SSG_ImpositionName	SSG_APA	SSG_Output	SSG_SigToOutput
KB_Group										
KB_Remote_Job	Created	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		<input type="checkbox"/>		<input type="checkbox"/>	<input type="checkbox"/>	
KB_Main_Group										
KB_Hub_Job	In Prepress	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		<input type="checkbox"/>		<input type="checkbox"/>	<input type="checkbox"/>	
KB_RBA_Job	In Prepress	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		<input type="checkbox"/>		<input type="checkbox"/>	<input type="checkbox"/>	
KB_Spoke_Job	In Prepress	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		<input type="checkbox"/>		<input type="checkbox"/>	<input type="checkbox"/>	

2. From your hub job, **<XX>\_Hub\_Job**, start the sender rule set **<XX>\_Rule\_Set\_XML\_Remote\_Sender\_Custom** and watch the workflow progression in Job Finder.

Name	Job Status	SSG_Start	SSG_AddInput	SSG_Refine	SSG_PagesRefined	SSG_Imposition	SSG_ImpositionName	SSG_APA	SSG_Output	SSG_SigToOutput
KB_Group										
KB_Remote_Job	Completed Final Output	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	48	<input checked="" type="checkbox"/>	\\RBA\AraxiVolum...	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	3
KB_Main_Group										
KB_Hub_Job	In Prepress	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		<input type="checkbox"/>		<input type="checkbox"/>	<input type="checkbox"/>	
KB_RBA_Job	In Prepress	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		<input type="checkbox"/>		<input type="checkbox"/>	<input type="checkbox"/>	
KB_Spoke_Job	In Prepress	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		<input type="checkbox"/>		<input type="checkbox"/>	<input type="checkbox"/>	



## Review what you know

### **Answer or consider the following:**

1. Name the available custom field types.
  
2. Name the available custom field scopes (the levels at which a custom field can be defined).
  
3. Why can each custom field at each different level of a job have a unique value?

*Answers to review questions are located in the Appendix section of this guide.*



## Locate and rename files in a folder

---

### Overview



#### Why you should complete this activity

This activity demonstrates how to use the **List Directory** and **Rename One File** actions to identify and correct files that do not conform to a specific naming standard.



#### Target audience

Prepress operators with some RBA experience



#### Time required

Approximately 25-30 minutes



#### What you'll learn

After completing this activity, you will know how to perform the following tasks:

- Create a new rule set that triggers when the job status changes to In Prepress
- Locate all PDF files in the UserDefinedFolder
- Prepend the job name to each file if it is missing
- Add the files to the job and test the rule set



#### Recommended Reading

- *Prinerly Connect Workflow 6.1 Rules-Based Automation User Guide*
- *Prinerly Powerpack Workflow 6.1 Rules-Based Automation User Guide*



## Locate and rename files in a folder

### Scenario

You want that all the input files will have the job name prepend to them. When the job status changes to **In Prepress**, you will rename the PDF files in the job's `UserDefinedFolder` before adding them to the job.

### Task 1: Create a new rule set that triggers when the job status changes to In Prepress

1. In Rule Set Manager, create a new rule set.
2. From the **Events** tab, drag the **Job Status Changed** event to the canvas.
3. From the **Flow** tab, drag the **Branch** action over the **Job Status Changed** event resulting action.
4. Set the **Branch** action's condition to `Job -> Status Is In Prepress`.

### Task 2: Locate all PDF files in the UserDefinedFolder

1. From the **Actions** tab, drag a **List Directory** action to the success event of the **Branch** action.
2. Edit the **List Directory** action's parameters as follows:
  - a. Set the **Directory** parameter to the job's `UserDefinedFolders`. You can copy the following path to the **Value** box:

```
%triggerEvent.Previous.Job.Home%\UserDefinedFolders
```
  - b. For the **Recurse** parameter in the **Value** list, select **True**.
  - c. For the **Pattern** parameter, in the **Value** box, type `*.pdf`

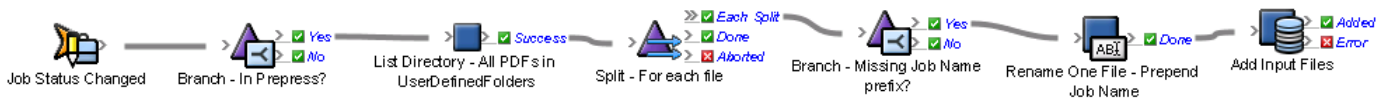


### Task 3: Prepend the job name to each file if it is missing

1. From the **Flow** tab, drag a **Split** action over the **Success** event of the **List Directory** action.
2. From the **Flow** tab, drag a **Branch** action over the **Each Split** event of the **Split** action.
3. Set the **Branch** action's condition to `Split (File) -> File Name Does Not Begin With %triggerEvent.RootEvent.Job.Name%`
4. From the **Actions** tab, drag a **Rename One File** action over the **Yes** event of the **Branch** action.
5. Edit the **Rename One File** action's parameters:
  - a. For the **File to rename** parameter, in the **Value** box, type `Previous Event (Each Item Group) -> Split (File)`
  - b. For the **New name** parameter, in the **Value** box, type the job name and file name separated by an underscore or copy the following: `%triggerEvent.Previous.RootEvent.Job.Name %_%triggerEvent.Previous.ItemGroup.FileName%`

### Task 4: Add the files to the job and test the rule set

1. From the **Actions** tab, drag the **Add Input Files** action over the **Done** event of the **Rename One File** action.  
The final rule set should look like the following:



2. Enable the rule set in the system environment.
3. Create a new job.
4. Add a number of PDF files to its `UserDefinedFolders` folder.
5. Change the job status to **In Prepress**.  
The files should be added to the job. Each should have the job name prepended to it.



## Review what you know

### Answer or consider the following:

1. What if the `UserDefinedFolders` folder contained sub-folders. Would the files in the sub- directories get renamed and added to the job as well?

2. What if it was a directory that was to be renamed. What would have to change in the **Rename One File** action?

*Answers to review questions are located in the Appendix section of this guide.*

## Update a daily production record with the names of all pages approved that day

### Overview



#### Why you should complete this activity

This activity demonstrates how to safely access a shared resource from multiple instances of a rule set.



#### Target audience

Prepress operators with some RBA experience



#### Time required

Approximately 25-30 minutes



#### What you'll learn

After completing this activity, you will know how to perform the following tasks:

- Create a new rule set and define the rule set variables
- Configure the rule set to write each page approval to the production record
- Create the initial CSV file on the disk
- Test the rule set



#### Recommended Reading

- *Prinerly Connect Workflow 6.1 Rules-Based Automation User Guide*
- *Prinerly Powerpack Workflow 6.1 Rules-Based Automation User Guide*

### What you need to know

The **Limit Concurrency** action is used to synchronize access to a shared resource. The synchronization can be across multiple instances of a

single rule set or across multiple rule sets. Access can be restricted to a single instance or limited number of instances at once. For a complete explanation of the action see the *Prinergy Workflow RBA User Guide*.



## Update a daily production record with the names of all pages approved that day

### Scenario

You need to maintain a record of all pages approved during the day. The record should include the job and page name, the user who approved it, and the time it was approved. The record should be stored as a CSV file.

The **Write Text To File** action can be used to produce a CSV file. Assuming it is fine for the names of the pages from different jobs to be interleaved, then just using the **Write Text To File** action would work most of the time. It will work most of the time because the **Write Text To File** action contains the retry logic that detects that the file is locked and retries to write multiple times. Assuming there are not too many instances that are trying to update the file at once this will work fine. However, if too many instance attempt to write to the file, some will be unable to access the file and fail.

To make sure that the writes always succeed, you can place the **Limit Concurrency** action in the rule chain. Thus, the number of instances that run concurrently can be controlled.

### Task 1: Create a new rule set and define the rule set variables

1. In Rule Set Manager, create a new rule set.
2. Save the new rule set as `Report Page Approvals` in your rule set group.
3. From the **Edit** menu in Rule Builder, select **Manage Variables**.
4. In the Manage Variables Editor make sure that the **Global Variables** tab is selected.  
A global variable is used because the variable is going to be referenced from a second rule set in the next activity. If it was only going to be used only in this rule set, a rule set variable would be sufficient.
5. Click the **Add Variable** button.
6. In the **Name** column, type `Page_Approval_Record_File`.
7. In the **Data Type** column, select **File**.
8. In the **Current Value** column, type a fully qualified UNC path to a file called `PageApprovalRecord.csv`.

