Paper 019-2012

De Rigueur - Adding Process to Your Business Analytics Environment

Diane Hatcher, SAS Institute Inc., Cary, NC
Falko Schulz, SAS Institute Inc., Brisbane, Australia
Steve Sparano, SAS Institute Inc., Cary, NC

ABSTRACT

In an environment where there are multiple stakeholders involved in delivering business analytics, process is required to establish "rigor," a check-and-balance system where reports and other analytic assets must be verified and approved before being shared across the enterprise. Defining an approval workflow is a common mechanism for implementing this process. With the release of SAS® 9.3 Integration Technologies, your SAS business analytics environment now includes the ability to define workflows. This paper describes how to set up a report approval process where reports are developed, submitted for approval, and then made available in production (if approved). This process is managed via a custom portlet that is made available through the SAS Portal, leveraging the new workflow capabilities to track progress and execute batch SAS code.

INTRODUCTION

Historically (and we are talking all the way back to the year 2000), generating reports has been a fairly tightly controlled process. A typical structure has a central group of IT folks who develop most of the organization's reports and push them out to the ultimate consumer. Since the actual report creation ecosystem in this scenario is fairly well contained, it is easy to have internal processes to manage reports through a DEV ▶ QA ▶ Production flow.

But, over the last decade, there has been a huge pendulum swing toward self-sufficiency with creating reports. Having a central team created a significant bottleneck in the process, so enabling end users to create their own reports has become the standard, rather than the exception. Technology (including SAS technology) has grown and extended to support this new report creation structure.

This capability has, of course, led to the proliferation of reports as well as the proliferation of information interpretations. Information interpretations reflect each report creator's interpretation of what the organizations' information represents and the story that it tells. Usually, there has been no checks-and-balance system in place to approve which reports were showing or with whom they were being shared.

In SAS 9.3, SAS introduced some underlying technologies to help add more structured process management into business analytics as a whole. These workflow services are available as part of SAS Integration Technologies, so they are available for you to leverage with virtually any SAS solution, including SAS® Enterprise BI Server.

EXAMPLE REPORT APPROVAL PROCESS

For this paper, the example scenario is around a report approval process for reports created in SAS Web Report Studio:

1) A report builder (RB) develops a report that needs to be deployed as a production report. When RB is ready for the report to be reviewed and approved for promotion, he puts the report in a "Development" folder.
2) RB logs on to the SAS Portal and accesses the Report Approval portlet. RB indicates that the report is ready for approval and submits it to the appropriate Report Approver (RA).
3) RA logs on to the SAS Portal and sees there is a report to approve. RA can review the report, then either approve or reject the report.
4) If the report is approved, the system automatically moves the report from the "Development" folder to the "Production" folder.

SAS COMPONENTS USED FOR THE SOLUTION

Here are the SAS components that we leveraged to build the solution. Figure 1, below, shows how the high-level architecture and how the components work together.

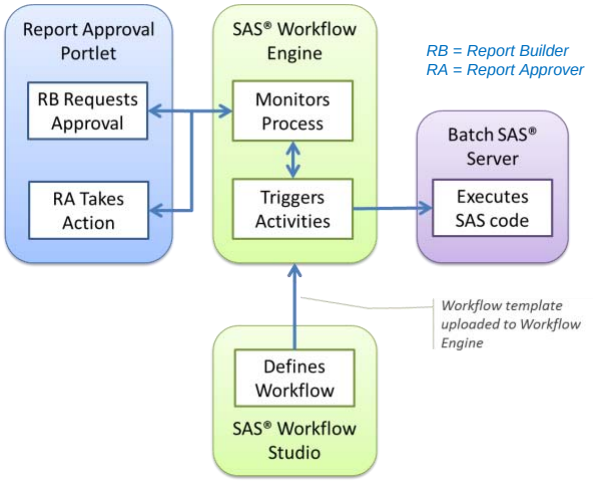


Figure 1. Architecture of Report Approval Process Solution

- **SAS Workflow**
  SAS workflow components are part of the SAS® Web Infrastructure Platform Database services that included in SAS® Integration Technologies. These components work together to model, automate, integrate, and streamline business processes, providing a platform for more efficient and productive business. SAS workflow consists of the following components:
  - SAS Workflow Services provide a set of standard interfaces that client applications can use to execute business processes and provide integration between people, systems, and business logic according to the steps defined in each business process.
  - SAS® Workflow Studio is a business process-modeling tool for the rapid development of process diagrams. It is used to design workflow templates that are administered by the workflow services.
- **Custom-Developed Portlet**
  A custom portlet was written, using SAS® AppDev Studio, to produce the workflow template. This portlet is available through the SAS Portal.
- **Foundation SAS® Code**
  Good old SAS procedural code was used to interact with the SAS® Metadata Server and the SAS® Content Server to programmatically move a report from one folder to another folder in the metadata.

PUTTING TOGETHER ALL THE PIECES

The solution leverages core underlying technologies that are provided as part of SAS Enterprise I Server.

- The SAS Metadata Server provides a central repository to store and manage your assets that are generated by SAS. SAS® Web Report Studio reports are stored in metadata in a folder structure that enables you to organize your content. In this example, we use two metadata folders to simply define a development and production environment.
- The SAS Content Server is the repository that contains the actual SAS Web Report Studio report definition file. This file also has an associated metadata object, and the two must be dealt with together when manipulating report locations.
- The SAS Portal is used as the channel for presenting the custom portlet that supports the report approval process.

In addition, a custom metadata user group has been created, `Web Report Studio: Report Approving`. The members of this group are allowed to approve reports to move to production. This provides a simple mechanism to control access to specific content and actions in the portlet.

SAS WORKFLOW

Figure 2 shows how the SAS workflow components fit into the SAS platform architecture:

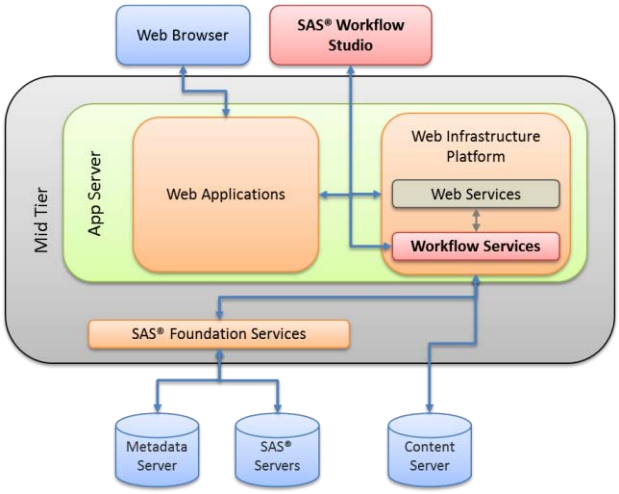


Figure 2. Workflow Services in SAS Platform

SAS Workflow Studio® is designed to quickly build, organize, and reorganize the workflows and business logic at the heart of a process integration application. SAS Workflow Studio handles complex business or application process logic and many common workflow and process modeling patterns. Process designers can use the point-and-click process authoring features of SAS Workflow Studio to define simplified automation of sophisticated business processes. Process diagrams from SAS Workflow Studio are represented in XML..

Basic business process logic can be defined and implemented in SAS Workflow Studio, including data-based process flow control using logical decisions (alternate paths) and parallel paths. Dialog boxes capture the author's definition of the process and create the XML templates that SAS Workflow Services uses to run the processes.

SAS Workflow Studio also provides predefined and customizable policies. Process designers can use policies to define actions (for example, send an e-mail) triggered by a particular workflow event (for example, process start). The policy represents an action at a specific step within the workflow. Because an activity can potentially initiate multiple actions, multiple policies can be associated to a single activity. Policies can perform arbitrarily complex programmed actions, such as sending notifications as portal alerts or e-mails, altering process flow, or even providing integration points across applications by invoking SAS code or Web services. Policies can also refer to data objects stored in a workflow to add, change, delete, or update peer processes during process execution (for example, updating a process with the completion date and time of a subprocess).

WORKFLOW LIFECYCLE

Each workflow defined in SAS Workflow Studio is stored as a workflow template in XML format. These templates can be opened from or saved to a local file system. In addition, SAS Workflow Studio supports persistence and versioning of the workflow templates using SAS Content Services. Finally, users can activate the workflow templates, which results in uploading the specified version as a formal workflow definition via SAS Workflow Services for instantiation by end-user client applications.

Together, the SAS Workflow Studio and SAS Workflow Services provide the ability to manage workflows that can be leveraged by solutions—including CRM, manufacturing, and health care—as platform components for process automation. Solutions integrate with SAS Workflow Services using technologies such as web services or published Java APIs to launch workflows, receive workflow status updates, generate notifications, generate alerts, and monitor workflow progress.

Figure 3 illustrates the lifecycle of a workflow from its creation in SAS Workflow Studio to its instantiation in an application.

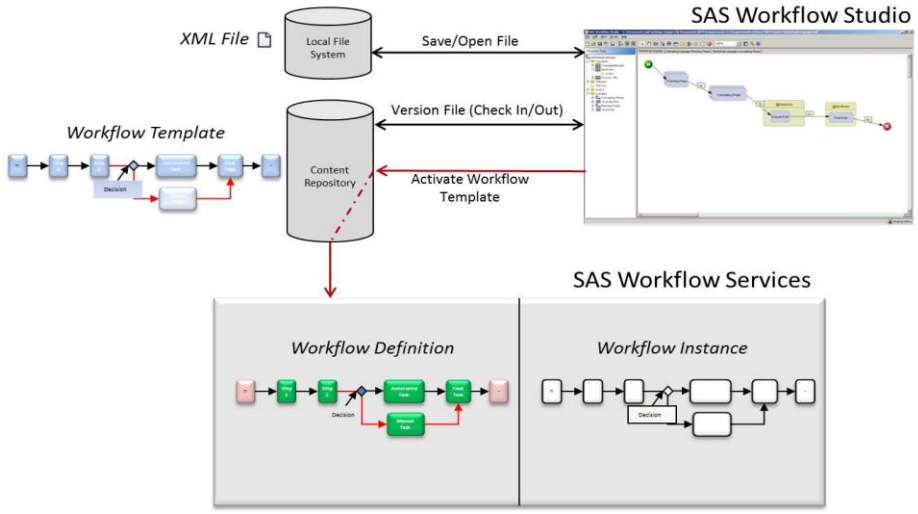


Figure 3. Lifecycle of a Workflow

Appendix A, "Creating a Workflow Template", reviews how to create a workflow template. Appendix B, "Designing the Approval Process Workflow" outlines how the workflow template for the report approval process can be created.

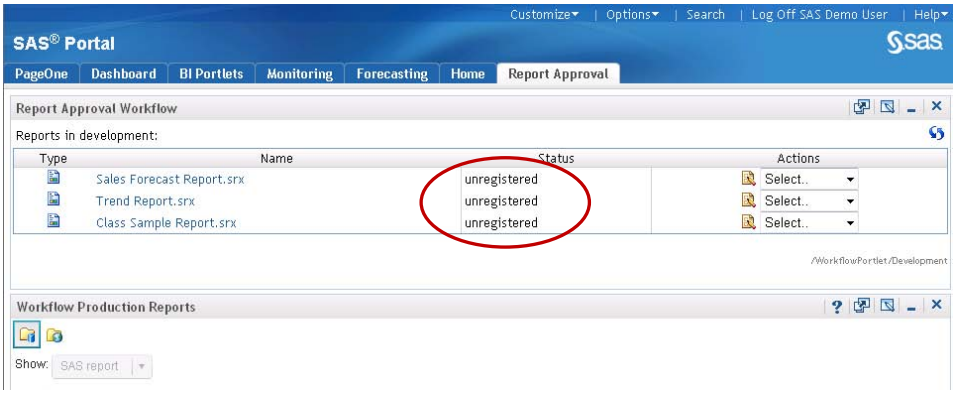
THE CUSTOM-DEVELOPED PORTLET

In order to demonstrate the report approval process, a portlet application was written and made available through the SAS® Portal. This application provides the user interface to interact with the underlying SAS Workflow Engine. The portlet is used for both the report builder (RB) as well as the report approver (RA). Depended on their role in the organization, they can monitor and track the report development until production.

In this example, all reports that are under development are located in a single SAS metadata folder, called "Development." The RB can continue to work on the reports in this folder until they are ready to be promoted to the production environment, which, in this case, happens to be another SAS metadata folder called "Production."

Portlet Interface

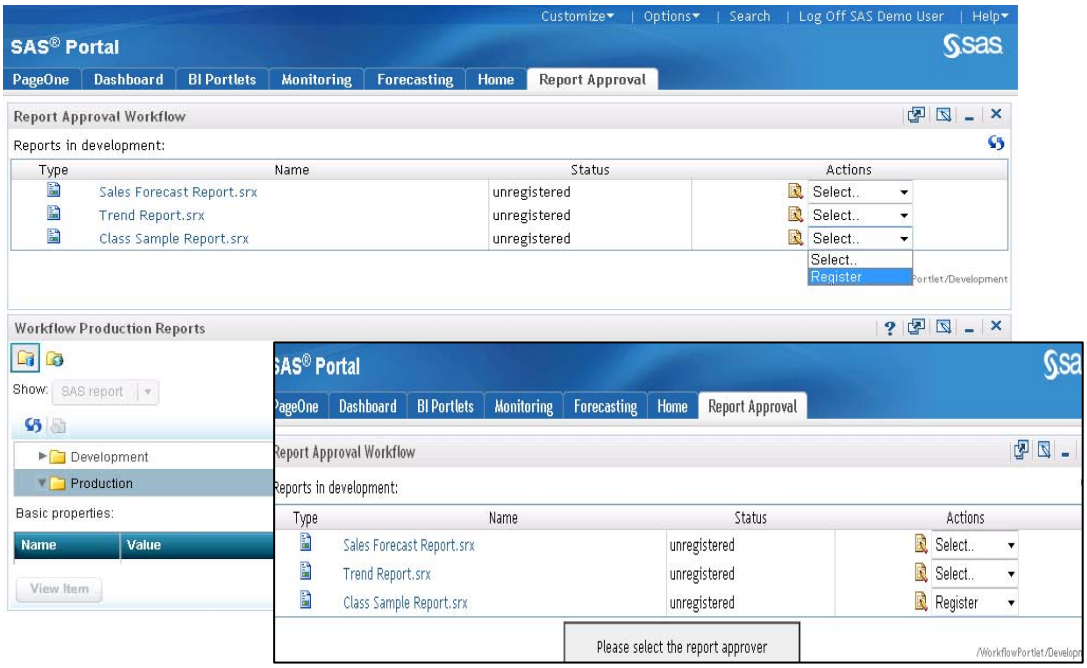
The RB logs in to the SAS Portal in order to see which reports he needs to work on. On the Report Approval page, there are two portlets shown. At the top is the Report Approval Workflow portlet, which shows a list of all available reports in the development area. Note that reports still under development are marked as "`unregistered`" (Display 1).