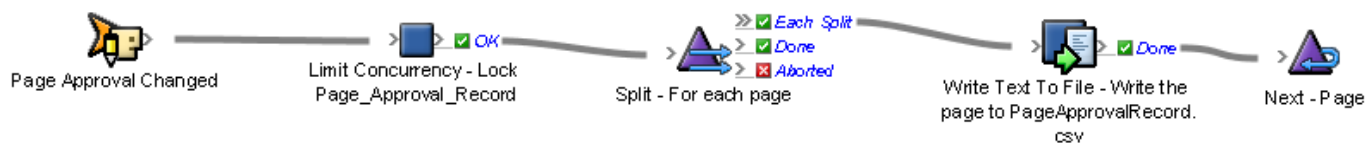
For example, if the name of your Prinergy server is `MyServer` and you want to place the file in the `Jobs` folder on the `J:` drive, you need to type the following path:

```
\\MyServer\AraxiVolume_MyServer_J\Jobs  
\PageApprovalRecord.csv
```

9. To save the changes and close the Manage Variables Editor, click **OK**.

## Task 2: Configure the rule set to write each page approval to the production record

In this task, you will learn to build the following rule set:



1. In Rule Builder, on the **Events** tab, in the **Prinergy Events > Page Events** group, locate an **Page Approval Changed** event and drag it to the canvas.
2. On the **Flow** tab, locate the **Limit Concurrency** action and drag it to the resulting action of the **Page Approval Changed** event.
3. Double-click the **Limit Concurrency** action and edit the parameters as follows:
  - a. For the **Concurrent Limit** parameter, in the **Value** column, type 1.
  - b. For the **Limit Name** parameter, in the **Value** column, type `Page_Approval_Record_File`.
  - c. Click **OK**.
4. On the **Flow** tab, locate the **Split** action and drag it to the **OK** event of the **Limit Concurrency** action.
5. Double-click the **Split** action and edit the parameters as follows:
  - a. For the **Items** parameter, in the **Value** column, from the list, select **Previous Event (Page Approved) > Pages**.
  - b. For the **Items Per Group** parameter, in the **Value** column, type 1.
  - c. For the **Per Group Delay** parameter, in the **Value** column, select **1 minute** and click **OK**.
  - d. Click **OK**.
6. On the **Actions** tab, locate the **Write Text To File** action and drag it to the **Each Split** event of the **Split** action.

7. Double-click the **Write Text To File** action and edit the parameters as follows:
  - a. For the **File to write to** parameter, in the **Value** column, click the **browse** button and navigate to `Global Variable` folder. In the folder, select the **Page\_Approval\_Record\_File** variable.
  - b. For the **Text** parameter, in the **Value** column, type a string that contains the job name, page name, new approval state, event time, and user name all separated by commas.  
For example, the final parameter should look like:

```
%triggerEvent.RootEvent.Job.Name%,  
%triggerEvent.ItemGroup.Name%,  
%triggerEvent.RootEvent.NewApprovalState%,  
%triggerEvent.RootEvent.EventTime%,  
%triggerEvent.RootEvent.History.UserName%
```

**Note:** The string must be typed in one single line with all the parameters separated by commas.
  - c. For the **Append** parameter, in the **Value** column, in the list, select **True**.
  - d. Click **OK**.
8. On the **Flow** tab, locate the **Next** action and drag it to the **Done** event of the **Write Text To File** action.
9. Double-click the **Next** action and edit the parameters as follows:
  - a. For the **Loop Start** parameter, in the **Value** column, from the list, select **Previous Event (Each item Group)**.
  - b. For the **Abort Loop** parameter, in the **Value** column, in the list, select **False**.
  - c. Click **OK**.
10. Save and close the rule set.

### Task 3: Create the initial CSV file on the disk

The next activity will read the CSV file using the **Read CSV** action. That action requires that column headers are included as the first row of the file.

1. On the primary server, open Notepad and paste the following text to define the column headers in the CSV file: `Job,File,State,Time,User`
2. Save the file as `PageApprovalRecord.csv` in the `J:\Jobs` folder.

### Task 4: Test the rule set

1. Enable the rule set in the system environment.
2. Approve and reject pages in a number of different jobs.
3. Open the `PageApprovalRecord.csv` file and confirm that there is a header row and each subsequent row contains the job name, page name, approval state, event time, and user name.





## Review what you know

### Answer or consider the following:

1. Why is the **Limit Concurrency** action used in this scenario?
2. What if there were number of additional actions to be performed after the record is updated?
3. What if you wanted to open and read the CSV file from another rule set?

*Answers to review questions are located in the Appendix section of this guide.*



# Send an e-mail with a summary of a daily production record

---

## Overview



### Why you should complete this activity

This activity demonstrates how to:

- Safely access a shared resource from multiple rule sets.
- Use the **Read CSV** action to read a CSV file.
- Use variables to tabulate a production report.
- Properly implement a rule set that can infinitely repeat.
- Delete a file.



### Target audience

Prepress operators with some RBA experience



### Time required

Approximately 25-30 minutes



### What you'll learn

After completing this activity, you will know how to perform the following tasks:

- Create the skeleton for an infinitely repeating rule set
- Use the production record name as the limiter to serialize access to the record
- Register a schema for the Approval Record CSV file
- Read the CSV file
- Use temporary variables to summarize the contents of the CSV file
- Send the report by e-mail
- Delete the record
- Recreate the CSV file with a header row
- Add a standalone exception handler
- Test the rule set



### **Recommended Reading**

- *Prinergy Connect Workflow 6.1 Rules-Based Automation User Guide*
- *Prinergy Powerpack Workflow 6.1 Rules-Based Automation User Guide*



## Send an e-mail with a summary of a daily production record

### Scenario

In the previous activity a production record was created. This activity will demonstrate how to send an e-mail message with a summary of the record on a daily basis and then reset the record. Access to the record will be synchronized between the update rule set and this reporting rule set.

### Task 1: Create the skeleton for an infinitely repeating rule set

An infinitely repeating rule set contains the following events:



The rule set is enabled, pauses for some time, then performs a series of actions before it is disabled, and then re-enables itself.

**Note:** Some type of timer is required to limit the frequency of rule set execution, otherwise it will place a heavy load on the system. In this activity, you will use a **Scheduled** timer, but you can also use a **Relative** timer.

**Note:** To stop an infinitely repeating rule set you must use the **Stop All Processing** option in Rule Set Manager. If you only disable the rule set, it will finish the rule chain and actually re-enable the rule set.

1. In Rule Set Manager, create a new rule set.
2. Save the new rule set as `Report Daily Kodak Approval Record` in your rule set group.
3. In Rule Builder, on the **Events** tab, locate the **This Rule Set Enabled** event and drag it to the canvas.
4. On the **Flow** tab, locate the **Timer (Scheduled)** action and drag it to the resulting action of the **This Rule Set Enabled** event.
5. Double-click the **Timer (Scheduled)** action and edit the parameters as follows:
  - a. In the **Scheduled Time** list, select an hour that is later than the current time.
  - b. In the **Schedule for a particular day of the week** list, select **Any Day**.
  - c. Click **OK**.
6. On the **Actions** tab, locate the **Disable Rule Set** action and drag it to the **Expired** event of the **Timer (Scheduled)** action.

7. Double-click the **Disable Rule Set** action and edit the parameters as follows:
  - a. For the **Rule Set** parameter, leave the **Value** column blank, so that the current rule set is disabled.
  - b. For the **Environment** parameter, in the **Value** column, from the list, select **System**.
  - c. Click **OK**.
8. On the **Actions** tab, locate the **Enable Rule Set** action and drag it to the **Disabled** event of the **Disable Rule Set** action.
9. Double-click the **Enable Rule Set** action and edit the parameters as follows:
  - a. For the **Rule Set** parameter, leave the **Value** column blank, so that the current rule set is enabled again.
  - b. For the **Environment** parameter, in the **Value** column, from the list, select **System**.
  - c. Click **OK**.

Now that you have created the skeleton for the rule set, in the next task, you will start to insert actions needed to perform the record processing.

## Task 2: Use the production record name as the limiter to serialize access to the record

1. On the **Flow** tab, locate the **Limit Concurrency** action and insert it between the **Timer (Scheduled)** and **Disable Rule Set** action by dragging it to the line that connects them.
2. Double-click the **Limit Concurrency** action and edit the parameters as follows
  - a. For the **Limiter Name** parameter, in the **Value** column, type `Page_Approval_Record_File`.
  - b. Leave the **Concurrent Limit** value as **1**, as we only want to allow one rule set instance access at a time.
  - c. Click **OK**.

### Task 3: Register a schema for the Approval Record CSV file

1. Open a text editor and paste the following XSD code into it:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="table">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="row" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="Job" type="xs:string"/>
              <xs:element name="File" type="xs:string"/>
              <xs:element name="State" type="xs:string"/>
              <xs:element name="Time" type="xs:string"/>
              <xs:element name="User" type="xs:string"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

2. Save the file as `Page Approval Record.xsd`.
3. In Rule Builder, from the **Tools** menu, select **RBA XML Schema Manager**.
4. In the **Actions** area, select **Add a Schema**.
5. Name the schema `Page Approval Record`.
6. Click the **Browse** button next to the **Schema File** box, and navigate to the `Page Approval Record.xsd` file that you have just created.
7. Click the **Add Schema** button.