Display 1. Report Approval Portal Page

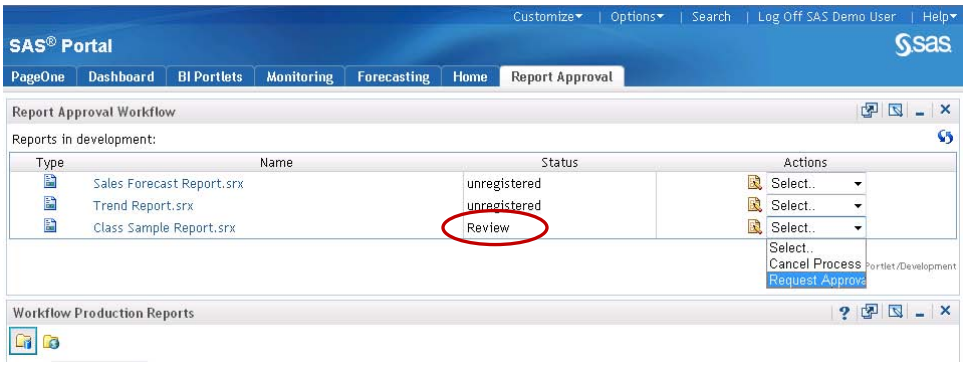
The bottom portlet is the SAS Navigator Portlet. This portlet is configured to show the contents of the report production folder. Because there are no production reports available yet this list is empty.

In order to start the report approval process, you must register a report. An Actions menu to the right of each report provides the "`Register`" option, as shown in Display 2. You are then prompted for which approver is wanted for the report. You typically select your manager or business unit head. Approvers need to be members of the metadata group "`Web Report Studio: Report Approving`" in order to be available in the selection box.



Display 2. Register a Report

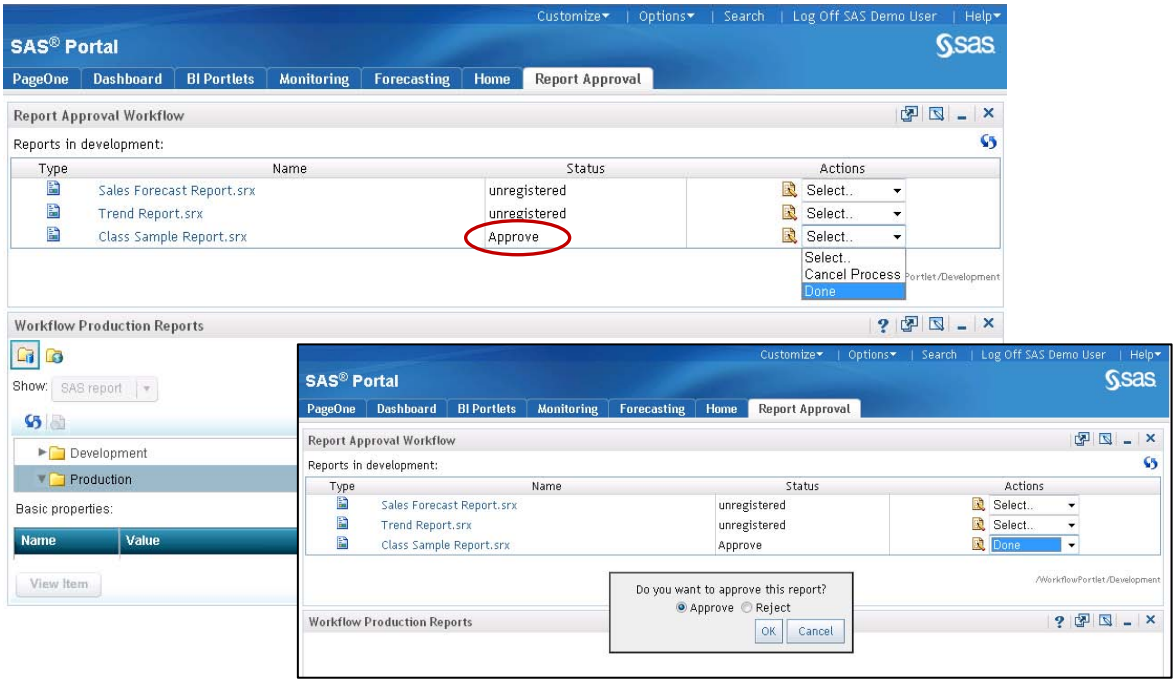
When the portlet has processed the request, the RB should see the status updated to "`Review`." At this point, the report is registered in the workflow engine with its own workflow instance attached to it. The RB can now start making changes to this report per business requirements. After development is finished, the request that the report be submitted for approval (Display 3) can be made.



Display 3. Request Approval

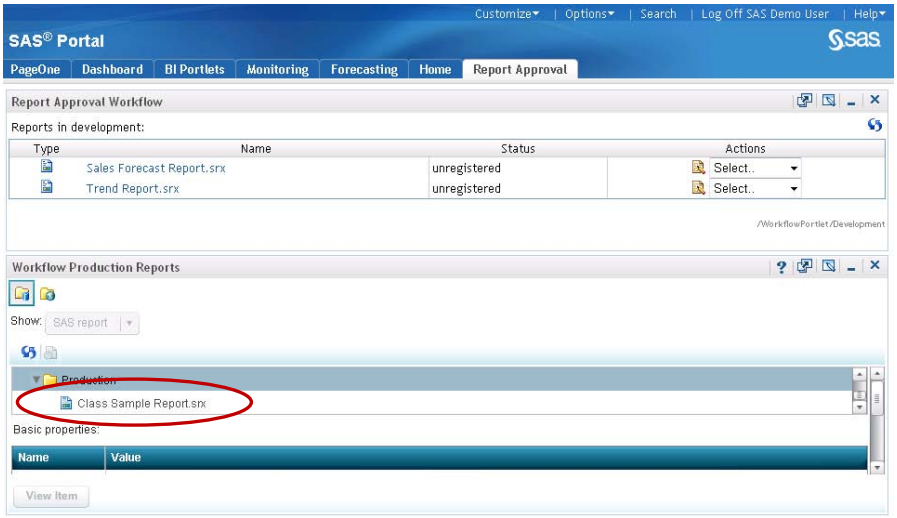
After the portlet has processed the request to submit for approval, the report automatically disappears from the list of available reports under development.

The RA uses the same Portlet to monitor reports in development and due to be approved. When RAis logged in to the Portal, the Approver sees that the status of one of the reports is updated to "`Approve`'=." RA can now take action on the report and select "`Done`" to approve or reject the report (Display 4).



Display 4. Approve Report

After the portlet has finished processing, the user can refresh the page to see the updated results. The processed report no longer shows in the Report Approval Portlet, but is now listed in the SAS® Navigator Portlet under the Production folder (Display 5).



Display 5. Report Approved

If the RA had rejects the report, the report appears back in the list of available reports for the RB. In this case, the RB can make further changes until the report is finally approved for production.

Portlet Structure

The Portlet is using the SAS Workflow Public API in order to communicate to the SAS Workflow Engine. The Portlet consists of two files:

- `sas.report.approval.par` = the SAS® Portlet responsible for rendering the user interface. Users are able to add this Portlet to any Portal page depended on permissions.

- `sas.report.approval.jar` = Java archive containing additional utility classes referenced by the Approval Portlet.

The Portlet has been developed using SAS AppDev Studio (Eclipse). The following diagram (Figure 4) visualizes the main tasks performed by the Portlet:

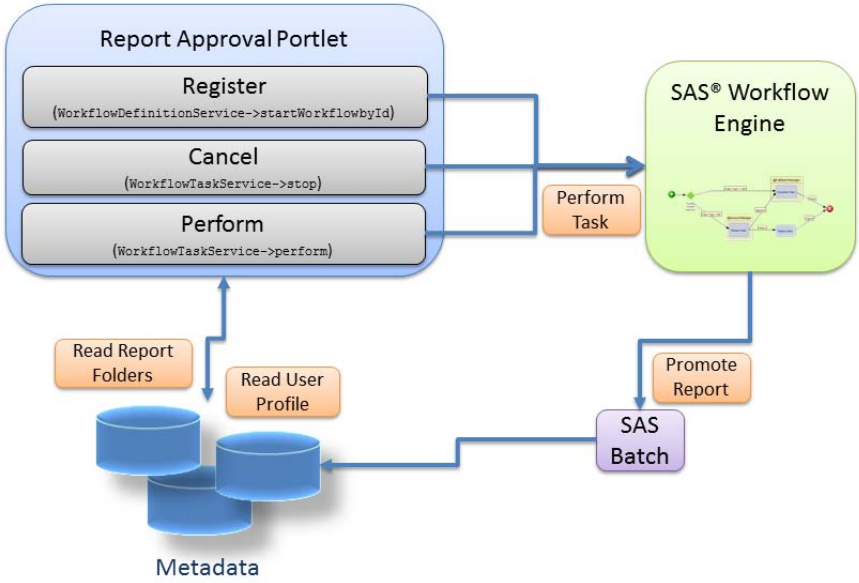


Figure 4. Portlet Actions

As shown in Figure 4. Portlet Actions, the portlet performs only three tasks:

1. Register – This action is executed using the Workflow Definition Service. It registers a new workflow instance. After it is registered, a unique instance ID identifies this workflow. The ID can be used later to perform specific tasks.
2. Cancel – If the user decides to stop the approval process, the Portlet removes the workflow instance for this report from the workflow engine
3. Perform – This action performs the current tasks assigned to the specific user. A typical task for a report approver would be, for example,an "Approve/Deny" report.

Note, that a SAS batch job is used to move a report from a development folder to a production folder. This batch job is executed by a registered policy inside the approval workflow.

THE SAS BATCH JOB

In the workflow template, it is possible to define a specific action based on a triggering event. These actions include calling a web service or executing a SAS program. In our example, when a report is approved to move to the production environment, the workflow template triggers an action to execute a SAS program. This program moves the report from one metadata folder to another metadata folder. As mentioned above, a report is actually made up of 2 specific objects–the metadata representation and the report definition file in the Content Server. Both parts need to be moved in this action.

Because there is no explicit "move" command in the SAS metadata architecture, the process of "moving" is actually a copy-then-delete process. We will explore how to do this with SAS code; you can find the complete sample program in Appendix C, "Sample SAS Program for Moving Reports between Folders."

Copy the Report

The first step is to copy the report by creating a SAS package file. This step must include both the metadata reference and the report definition file as part of the package. SAS Management Console provides an interface to do this manually, but here we do it programmatically. Below is the code snippet:

```
/*
/* Create export package file based on specified report
*/

x del &exportlog;

data _null_;
    cmd='C:\Progra~1\SASHome\SASPlatformObjectFramework\9.3\ExportPackage -profile "' ||
"&sasprofile" || '" -package "' ||
         "&package" || '" -objects "' || "&okreport" || '(Report)" -types Report -log "' ||
"&exportlog" ;
    put "Executing command::" cmd=;
    call system(cmd);
run;
```

The utility provided by SAS to create the package file is a Java-based command, so we have to call the command from outside of the SAS environment. On our Windows machine, we have to construct the command to point to the utility location and provide the appropriate parameters, as highlighted in yellow. The last two commands (before the `run;` statement) log the command that we are executing and make a Windows system call to execute it. As a result of this segment of code, a SAS package file is created.

- `&sasprofile` = the name of the metadata profile to use to connect to the SAS Metadata Server. The user in the profile must have ReadMetadata access to the reports being promoted.
- `&package` = the name of the SAS package file.
- `&okreport` = the name and location of the report to be promoted. The name of the report is passed from the workflow template based on which report was selected in the portlet.
- `&exportlog` = the name of the log file.

Write the Report to the Production Folder

The next step is to write the report to the new location, by importing the package file into the Production folder.

```
/*
/* Import package file to target folder
*/

%let impcommand = C:\Progra~1\SASHome\SASPlatformObjectFramework\9.3\ImportPackage -profile
"&sasprofile" -package "&package"
                              -target "&target" -types Report -log "&importlog" ;
x del &importlog;
x &impcommand;
run;
```

Once again, in our Windows environment, we have to construct the appropriate command and submit it to the Windows OS. In this case, we are using X-commands—just to show a different method of doing this.

- `&sasprofile` = the name of the metadata profile to use to connect to the SAS Metadata Server. The user in the profile must have WriteMetadata or WriteMemberMetadata access to the target folder.
- `&package` = the name and location of the package file that was created during the export process.
- `&target` = the target folder in the metadata server. This is provided by the workflow template.
- `&importlog` = the name of the log file.

Delete the Original Report

At this point, we have two copies of the report–one in the Development folder and one in the Production folder. The next step is to delete the report from the Development folder. This is a two-step process–we must delete the metadata reference and then delete the report definition file from the Content Server. First, t look at deleting the metadata reference:

```
proc metadata
 in=
```

```
"<DeleteMetadata>
   <Metadata>
      <Transformation Id='&deleteID'/>
  </Metadata>
   <NS>SAS</NS>
   <Flags>268435456</Flags>
   <Options/>
</DeleteMetadata>"
;
run;
```

This method leverages the SAS procedural interface for accessing metadata to submit the appropriate XML-structured command. In this case, a deletion is fairly straightforward, because you essentially need to provide only the object type (a report is an object type of "`Transformation`") and the object ID. The macro variable, `&deleteID`, is the metadata object ID of the report, and this is passed from the workflow template.

The last step is to delete the report definition file in the Content Server, using an HTTP interface to access the webDAV repository. After we know the report URL, which can be constructed from the name of the report provided by the workflow template, we can make the appropriate call.

```
proc http
     method="DELETE"
     url="http://localhost:8080/SASContentServer/repository/default/sasfolders&rptURL(Report)"
     webusername="&admuser"
     webpassword="&admpass";
run;
```

In this case, we are requesting a DELETE method on the report definition file.

- `&rptURL` = the URL address of the report, which is constructed based on the name of the report passed from the workflow template. We also need to append "`(Report)`" to the end of the URL as a configuration for reports that is specific to SAS.
- `&admuser` = User ID with Write access to the Content Server repository.

The URL for the report must follow standard HTTP-structure, which means that there cannot be any spaces in the URL. The sample program in "Sample SAS Program for Moving Reports between Folders" includes a macro that replaces spaces with "%20" to conform to the appropriate structure.

CONCLUSION

In SAS 9.3, SAS introduced some underlying technologies to help add more structured process management into business analytics as a whole. Leveraging these workflow services enables you to add process management back into the report creation process, and do it in a way that can be automated, rather than falling back to a central group to manage. Workflow services are designed to work within your current organizational dynamics while adding rigor around your decision-making process.