### Task 4: Read the CSV file

1. Drag a **Read CSV** action onto the link between the **Limit Concurrency** action and the **Disable Rule Set** action.
2. Double-click the **Read CSV** action and edit the action's parameters as follows:
  - a. For the **Schema Name** parameter, in the **Value** column, in the list, select the **Page Approval Record** schema that was registered in the previous task.
  - b. For the **File** parameter, in the **Value** column, navigate to the **Global Variables** folder and select the global variable `Page_Approval_Record_File`.

The final rule chain up to the **Read CSV** action looks like this::



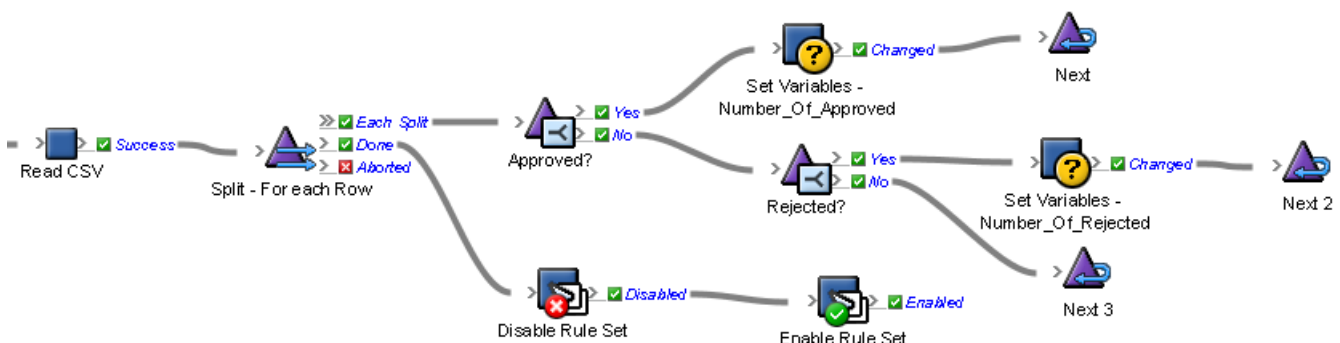
### Task 5: Use temporary variables to summarize the contents of the CSV file

To summarize the contents of the record for the last 24 hours, you will iterate over the rows in the CSV file and use a couple of the temporary variables to count the number of approvals and rejections.

1. In Rule Builder, from the **Edit** menu, select **Manage Variables**.
2. Click the **Temporary Variables** tab.
3. Click the **Add Variable** button.
4. In the new row that appears, in the **Name** box, type `Number_Of_Approved`.
5. In the **Data Type** list, select **Integer**.
6. Click the **Add Variable** button again. and
7. In the **Name** box, type `Number_Of_Rejected` and in the **Data Type** list, select **Integer**.
8. To confirm the creation of the variables and to close the Manage Variable Editor, click **OK**.
9. From the **Flow** tab, drag a **Split** action onto the link between the **Read CSV** action and the **Disable Rule Set** action.
10. Double-click the **Split** action and edit the action's parameters as follows:
  - a. For **Items** parameter, in the **Value** column, navigate to **CSV Parsed OK > CSV Data (Page Approval Record) > row**.
  - b. For **Per Group Delay** parameter, in the **Value** column, click the **Edit Value** button and in the **Minutes** list, select **1**.  
Such a long delay is fine, as a **Next** action is going to be appended to the end of the rule chain to ensure that each iteration of the split completes as quickly as possible.
  - c. Click **OK**.
11. In the **Split** action, disconnect the **Disable Rule Set** action from the **Each Split** event and connect it to the **Done** event by selecting the connecting link and dragging it from the **Each Split** event to the **Done** event.
12. In the **Changing Trigger Event** message that appears, click **OK**.
13. On the **Flow** tab, locate the **Branch** action and drag it to the **Each Split** event of the **Split** action.



14. Set the **Branch** action's condition to **Split (tableRow) > State Begins With Approved**.
  15. From the **Actions** tab, drag a **Set Variables** action over the **Yes** event of the **Branch** action.
  16. Double-click the **Set Variables** action.
  17. In the **Parameter** column, click **%triggerEvent%**. Then in the list that appears, expand the **Temporary Variables** group and select the **Number\_Of\_Approved** variable.
  18. Use the **Set Variables** action to increment the **Number\_Of\_Approved** temporary variables value. The following expression can be used to increment the value—it must be pasted directly into the **Number** tab of the Parameter Value editor: `1+ %@Number_Of_Approved%`.
  19. From the **Flow** tab, drag the **Next** action over the **Set Variables Changed** event resulting action.
  20. From the **Flow** tab, drag the **Branch** action over the **No** event of the existing **Branch** action.
  21. Set the **Branch** action's condition to **Split (tableRow) > State Begins With Rejected**.
  22. From the **Actions** tab, drag a **Set Variables** action over the **Yes** event of the **Branch** action.
  23. Double-click the **Set Variables** action.
  24. In the **Parameter** column, click **%triggerEvent%**. Then in the list that appears, expand the **Temporary Variables** group and select the **Number\_Of\_Rejected** variable.
  25. Use the **Set Variables** action to increment the **Number\_Of\_Rejected** temporary variables value. The following expression can be used to increment the value—it must be pasted directly into the **Number** tab of the Parameter Value editor: `1+ %@Number_Of_Rejected%`.
  26. From the **Flow** tab, drag the **Next** action over the **Set Variables Changed** event resulting action.
  27. From the **Flow** tab, drag another **Next** action over the **No** event of the second **Branch** action.
- The split loop of the rule chain should now look as follows:



## Task 6: Send the report by e-mail

Now that the record has been analyzed and the limiter released the next step is to send an e-mail with the record and a summary.

1. From the **Actions** tab, drag an **Email** action onto the link between the **Release Concurrency** action and the **Disable Rule Set** action.
2. Double-click the **Email** action and edit the action's parameters as follows:
  - a. For the **To** parameter, in the **Value** column, click the **Edit Value** button and navigate to `Global Variable` folder. In the folder, select the **prepress\_operators\_emails** global variable if you have one configured already. Otherwise, type your e-mail address.
  - b. For the **Subject** parameter, in the **Value** column, click **Enter a Value** and type (or paste) :  
Approval Report - Total:  
`%triggerEvent.Previous.CSVData.row.Length%,`  
Approved:  
`%@Number_Of_Approved%, Rejected:`  
`%@Number_Of_Rejected%`  

In the subject line of the e-mail, the two temporary variables will display a summary of the number of approved and reject approvals in the last 24 hours.
  - c. Attach the complete record to the e-mail; for the **Attachment** parameter, in the **Value** column, click the **Edit Value** button and navigate to `Global Variable` folder. In the folder, select the **Page\_Approval\_Record\_File** global variable, and click **OK**.
  - d. Click **OK**.

### Task 7: Delete the record

Once the report has been sent by e-mail, the record can be deleted in order to reset it for the next day's production.

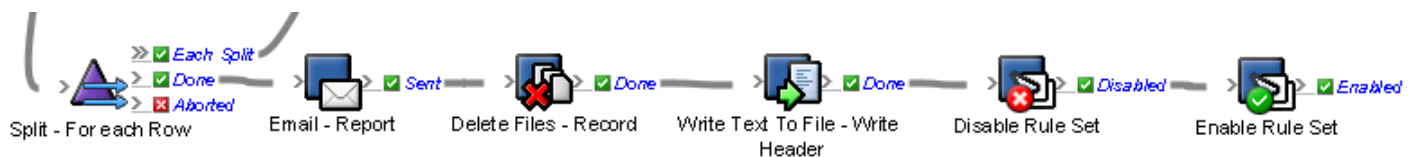
1. From the **Actions** tab, drag a **Delete Files** action onto the link between the **Email** action and the **Disable Rule Set** action.
2. Double-click the **Delete Files** action and edit the action's parameters as follows:
  - a. For the **Files to delete** parameter, in the **Value** column, click the **Edit Value** button and navigate to *Global Variable* folder. In the folder, select the **Page\_Approval\_Record\_File** global variable, and click **OK**.
  - b. Click **OK**.

### Task 8: Recreate the CSV file with a header row

To make sure that the next CSV file has a proper header row again, it will be created now.

1. From the **Actions** tab, drag a **Write Text To File** action onto the link between the **Delete Files** action and the **Disable Rule Set** action.
2. Double-click the **Write Text To File** action and edit the action's parameters as follows:
  - a. For the **File to write to** parameter, in the **Value** column, click the **Edit Value** button and navigate to *Global Variable* folder. In the folder, select the **Page\_Approval\_Record\_File** global variable.
  - b. For the **Subject** parameter, in the **Value** column, click **Enter a Value** and type *Job, File, State, Time, User*.
  - c. Click **OK**.