RECOMMENDED READING

- SAS® Workflow Studio 1.2: User's Guide
- PROC METADATA – 9.3: SAS® 9.3 Language Interfaces to Metadata
- PROC HTTP – 9.3: Base SAS® 9.3 Procedures Guide
- SAS® AppDev Studio™ 3.4 Eclipse Plug-ins: User's Guide

ACKNOWLEDGMENTS

Special thanks to Beth Ayres and Lori Small for their support in pulling together this sample workflow application!

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:

Name:	Diane Hatcher
Enterprise:	SAS Institute Cary HQ
Address:	SAS Campus Drive
City, State ZIP:	Cary, NC 27513
Work Phone:	(919) 531-0503
E-mail:	Diane.Hatcher@sas.com
Web:	http://www.sas.com

Name:	Falko Schulz
Enterprise:	SAS Institute Australia
Address:	1 Eagle St
City, State ZIP:	Brisbane, QLD, 4001
Work Phone:	07-3233 320
E-mail:	Falko.Schulz@sas.com
Web:	http://www.sas.com

Name:	Steve Sparano
Enterprise:	SAS Institute Cary HQ
Address:	SAS Campus Drive
City, State ZIP:	Cary, NC 27513
Work Phone:	(919) 531-0975
E-mail:	Steve.Sparano@sas.com
Web:	http://www.sas.com

APPENDIX A: CREATING A WORKFLOW TEMPLATE

Using SAS Workflow Studio's diagram editor, the process analyst can graphically design the relevant sequence of activities that make up the workflow. Each workflow or process is a collection of elements assembled using the following object model:

- activities
- data objects
- policies
- statuses
- participants

Activities

Activities can be atomic nodes (tasks) or can contain collections of related nodes (local subprocesses). The process hierarchy is represented in the process tree as expandable folders. Tasks are represented by an activity icon, and the root workflow and its subprocesses are denoted using the process icon.

Data Objects

General process data and logic should be separated from actionable business data and logic. Processes embody the abstract data and logic; data objects embody the relevant business data and policies the business actions.

Policies

Policies encapsulate event-triggered, executable business logic that can reference process data to add, change, delete, or update peer processes at run time. These events occur when there is a change in the process. Events can be triggered when there is either a change in the state or status of the process, generated by using a timer for single or repeated actions or received by external systems.

Statuses

Statuses link the process logic and business logic because they represent transitional states between activities and process flow elements. The status values are precondition logic used to initiate state changes in processes or trigger the execution of policies.

Participants

Participants drive process access and authorization by linking the workflow authorization roles to the SAS platform users, groups, and organizational roles.

Process Diagram Elements

Global data objects, policies, statuses, and participants are associated with the top-level folders under the workflow root. Activities can also contain locally defined data objects, policies, statuses, and participants. Local elements that exist in the context of a specific activity are accessible only at the activity-level, not by the other tasks or subprocesses. Data objects can be defined on the root or subprocess level, but the visibility can be configured by enabling the **`Visible in entire subtree`** option, which means the data definition is available to the children within that process level for logical evaluation or policy execution. However, these global elements are not re-created as local values associated with the child elements.

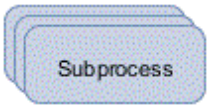
Each process begins with a Start node and contains one or more activities (tasks or subprocesses) before terminating with at least one Stop node. Each new diagram includes a Start node and a Stop node, but a single Stop node might terminate multiple activities. Likewise, the Start node can be used to initiate multiple activities.

The following elements can be used in a process diagram:

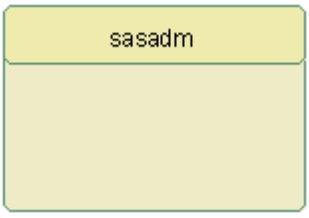
The Start node must precede the first activity in the process.

The Stop node must be connected to any activity that leads to process termination.

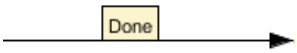
Activities are generally individual work items in the process that can represent automated or manual tasks.



An activity element with a stacked appearance represents a subprocess. A subprocess contains one or more activities that might, in turn, represent subprocesses resulting in a process hierarchy. You can create subprocesses and edit the contained activities from the drawing editor.



Swimlane elements are used in SAS Workflow Studio to group activities assigned to the same participant definition. They can be explicitly assigned to a Participant object or they can be implicitly assigned via a swimlane policy. The swimlane policy derives the user, group, or role value defined by the specified data object at run time.



The Sequence Flow tool is used to connect process elements. This connection element can also be used to designate process status (that is, state changes or transitions between activities).

Annotations are used to hold additional information. These notes are for presentation only and are not associated with the run-time process definition.

You can use the drawing tools in the Studio toolbar to place activities on the diagram editor and connect them using the Sequence Flow element. You can also select and right-click any activity or connection on the diagram editor to add objects. Alternatively, you can use the process tree pop-up menus to add activities and other process elements.

APPENDIX B: DESIGNING THE APPROVAL PROCESS WORKFLOW

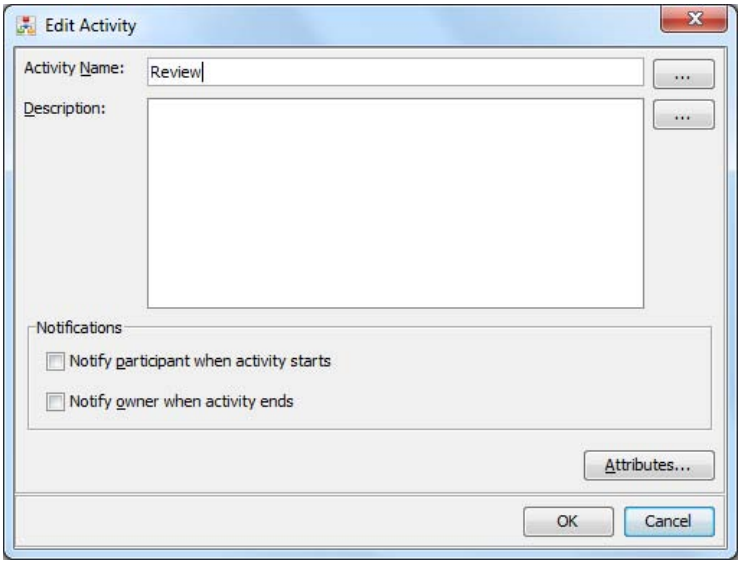
The Report approval process has the following activities:

- Review
- Approve/Reject
- Move Report to Production

The first two of these activities are performed by a Role, and the last is performed by an automated mechanism. In our workflow, the automated mechanism will be a batch SAS job. The SAS job will promote the report from the development folder to the production folder.

To add the first activity (Review) to the process, select the ACT icon from the toolbar and drag the activity to the drawing area. The drawing area, by default ,will have a start node and an end node added.

Double-click the new activity to open the edit window. Rename the activity "Review" by entering the name in the **Activity Name** field. Press **OK** to close the Edit Activity window.



The process will need two other activities:

- Approve: activity for approver to approve the report
- SASCodeInvoked: to perform the automated step of promoting the report from the Development folder to the Production folder.

Perform the same steps done for the Review step for each of the other two steps. After these are complete, the process design looks like the figure below.

In addition to the human-based and automated activities, there are decision points within the process that must be included. These decision points are used to assess input from users and manage control flow using the user input to make decisions.

The process will need to support routing the control flow to the SASCodeInvoked activity if the user approves the report. If the user rejects the report, then control flow must pass back to the review activity to alert the user that more review is needed before the report can be approved.

To add the decision node, select the Decision node icon and drag to a point between the Approve and SASCodeInvoked activity. Later in this paper (“Designing Transitions between Activities”), we will show how to configure this decision node to control the processing flow. After the **Decision** node has been added, the process design should look like the figure below.