The rule chain after the **Split** action is completed, should look as follows:



### Task 9: Add a standalone exception handler

In the off chance that something would go wrong while the rule set is being performed, you want to send an e-mail to the site administrator and schedule the rule set again for the next day.

1. From the **Events** tab, drag an **Exception** event to the canvas.
2. From the **Actions** tab, drag an **Email** action to the **Exception** resulting event.
3. Double-click the **Email** action and edit the action's parameters as follows:
  - a. For the **To** parameter, in the **Value** column, click the **Edit Value** button and navigate to `Global Variable` folder. In the folder, select the `prepress_operators_emails` global variable if you have one configured already. Otherwise, type your e-mail address.
  - b. Set the **Subject** parameter to the name of the rule set and include the rule and message of the exception. For example, For the **Subject** parameter, in the **Value** column, click **Enter a Value** and type `Report Daily Approval Record failed. Rule: %triggerEvent.RuleName% Error: %triggerEvent.Message%`
  - c. Click **OK**.
4. From the **Actions** tab, drag a **Disable Rule Set** action to the **Sent** event of the **Email** action.
5. Double-click the **Disable Rule Set** action and edit the action's parameters as follows:
  - a. For the **Environment** parameter, in the **Value** column, in the list, select **System**.
  - b. Click **OK**.
6. From the **Actions** tab, drag a **Enable Rule Set** action to the **Disabled** event of the **Disable Rule Set** action.
7. Double-click the **Enable Rule Set** action and edit the action's parameters as follows:
  - a. For the **Environment** parameter, in the **Value** column, in the list, select **System**.
  - b. Click **OK**.

The completed exception handler should look as follows:



## Task 10: Test the rule set

Now it is time to test the rule set and make sure that it is working correctly. To avoid having to wait for the scheduled time the **Debugger** will be used to step through the **Timer (Scheduled)** action.

1. Save and close the rule set.
2. Enable the rule set in system environment.
3. In Rule Set Manager, select the rule set.
4. From the **File** menu, select the **Debug** option.
5. In Rule Debugger, in the **Rule Set Execution History** list, select the currently running instance, if it is not already selected.
6. To force the **Timer (Scheduled)** action to execute immediately, from the **File** menu, select the **Go** option.  
The rule set proceeds to the **Split** action where there is an automatic breakpoint.
7. To clear all the breakpoints, from the **File** menu, select **Remove All Breakpoints**.
8. To step through the rule set, from the **File** menu, select the **Step All** option repeatedly or select the **Go** option to have the rule set run to completion.
9. Confirm that the report e-mail is sent and that the content of the report file is re-initialized.

**Important:** To stop an infinitely repeating rule set you must use the **Stop All Processing** option in Rule Set Manager. If you only disable the rule set, it will not stop—the rule chain will be executed to its final step, which will result in it being re-enabled.

**Important:** If it appears as if a concurrency limiter is deadlocked because none of the **Limit Concurrency** action instances that are using that limiter are proceeding, perform the **Deadlock** procedure. For more information about the **Deadlock** procedure, see the *Limiting Rule Set concurrency* section in the *Prinerly Connect Workflow 6.1 Rules-Based Automation User Guide*.



## Review what you know

### **Answer or consider the following:**

1. What would be another way of implementing an infinitely repeating rule set?

*Answers to review questions are located in the Appendix section of this guide.*

## Actions and events of interest

---

### Overview



#### Why you should complete this activity

This activity discusses some additional events and actions that you might find useful.



#### Target audience

Prepress operators who have completed the earlier parts of this course.



#### Time required

Approximately 30-45 minutes



#### What you'll learn

After completing this activity, you will know how to perform the following tasks:

- Configure the Transfer Files action
- Create a rule set that uses Transfer Files
- Create a rule set with a generic Exception handler
- Create a rule set with a specific Exception handler
- Include the rule set name in the exception e-mail message
- Enable/disable hot folders
- Trigger events when a rule set is enabled/disabled
- Configure transfer files
- Use the Publish File Done event
- Use the Page Edit Done event



#### Recommended Reading

- *Prinerger Connect Workflow 6.1 Rules-Based Automation User Guide*
- *Prinerger Powerpack Workflow 6.1 Rules-Based Automation User Guide*



## How to use the Transfer Files action

The **Transfer Files** action is used to copy files from one server to another.

The real advantage in using this action (over other similar actions) is that it incorporates the Prineroy Insite file transfer technology, ensuring fast, reliable file transmission.

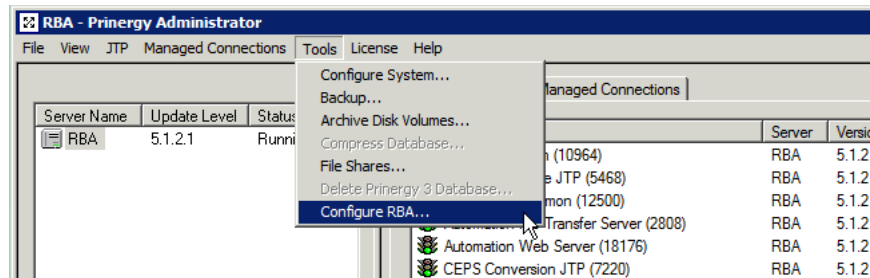
The action is based on the **Remote Trigger** action and therefore allows a rule set to run when the payload is delivered.

### Scenario

There are many scenarios that make sense for this action. It's useful for any situation that requires the reliable transfer of files from one server to another, especially if a rule set is required to run on receipt of the files.

### Task 1: Configure the Transfer Files action

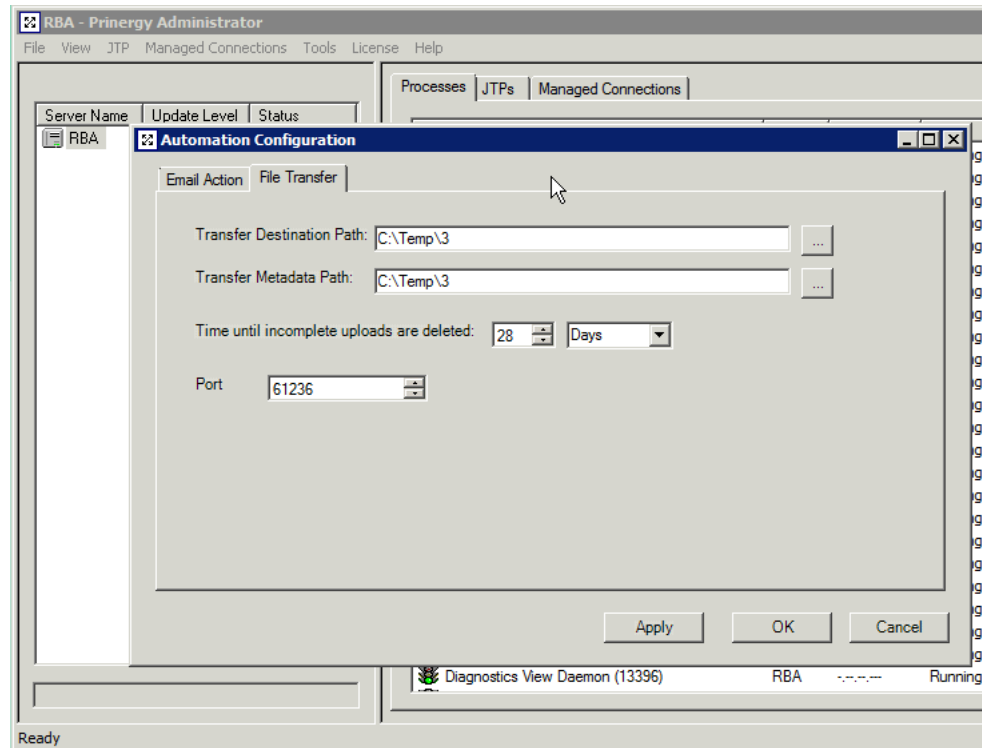
1. On the Prineroy primary server, open the Administrator.
2. Select **Tools > Configure RBA**.



The Automation Configuration dialog is displayed.



### 3. Select the **File Transfer** tab.



The system settings for the **Transfer Files** action are on this tab. There is no need to change anything in this dialog, as the defaults will work as supplied.

### 4. You can change these settings if required.

## Task 2: Create a rule set that uses Transfer Files

1. Create a new rule set, as shown in the screenshot. Remember to add an exception handler.

The screenshot displays a rule set configuration in a software interface. The top part shows a flowchart with actions: Exception, Log Exceptions to System History, Input File Added, Refine Input File, and Transfer Files. The Transfer Files action is highlighted with a green box. Below is a "Rule Parameters Editor - Transfer Files" dialog box. It shows a flowchart with "Refine Input File" and "Transfer Files" actions. The "File Paths" parameter is set to "Pages Produced" from "Files". There are sections for "Additional Properties" (String1-6) and "Advanced Options".