Designing the Data Objects for the Process

Data must be passed into the process when initiated. This data is then used by the workflow to control flowm or it is passed to external calls to control external processing. SAS Workflow uses Data Objects defined within the workflow to hold these values as well as to evaluate and pass to SAS programs as input parameters, for example.

For our design, we need to capture some data for use both within the process and to pass to the SAS code that will be invoked to perform the automated task of promoting the report from the development folder location to the production folder location. The data objects that we will need to define include the following:

- DevWRSFolder: folder location where report is placed that needs to be reviewed
- ReviewWRSFolder: folder location where report is placed when it is being reviewed
- ProdWRSFolder: folder location where report is placed after it has been reviewed and approved
- ReportApproved: indicator that report has been approved or not approved
- ReportID: Unique ID of the report (passed to the workflow)
- ReportURL: Location of the report (passed to the workflow)
- ReturnCode: Return code passed to workflow by SAS code invoked during processing
- WRSApprover: Person or Role responsible for approving the report
- WRSReviewer: Person or Role responsible for reviewing the report

To create Data Objects for each location, you need to perform the following steps:

1. In the process tree, right-click the top-level or local data object folder.
2. Select **Add Data Object** from the resulting pop-up menu.
3. Specify a label for the object in the **Data Object Label** field in the New Data Object dialog box

For each of the data objects in the table below, select the designated values for the data object type:

Data Object	Type
DevWRSFolder	Short Text
ReviewWRSFolder	Short Text
ProdWRSFolder	Short Text
ReportApproved	Short Text
ReportID	Short Text
ReportURL	Short Text
ReturnCode	Short Text
WRSApprover	Short Text
WRSReviewer	Short Text

Data objects that are accessed at the individual activities level must also be defined at the local (Activity) level as well. You can add global data objects that are available for use by the entire process or as local data at the activity level, which is accessible only for use by the parent activity. Because the activities modify certain data objects and because their values need to be accessible by the specific (or parent) activity, they must also be defined at the individual activity.

The activities that require local data objects and the data objects that we will define for the activities include the following:

Activity	Data Object	Type
Approve	ReportApproved	Short Text
	ReportID	Short Text
	ReportURL	Short Text
Review	ReportID	Short Text
	ReportURL	Short Text

The process is the same to add data objects at an Activity level:

1. In the process tree, right-click the Activity rather than the top-level process name.
2. Select **Add Data Object** from the resulting pop-up menu.
3. Specify a label for the object in the **Data Object Label** field in the New Data Object dialog box

Identifying Users Groups and Roles for Each Activity

Within the process design, we will designate two roles to perform two of the activities in the process.

- Report Builder
- Report Approver

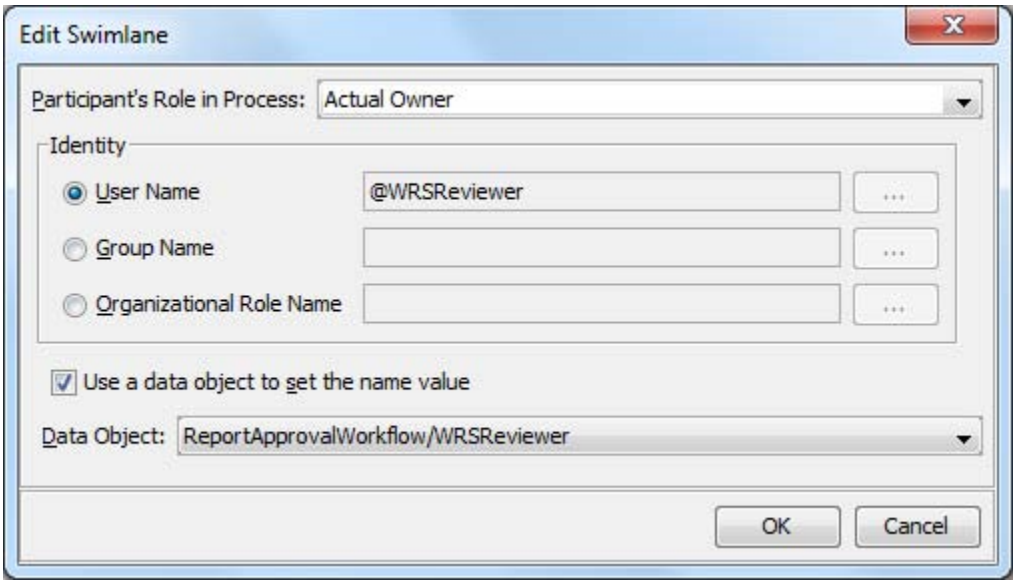
The roles defined are assigned to perform the respective activities using the swimlane construct.

Earlier, we defined two data objects to contain the values that represent the participants that will perform the activities for Review and Approve. Now we will use those data objects to configure the swimlanes to use those values passed at run time to assign the activities to those participants.

To do that, create a swimlane and assign the value of the data object as the value.

1. Select the option **Use a data object to set the name value** as shown below.
2. For Review, select **WRSReviewer** data object using the menu.
3. For Approve, select **WRSApprover** data object using the menu.

For the Reviewer Activity, after you've configured the swimlane, the Edit Swimlane window will look similar to this:

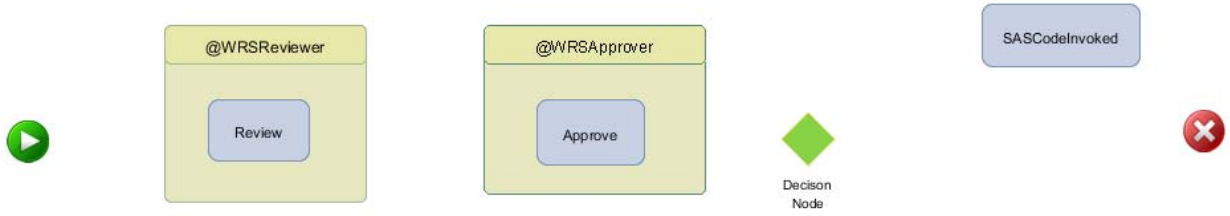


The data object acts as a placeholder (designated by the @ sign preceding the data object name) for the value until it is assigned at run time. After the swimlane is created, drag the activity onto the relevant Swimlane in the diagram.

For each of the two activities, configure the swimlane using this information:

Activity	Swimlane Data Object
Approve	WRSApprover
Review	WRSReviewer

After the two activities have the swimlane configuration complete, the workflow diagram will look like the following:



Note that workflow participants assigned in a swimlane using the Data Objects (as used above and showing the @ sign) are defined using the SAS Authentication Service. The following participant identity types are associated with the SAS Metadata Server definitions:

- If a user type is specified in the data object used, then the name must be a SAS platform user.
- If a group type is specified in the data object used, then the name must be a SAS platform group.
- If an organizational role type is specified in the data object used, then the name must be a SAS platform role.

Designing Transitions between Activities

The process that we are designing will manage a report as it moves between different states. The SAS Workflow Studio supports the different states by using transitions to model the paths that the report will follow in the process.

For our design, the report can assume the following states:

- Unregistered: before the workflow is started and before the report is submitted for review and approval
- Registered: the report has been submitted and is now being reviewed
- Approved: the report has been approved
- Rejected: the report has been rejected and needs to be reviewed further

The transitions between the activities are how we will model the states. They will be used by the portlet to move the process from one state to another. To model these transitions, we must do the following:

- add the statuses on each of the local activities that represent the status after the activity completes
- connect the activities using connectors
- assign status to the connectors to represent the paths that the workflow will follow when the status is set by the portlet

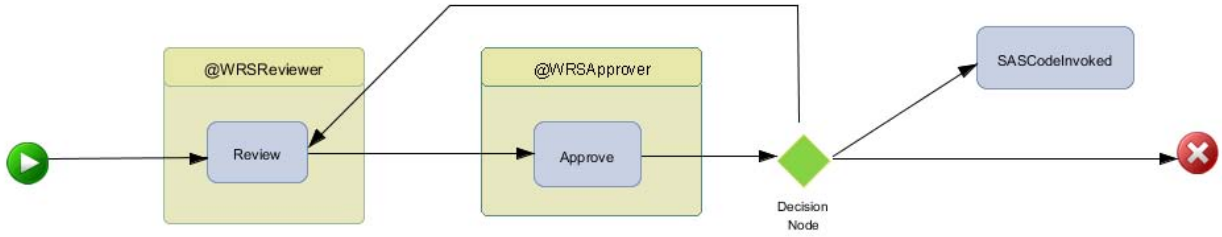
To define a new status, follow these steps:

1. In the process tree, right-click the Statuses folder and then select **New Status** from the resulting pop-up menu.
2. In the New Status dialog box, enter a name for the status in the **Status Name** field. Description is optional.
3. Select the option **Visible in entire subtree.**. This makes this status definition accessible to other activities in this process. When you are done, click **OK** to save the new status definition.

Using the connector icon, link the activities in the process following these steps:

1. Connect the **Start** node to the Review activity.
2. Connect the Review activity to the Approve activity.
3. Connect the Approve Activity to the **Decision** node.
4. Connect the **Decision** node to three different activities.
5. **Decision** node to End.
6. **Decision** node to SASCodeInvoked.
7. **Decision** node to Review.

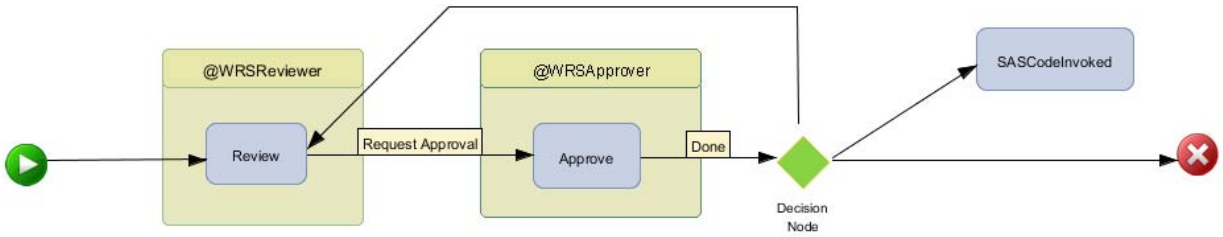
After you complete these steps,, the process design will look similar to the figure below:



For each of the connectors drawn, statuses are assigned using the statuses designed earlier for each activity. To assign a status to each connector, right-click on the connector line, and select the status using the table below:

Activity	Status
Review	RequestApproval
Approve	Done

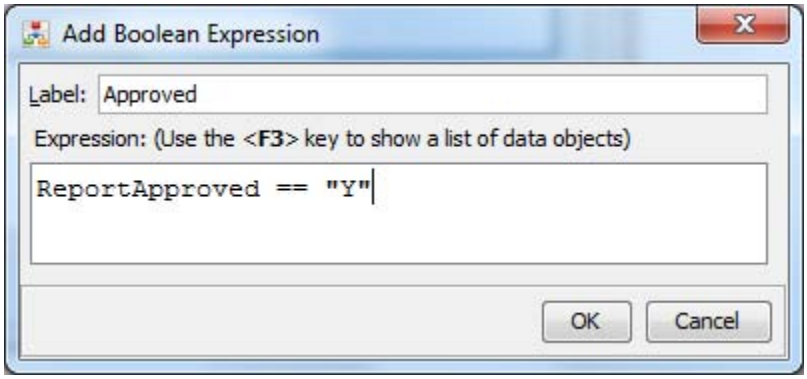
After the statuses have been assigned, they appear in the process and look similar to the figure below:



The **Decision** node is configured differently than other activities in the workflow because there might be many decisions made in the **Decision** node and each can result in a different path. From the **Decision** node, there will be two possible decisions, Approved or Rejected, resulting in three possible paths that can be executed.

The **Decision** node must be configured with decision logic that will be used to control the paths that originate from the **Decision** node. We have defined that there will be either an Approved decision or Rejected decision, and the logic for the decisions is defined using the following steps:

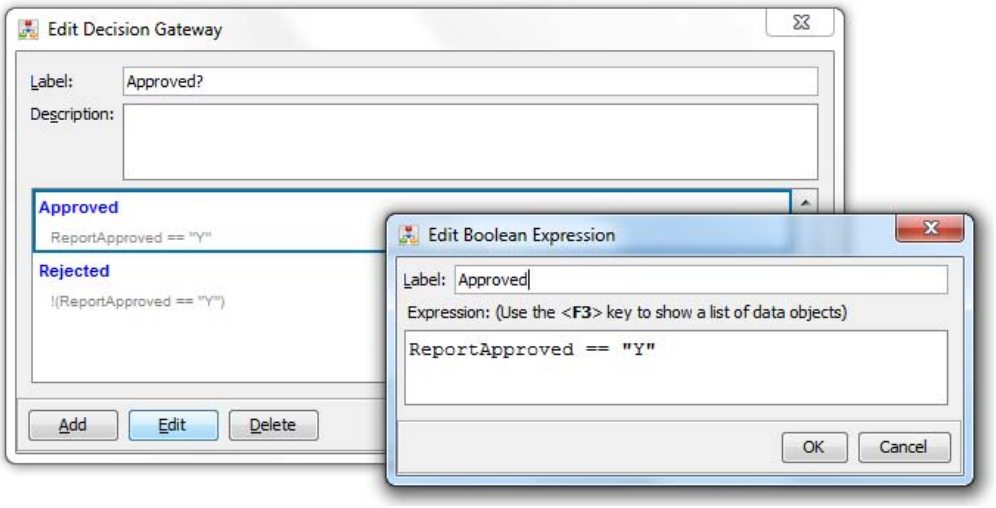
1. Double-click the **Decision** node to open the editor.
2. In the **Label** field, enter **`Approved?`** for the **Decision** Node.
3. Select **Add** to add a decision.
4. In the **Label** field, enter **`Approved`**.
5. In the **Expression** field, press F3 to see a list of the Data Objects defined for the workflow.
6. Find and double-click on **ReportApproved**, the data object we will use to make our decision.
7. Enter the decision logic as shown below: **`ReportApproved == "Y".`**
8. Click **OK**.



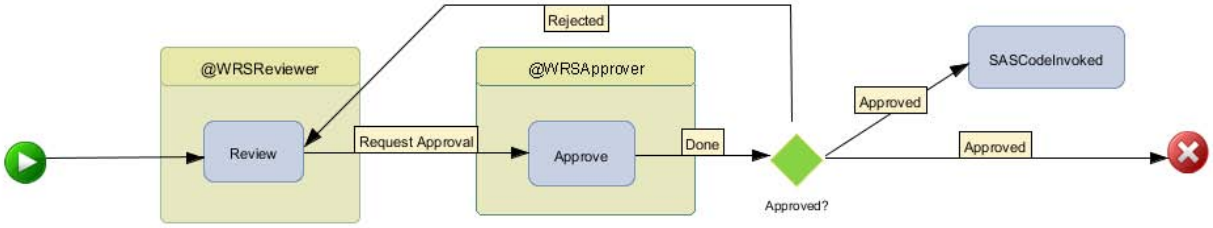
Before closing the Edit Decision window, repeat the process to add the second decision.