Parameter	Value
<b>Destination Computer</b>	SPOKE
Destination Context	File Receiver Job
<b>Rule Set Path</b>	File Receiver Rule Set
<b>File Paths</b>	Pages Produced → Files
<b>Additional Properties</b>	
String1	
String2	
String3	
String4	
String5	
String6	
<b>Advanced Options</b>	

Additional Properties  
Additional Properties that can be passed over to the destination computer.

Buttons: ? Advanced: Convert to Code Cancel OK

2. Configure the **Transfer Files** action to call a rule set in an environment and server suitable to your particular setup. The **File Paths** parameter carries the **Pages Produced** value from the **Refine Input File** action.

**Note:** Note that you can also configure the **String1** through **6** parameters to pass to the receiving rule set for additional information that may be required by the receiving rule set.



## How to use the Exception event

The term *exception* is shorthand for the phrase "exceptional event".

An exception is an event that occurs during the execution of a program, which disrupts the normal flow of the program's instructions. This is an unexpected error and different from an error condition that has been anticipated and for which there's a defined response.

These are some of the causes of program exceptions:

- Division by zero
- SQRT of a negative number
- Invalid array index
- Error on a call
- Error return from called program
- Start position or length out of range for a string operation

### Scenario

It is always possible that you could experience an unexpected error as a rule set runs. If this happens due to an exception, it is often difficult to see any evidence of the crash. Usually, the only notification will appear in the system diagnostics, which is rarely visible. This differs from a more normal error that would appear in a job's history.

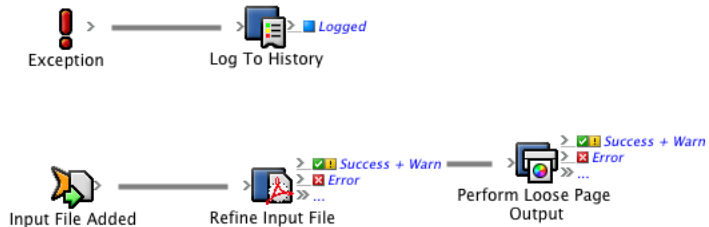
What we need is a method for the rule set itself to tell us that there was a problem and we can do that in a couple of ways.

- Build a generic exception handler into the rule set. I would recommend this as a "best practice" for every rule set you write.
- Attach an exception handler to a particular action. In a long rule chain, this becomes a lot of work and recommended only for actions that you suspect are behaving badly.

It is possible and sometimes desirable to have both a generic handler that performs a certain workflow when an exception occurs as well as a specific handler (for the same exception) that does something different. The example that follows will demonstrate this.

## Task 1: Create a rule set with a generic Exception handler

1. Create a new rule set as shown in the screenshot and use standard process templates for refining and proofing.



2. Configure the **Log To History** action to record an exception message.
3. Set the **Severity** parameter to **Error** and provide the action name (**Rule Name**) and exception message (**Message**) in the **Message** parameter.

Rule Parameters Editor - Log To History

Parameter	Value
Job	
Message	%triggerEvent.RuleName%: %triggerEvent.Message%
Severity	Error

Message (Type: String)  
The pre-translate

Parameter Value - Log To History : Message

String

Event Properties

%triggerEvent.RuleName%: %triggerEvent.Message%

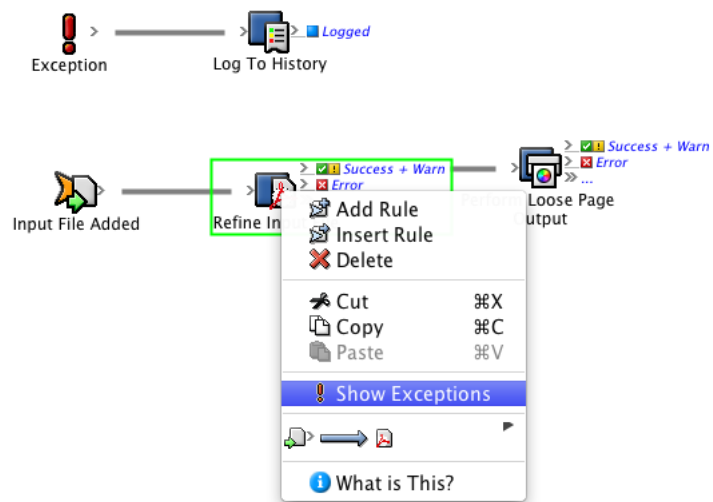
Exception

- Event time
- Is Root Event
- Message
- Rule Name

4. Leave the **Job** parameter empty.  
This tells the **Log To History** action to log this message to the system history. This is a convenient place to log generic exception handler messages.

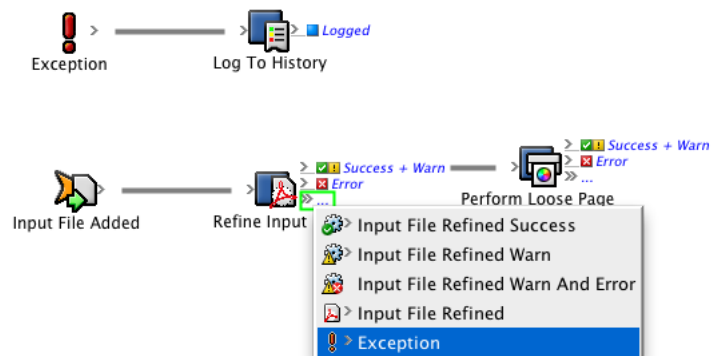
## Task 2: Create a rule set with a specific Exception handler

1. Right- click the **Refine Input File** action and choose **! Show Exceptions**.

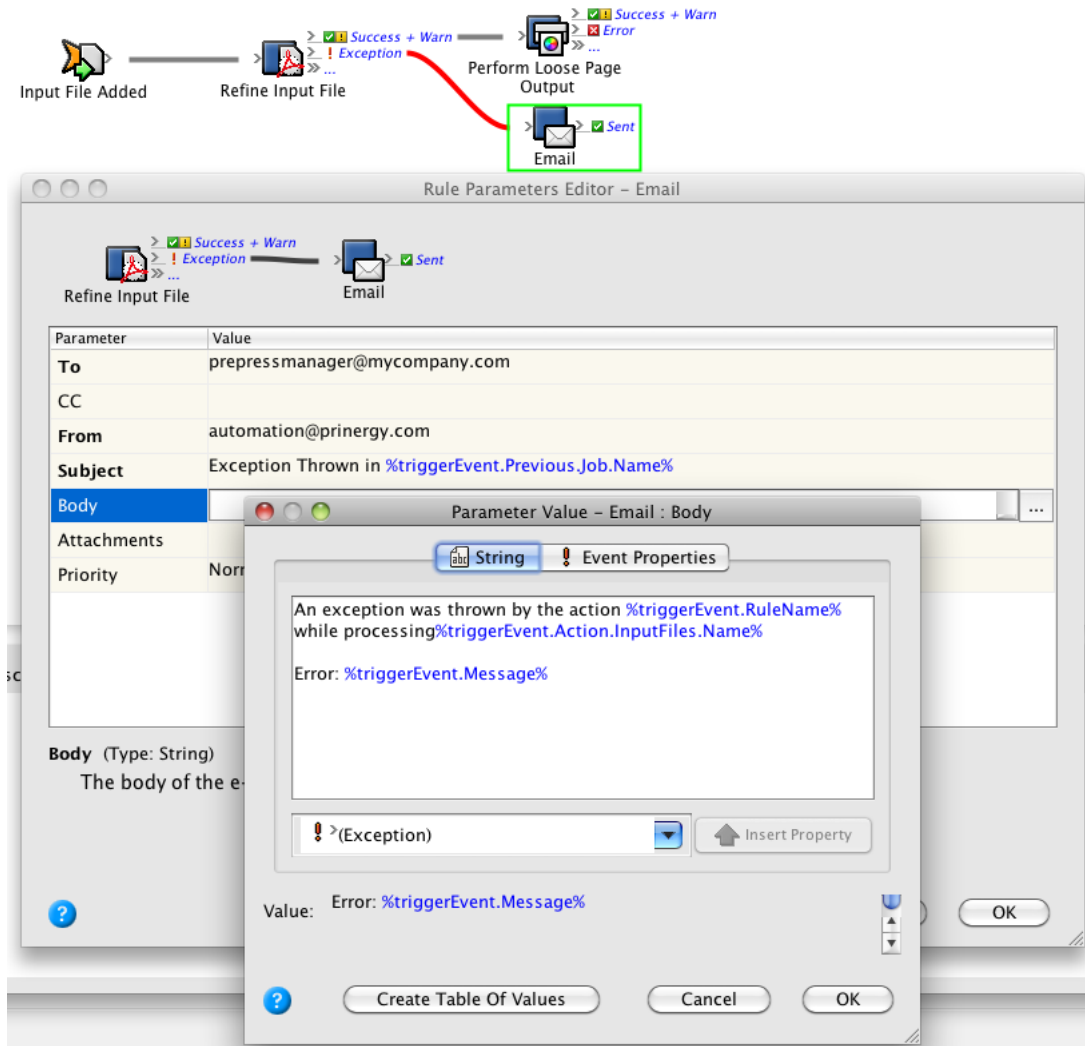


This makes the action's exception handler available.

2. Drag an **Email** action to the **Exception** resulting event.



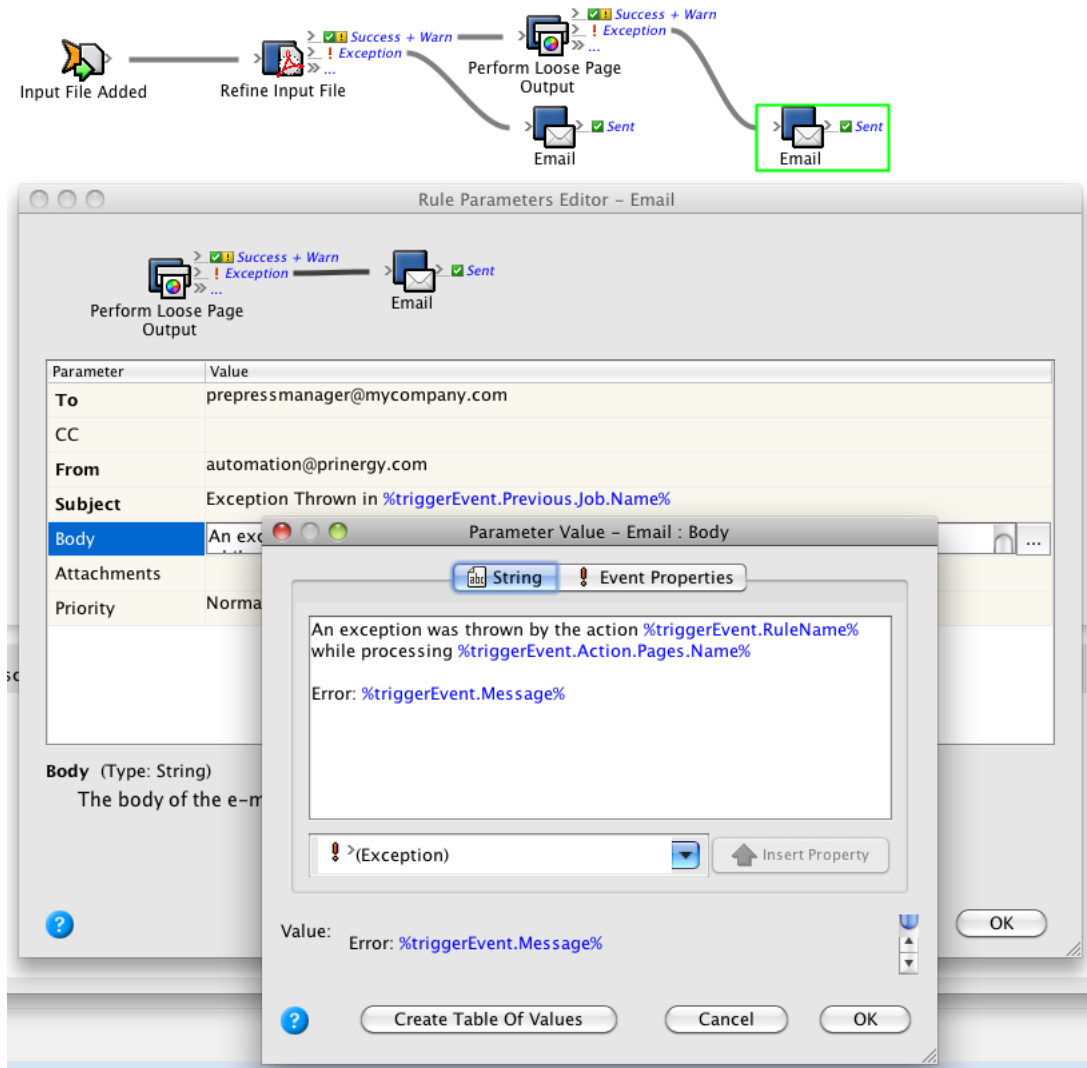
- Configure the **Email** action to capture relevant information about the exception, such as the action name and the exception message.



If an exception is thrown by this action, two different workflows will start—one to send this email and the other to log the information into system history.

- Drag an **Email** action to the **Exception** resulting event of the **Perform Loose Page Output** event.

- Configure the second **Email** action in a similar way to the previous one.



This rule set is now ready to handle an exception. If you wanted to test this, you would have to fabricate an exception. For example—find a numeric parameter and perform a "divide by zero" operation.

### Task 3: Include the rule set name in the exception e-mail message

If you test the above exception handler, you will find that the name of the rule set is not identified in the e-mail message. In the following example, you will find it easy to tell which rule failed because you only have one rule set and a couple of rules. In a production environment you may have many rule sets, each containing many rules, making it difficult to tell in which rule set the error occurred. To help identify the rule set, a rule set variable that contains the name of the rule set, can

be created. The value of variable can then be included in the exception e-mail message.

1. In Rule Builder, from the **Edit** menu, select **Manage Variables**.
2. In the Manage Variables Editor window that appears, click the **Rule Set Variables** tab.
3. Click the **Add Variable** button.
4. In the **Name** column, type the rule set name.
5. In the **Data Type** list, select **String**, and in the **Current Value** column select the default value of the rule sets current name.
6. To save the variable and close the editor, click **OK**.
7. For each exception that is being handled specifically with an e-mail, edit the body of the e-mail to include the value of the rule set name variable. The syntax to reference the rule set variable is `##Rule Set Name%`. So the new subject would be:

```
An exception was thrown by the action
%triggerEvent.RuleName% in rule set ##Rule Set Name
% while processing %triggerEvent.Action.Pages.Name%
Error: %triggerEvent.Message%
```

**Note:** If the rule set is copied or renamed, the default value of the rule set name variable must be manually updated to the new name of the rule set.





## How to enable/disable hot folders

RBA considers all the hot folders in a job as a single list.

The **Enable Hot Folder** and **Disable Hot Folder** actions allow you to enable or disable all the hot folders in a job.

Future versions of RBA may allow you to affect an individual hot folder, but in versions up to and including 5.1.2.1, you can only manipulate the entire list.

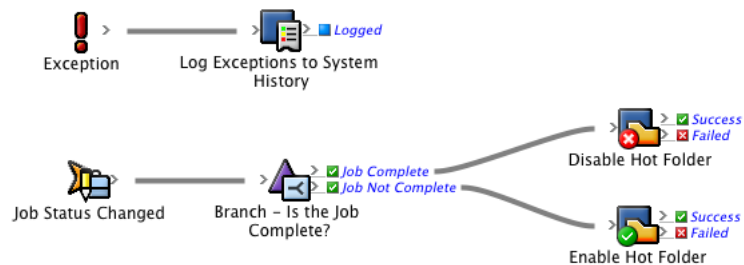
### Scenario

Hot folders, particularly in large numbers, can have a performance impact on a Prinerger server, so it makes sense to reduce their number where they are no longer being used.

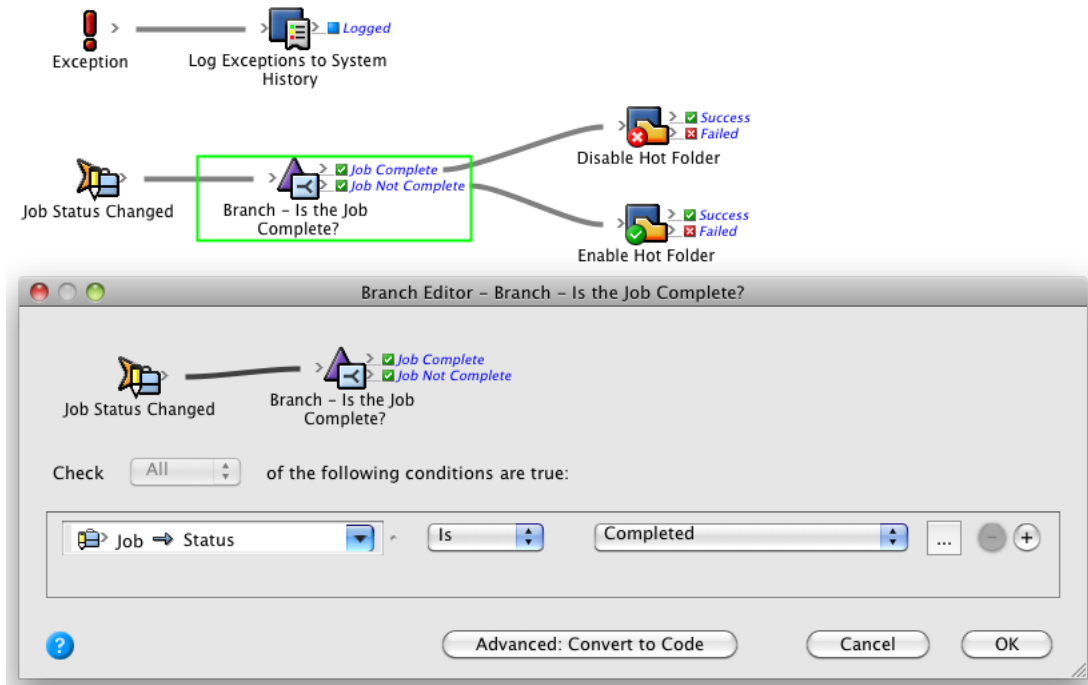
The example that follows will show how to turn off unused hot folders and then turn them back on as needed. This is a very simple example and you will probably want to enhance it to better suit your requirements