1. Select **Add** to add a second decision.
2. In the **Label** field, enter **`Rejected`**.
3. In the **Expression** field, press F3 to see list of data objects defined for the workflow.
4. Find and double-click on **ReportApproved**, the data object we will use to make our decision.
5. Enter the decision logic as shown below: **`ReportApproved != "Y"`**.
6. Click **OK**.
7. Click **OK** to close the Edit Decision window.

When you finish designing the **Decision** node, the Edit Decision window will look similar to the window below:



After the Decision is configured, assign the status of Approved or Rejected to the connectors that connect the **Decision** node to the other activities. This is accomplished by right-clicking on the connector, and choosing the appropriate status. After you've finished, the process design will look similar to this.



Designing External Actions

At the end of the Report Approval process workflow, actions are invoked based on the decisions (approvals or rejections) made during the workflow. As you have seen, the processing can take various paths based on input from the user.

In this workflow design, a decision node controls flow. Upon rejection, control returns to the Reviewer and the Review activity. This behavior is modeled in the figure above. However, in the case of approval, the *SASCodeInvoked* activity is executed, along with the process going to the end node for completion.

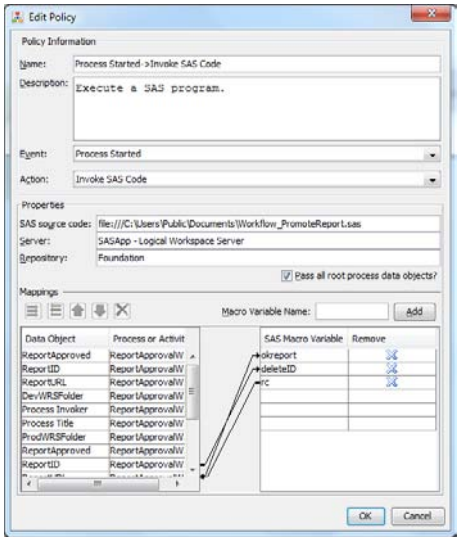
Within the *SASCodeInvoked* activity, a workflow policy is configured to invoke SAS code written to perform the promotion of the report from the development folder to the production folder. Invoking SAS code is one example of what SAS Workflow Policies can be configured to perform. More details about the other actions that can be performed are found in the *SAS® Workflow Studio: User's Guide*.

To configure a policy for an activity, the following general steps are performed. These steps are general due to the variability across various policies:

In the process tree, right-click the top-level or local Policies folder in the process tree and then select **New Policy** from the resulting pop-up menu.

In the Edit Policy dialog box, select the desired values.

- Event: corresponds to the workflow event that triggers the policy.
- Select **Process Started**.
- Action: corresponds to the policy type. The selection controls which fields appear in the Properties section.
- Select **Invoke SAS Code**. In the display below are specifics for the settings to configure the rest of the policy. These are specific to the SAS Code written for this process and can vary in your environment.
- Select **OK** to save the policy definition.



SAS code will be executed to perform a SAS action to promote the report from a development location to a production folder location. In order to accomplish this, SAS code has been written and will be executed using the SAS Workflow *InvokeSASCode* policy.

Publishing the Process for Execution

After the process design is complete, the process template needs to be uploaded to the Process Repository and activated. After it is activated, the process can be configured and launched from the SAS Portlet.

Before it is activated, the process that we have designed must be uploaded, or stored, in the process repository. After it is uploaded to the repository, the process is managed by the SAS Content Server using version control; changes are managed using locking and check-in and check-out procedures. This allows other authorized users to manage the process design changes over time while tracking those changes in a centralized repository.

The following steps must be taken to upload the process and activate it on the process repository:

1. From the Server menu, select **Save to Repository**.

   Note: These options are not active until you log on to the server.

2. Select the **Activate** option if you also want to activate the current version in the workflow. After a process template is activated, it is then available to be launched by the appropriate application.

APPENDIX C: SAMPLE SAS PROGRAM FOR MOVING REPORTS BETWEEN FOLDERS

```
/* redirect log file for debugging purposes */
filename logout "D:\MyDemo\WorkflowPortlet\Workflow_PromoteReport.log";
proc printto log=logout new;
run;

/* set sas options and print out all macro variables */
options symbolgen NOXWAIT XSYNC
        metapass="Xxxx123" metaserver="localhost" metauser="sasadm@saspw"
        metarepository="Foundation" metaport=8561;
%put _all_;


/*
/* Modify these parameters for your system
*/
%let sasprofile=SASAdmin;
%let admuser=sasadm@saspw;
%let admpass=Xxxx123;
%let package=D:\MyDemo\WorkflowPortlet\PromoteReports.spk; /* name and path for package file */
%let exportlog=D:\MyDemo\WorkflowPortlet\batchexport.log; /* location for export log */
%let importlog=D:\MyDemo\WorkflowPortlet\batchimport.log; /* location for import log */


/*
/* Create export package file based on specified report in portlet
/*
/*     &okreport - report name and location is passed from portlet
*/

x del &exportlog;

data _null_;
    cmd='C:\Progra~1\SASHome\SASPlatformObjectFramework\9.3\ExportPackage -profile "' ||
"&sasprofile" || '" -package "' ||
         "&package" || '" -objects "' || "&okreport" || '(Report)" -types Report -log "' ||
"&exportlog" ;
    put "Executing command::" cmd=;
    call system(cmd);
run;

/*
/* Import package file to target folder
/*
/*     &target - target location is passed from portlet
*/

%let impcommand = C:\Progra~1\SASHome\SASPlatformObjectFramework\9.3\ImportPackage -profile
"&sasprofile" -package "&package"
                            -target "&target" -types Report -log "&importlog" ;
x del &importlog;
x &impcommand;
run;

/*
/* Two steps to delete the report from the original location. Delete metadata then DAV content.
/*
/*     &deleteID - report ID is passed from portlet
*/

proc metadata
 in=
"<DeleteMetadata>
   <Metadata>
      <Transformation Id='&deleteID'/>
  </Metadata>
   <NS>SAS</NS>
   <Flags>268435456</Flags>
```

```
   <Options/>
</DeleteMetadata>"
;
run;

/*
/* Fix up URL for report to delete from webDAV - i.e., replace spaces with %20
/*
/*     &okreport - report name and location is passed from portlet
*/

%let rptURL=%qscan(&okreport,1,%str( ));
%macro okURL(string);
%let word cnt=%eval(%sysfunc(count(%cmpres(&string),%str( )))+1);
  %do i = 2 %to &word_cnt;
    %let word=%qscan(&string,&i,%str( ));
        %let rptURL=&rptURL%20&word;
        %put &rptURL;
  %end;
%mend;

%okURL(&okreport);
 run;

proc http
     method="DELETE"
     url="http://localhost:8080/SASContentServer/repository/default/sasfolders&rptURL(Report)"
     webusername="&admuser"
     webpassword="&admpass";
run;

%let rc = 0;

*%let okreport=/WorkflowPortlet/Development/Copy of Class Sample Report.srx;
*%let deleteID=A5JY95XW.B10002PV;
*%let target=/WorkflowPortlet/Production;
*%let okreportURL=&okreport; /* needs HTML encoding */
```