### Task 1: Enable/disable hot folders

1. Create a new rule set, as shown in the screenshot, and include an exception handler.

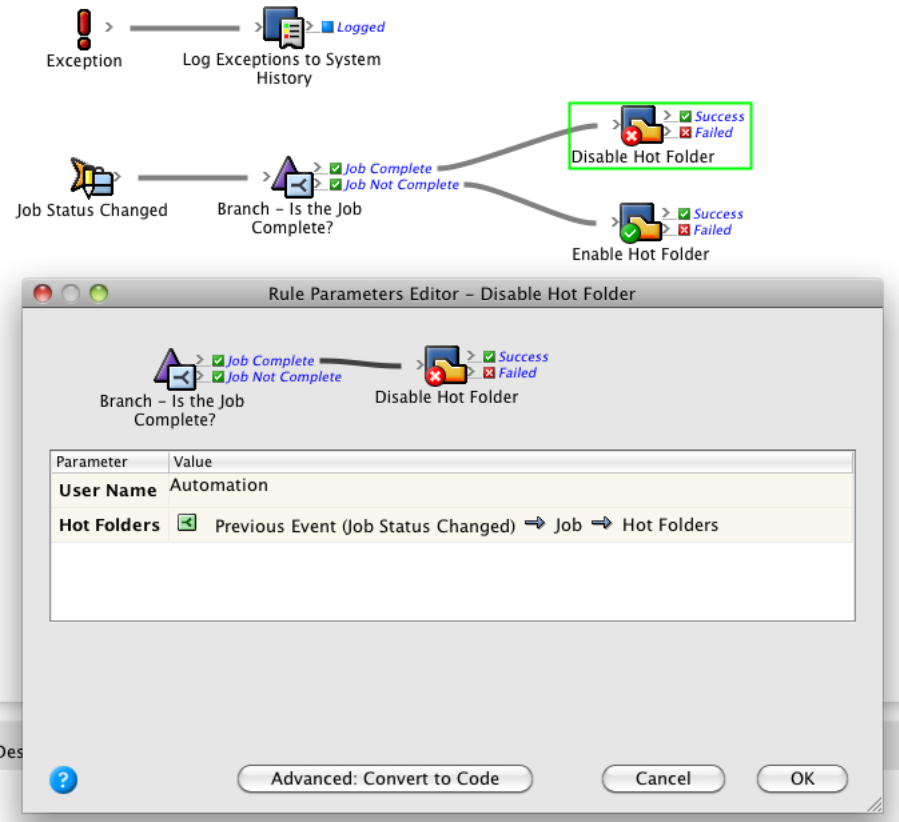


## 2. Configure the branch to check the current status of the job.



Note that this will be done only after the job status changes, so the current value has changed from what it was before the **Job Status Changed** event fired.

- 3. Configure both the **Disable Hot Folder** and **Enable Hot Folder** actions as shown.



- 4. Enable this rule set in the system environment.



## How to trigger events when a rule set is enabled/disabled

### Scenario

The rule set described in this activity will do two things:

1. Send an email when it is enabled
2. Send an email when it is disabled.

This is not a very compelling use-case, but you may want to use such events to manage the archiving and retrieval of Prinerly jobs, based on the jobs' rule set use, for example.

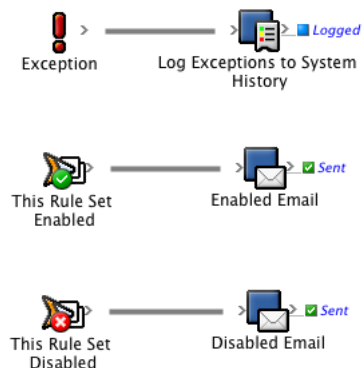
You could include a **This Rule Set Disabled** trigger event in a job that would move the job to an archive group in Prinerly when the rule set was disabled. This may be a signal that the job is complete.

Similarly, you could include a **This Rule Set Enabled** trigger that moves the Job back into a Prinerly production group, signaling that there is now more work to do with this job.

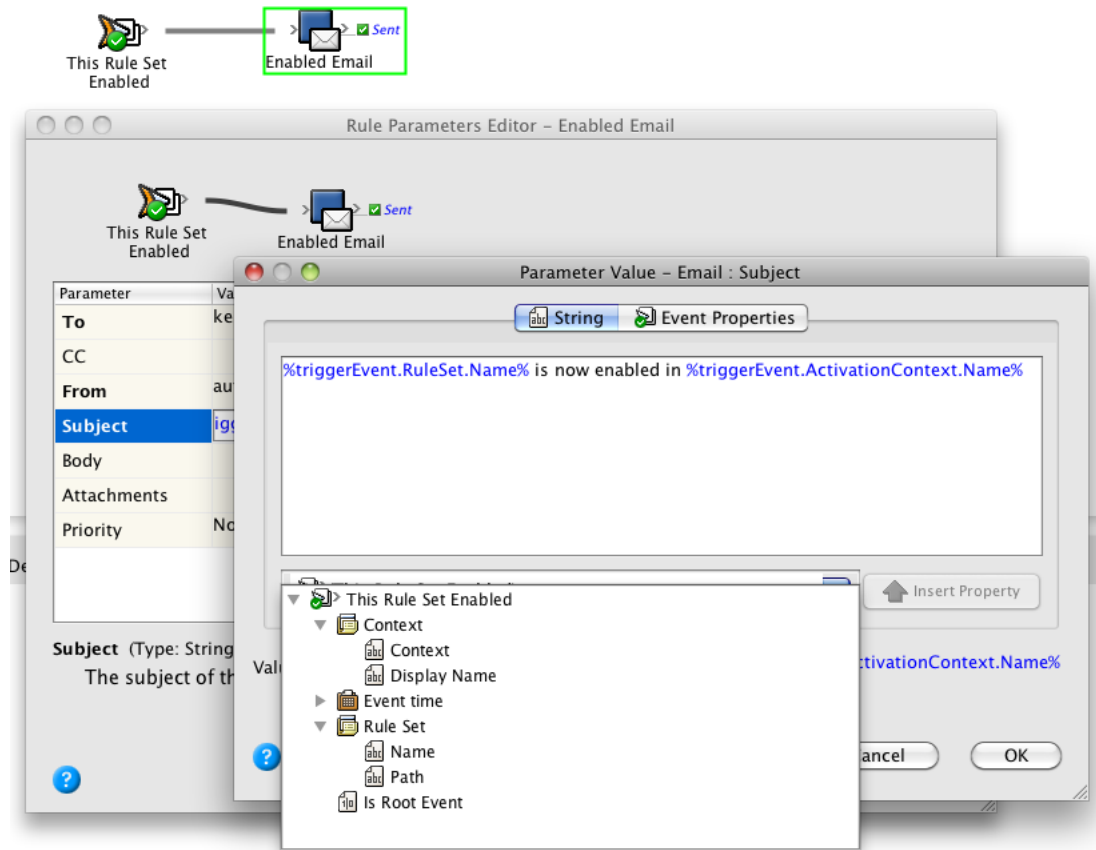
This is only one possible use of these trigger events. I'm sure there are many more creative uses out there.

### Task 1: Trigger events when a rule set is enabled/disabled

1. Create a new rule set, as shown in the screenshot, and include an exception handler.



- Configure both email actions as shown in the screenshot, to include the name of the rule set and the affected context in the email **Subject** parameter.



- Enable the rule set in your job.
- Check your inbox for messages as you disable and re-enabled the rule set.  
Be aware that there is a matching set of actions for these events, which means you can enable and disable one rule set from another.



## How to use other events and actions

This section provides a quick look at some additional events and actions, which you may find useful.

### Task 1: Configure transfer files

The **Publish File** action is associated with the **Publish File** process template.

This process allows you to output to:

- PDF Raster
- PDF (Vector output)
- PDF/X-1a:2001 (Vector output)
- PDF/X1-3 (Vector output)
- PS2 (Vector output)
- PS3 (PostScriptOut)

Output options:

The default output location for **Publish PDF** is `Job\System\DigitalPrintOutput\<GUID Folder>`.

In addition to generating PDF output, the **Publish File** action is used to configure files for submission to digital printers using managed connections.

### Task 2: Use the Publish File Done event

This event fires when a publish file process has completed, and a PDF or Adobe PostScript file was produced via the **Publish File** event (from Prinergy pages or impositions).

**Note:** The **Publish File** event differs from the **Publish to PDF** event.

Scenarios include when you need to attach a PDF proof to an email or when you submit a loose/imposed PDF to a digital printer.

### Task 3: Use the Page Edit Done event

This event is triggered after a page (or pages) have been edited with the Prinergy PDF File Editor. With the PDF File Editor, you can convert refined PDF files so that they can be edited in Adobe Illustrator, and you can edit photographs or images in the PDF file in Adobe Photoshop.

When you finish editing the PDF file, select **File > Scripts > Kodak > Submit PDF to Prinergy** to submit the PDF file to Workshop.

A possible scenario is when you automatically generate a loose page proof and request page approval after the page edit is done.





## Activity 1 - Create a rule set with a generic exception handler : Activity review questions

### 1. What is the benefit of adding a generic exception handler to a rule set?

This is the easiest way to listen for any exception coming from any rule in your rule set.

### 2. What are the benefits of having a "specific" exception handler in a rule set as well as a "generic" handler?

This is a way to have multiple responses to the same exception. You could do something different for both generic and specific handlers. For example, you could get RBA to log to history for the generic handler and to send an email for a specific handler.

### 3. What will be the results of concatenating the rule name and the exception error into one stored history message?

You will see the rule that threw the exception and the exception message stored in history.

## Activity 2 - Register the schema : Activity review questions

### 1. What is the purpose of registering a schema in RBA Schema Manager?

This is required for RBA to understand XML that is to be used in a rule set.

### 2. Do you have to register a schema multiple times?

No, once registered, a schema is stored and available for use by any rule set.

### 3. Would you need to update (re-register) a schema if it changes?

Yes, if the schema changes then it needs to be updated in RBA Schema Manager and any rule set that uses it needs to be re-compiled, (i.e., saved).

#### 4. Can you validate XML for a specific schema before you use it in a rule set?

Yes.

### Activity 3 - Associate a rule set with a job hot folder : Activity review questions

#### 1. What do you expect will happen when you drop an XML file in your job's hot folder?

The **Read XML** action will make the intent from the dropped XML file available to the rule set via its success resulting event. Action parameters from this point on can be populated with intent from this event.

#### 2. What will happen if you drop an input file (instead of an XML file) in your job's hot folder?

The **Read XML** action will fail and the rule set processing will stop. This is not the expected file type for this action.

#### 3. What will happen if you drop multiple XML files in your job's hot folder?

To process multiple XML files you need to use the **Split** action and handle each XML file separately. This is a valid use of XML hot folders and you might do this if you expect to get multiple file drops from external sources.

### Activity 4 - Hot swap the rule set and create a job group : Activity review questions

#### 1. Why would you hot-swap an enabled rule set instead of disabling, editing, and re-enabling?

If you hot-swap a rule set, you can be guaranteed that the trigger event of the rule set will not 'miss' an event. There will be no time when you do not have either version 1 or 2 of the rule set listening for trigger events.

#### 2. Under what circumstances should you preserve the enabled rule set name on hot-swap?

If you have two rule sets participating in a remote trigger relationship, you would want to preserve the name of the 'receiver' rule set. The 'sender' will be referring to it by name, so if you change that name, you will break the relationship.

---

## Activity 10 - Get the rule set to create final output : Activity review questions

### 1. Why are the four parameters **Imposition Plans, Surfaces, Separations, and Signatures** mutually exclusive in the **Perform Final Output** action?

Final output can only be performed on one imposition object (plan, surface, separation, or signature) at a time. You couldn't simultaneously output a signature and a surface at the same time, for example.

### 2. Under what circumstances is it necessary to provide a parameter value for only one of these imposition parameters?

When the imposition object is known at rule set runtime, then that object must be specified at rule set creation time. For example, if the resulting event leading into the **Final Output** action is from a **Signature Full** event, then the **Final Output** action must deal with signatures only.

### 3. Under what circumstances is it valid to provide a parameter value for all of these imposition parameters?

If the trigger event is a **Manual Trigger**, then we don't know at runtime which of the four possible imposition objects we will have to process, so it's valid to accommodate all four at rule set creation time. At runtime, only the manually-selected imposition object will be processed by the **Final Output** action.

### 4. Name two ways to identify a set of mutually exclusive action parameters.

By their blue shading and by the parameter description at the bottom of the action configuration window.

### 5. Would this rule set need to change for different XML files?

No, this is the power of using intent to drive a workflow. The intent can (and will) change, but a well-written rule set needn't.

## Activity 11 - Use temporary variables to simplify event references : Activity review questions

### 1. What would happen if rule set variables were used instead of temporary variables to store the elements from the **XML Data** object?

If more than one instance of the rule set executed at the same time the values from one instance would overwrite the others causing corrupted output or strange behavior.

## 2. What is another use for a temporary variable?

Hold an intermediate value that was calculated or produced dynamically for later reference down the rule chain.

## Activity 12 - Create a table of values : Activity review questions

### 1. Where would you use a table of values?

In any parameter that may require one of many possible values at rule set runtime.

### 2. Can a table of values be used more than once in an action?

Yes, almost every parameter in any action can have a table of values. This is an advanced technique but can result in a very powerful action.

### 3. At what point would a table of values make more sense than a series of branches?

A branch is an "If, Then, Else" calculation, so once you get past more than one branch making a decision on the same data, you would be better off using a table of values which is a "Select case statement".

## Activity 13 - Set up remote triggers : Activity review questions

### 1. Where would you use a pair of rule sets that relied on remote triggers?

The most common use of this would be between two different Prinerdy systems that needed to cooperate in a prepress workflow. For example, a hub site sending a request (trigger) to a spoke site to start a workflow with the provided information on the remote trigger.

### 2. How many string arguments can you provide to a remote trigger?

Six. If even more are needed, one option would be to write them to a file as key value pairs and send that file instead.

### 3. Is it possible to send a remote trigger to a rule set from outside of Prinerdy?

Yes, by using `raiseevent.exe`. (For more information, see the online help.)

Another option is to use the ExtAutomationApi web service interface:

---

Documentation: <http://<primary name or IP>:61235/ExtAutomation/RaiseExternalEventImp/Index.htm>

Example: <http://<primary name or IP>:61235/ExtAutomation/RaiseExternalEventImp/SendRemoteTrigger.htm>

## Activity 14 - Get the rule set to populate custom fields : Activity review questions

### 1. Name the available custom field types.

String, Boolean, Integer, Date, Decimal.

### 2. Name the available custom field scopes (the levels at which a custom field can be defined).

Job, Page, Page Set, Page Position, Imposition Plan, Signature, Surface, Separation.

### 3. Why can each custom field at each different level of a job have a unique value?

Custom fields are created and stored as arrays. This means that a string custom field called "myString" and assigned at job level can have a different value for every job.

## Activity 15 - Locate and rename files in a folder : Activity review questions

### 1. What if the `UserDefinedFolders` folder contained sub-folders. Would the files in the sub-directories get renamed and added to the job as well?

Yes they would. By default the **List Files In Folder** action searches sub-directories as well. To restrict the search only to the specified directory change the **Recurse** parameter to **False**.

### 2. What if it was a directory that was to be renamed. What would have to change in the **Rename One File** action?

The **File to rename** parameter should list the full path to the directory. The **New name** parameter should specify the new directory name.

## Activity 16 - Update a daily production record with the names of all pages approved that day : Activity review questions

### 1. Why is the **Limit Concurrency** action used in this scenario?

By placing a **Limit Concurrency** action in a rule chain, the number of instances that run concurrently can be controlled. If the **Limit Concurrency** action is not placed then sometimes the **Write Text To File** action may fail if too many instances are trying to write to the file at once.

### 2. What if there were number of additional actions to be performed after the record is updated?

A **release Concurrency** action should be inserted after the **Write Text To File** command in order to limit the time that the record is locked and ensure maximum concurrency in the system.

### 3. What if you wanted to open and read the CSV file from another rule set?

The other rule set should use a **Limit Concurrency** action with the same limiter name to ensure that the record is not being updated at the same time it is being read.

## Activity 17 - Send an e-mail with a summary of a daily production record : Activity review questions

### 1. What would be another way of implementing an infinitely repeating rule set?

Use the Microsoft Windows AT command to schedule a script that invokes a remote triggered rule set.



# Kodak



Eastman Kodak Company  
343 State Street  
Rochester, NY 14650 U.S.A.

© Kodak, 2016. All Rights Reserved.  
TM: Kodak

To learn more about solutions from Kodak, visit  
<http://graphics.kodak.com>.

Subject to technical change without notice.