SAS® Workflow Studio 1.1: User's Guide

SAS® Documentation

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2011. *SAS® Workflow Studio 1.1: User's Guide®*. Cary, NC: SAS Institute Inc.

SAS® Workflow Studio 1.1: User's Guide

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, July 2011

Contents

Chapter 1
Introduction to SAS Workflow Studio

What is Business Process Management?

Organizations of all sizes oscillate between conditions of chaos and order, with the forces of leadership and purpose acting to bring order, and the opposing forces of a changing marketplace, technological advances, and shifting customer priorities tending to create chaos. In other words, all successful organizations undergo ceaseless struggle to achieve and sustain the efficient conversion of their competencies and resources into received value for their customers. Success requires a delicate balance between establishing efficient, repeatable processes and maintaining the agility to adjust or completely replace these processes to fit current conditions.

Common challenges include the following:

People acting in concert

The actions of good managers, along with prior training and experience, largely determine how effectively members of a group can work together to bring about an aggregate result. Yet all of these factors take time to develop, and might never fully develop if the pace of change is high. A well designed process management system can help by orchestrating—by way of notifications, reminders, delivery of resources, and tracking—the work of many individuals involved in a business process. It can also automate much of the start up (for example, finding the right forms, locating relevant policies and procedures, and so on) and cleanup (for example, forwarding on to the next person in the process) in each individual activity.

Interleaving automation

Not all processes can be usefully automated, but even partially automated processes are more efficient than completely manual processes. A well designed process management system can help identify where automation can have the highest impact, along with an operational framework for deploying and managing automated processes.

Performance analysis and optimization

High-level summary results can indicate problems, but detailed analysis is required in order to pinpoint and fix bottlenecks and inefficiencies in operations. A properly implemented process management system can collect detailed metrics on actual performance of key processes in real time, giving management a concrete basis for making decisions about how and when to make improvements.

Business process management (BPM) is a disciplined approach focused on aligning all aspects of an organization on fulfilling the needs of its clients. It emphases integrating technology into the business process such that the process itself drives the business goals decoupled from the underlying systems and applications. In other words, IT plays a supporting role that allows for continuous process improvement in a flexible, technology-agnostic approach. Specifically, BPM advocates an emphasis on how the work is done within an organization, in contrast to a product's focus on what.

Moreover, BPM can be used to understand relationships between processes – both within the organization and across organizational boundaries – which, when included in a process model, allow sophisticated, horizontal reporting and analysis.

Critical success factors for BPM include the following:

- understanding the current state business process and client needs
- applying governance and standards based on business policies and practices
- using metric and key performance indicator (KPI) definitions that support measurable business goals

More specifically, a business process is a collection of activities designed to produce a specific output for a particular objective, possibly involving both human and system interactions. Essentially, a process is an ordered sequence of work activities defined with respect to time and place, with a beginning, an end, and clearly defined inputs and outputs: a structure for action.

Initially, BPM focused on automation of business processes, but it has evolved to integrate manual processes in which human interaction takes place in series or parallel with the use of technology. For example, in basic workflow systems, when individual steps in the business process require human intuition or judgment to be performed, these steps are assigned to appropriate members within the organization. Consequently, the difference between workflow and BPM is not distinct. Generally, workflow management is considered to be a subset of BPM that emphasizes static routing and administration of human tasks. In contrast, a business process might include a combination of automated and manual activities with dynamic routing based on embedded business logic. Today, many products include varying aspects of customization and control, but both approaches emphasize the elimination of bottlenecks, minimization of redundancies, and improved operational efficiency.

In short, workflow systems can be thought of as a sort of operating system for the enterprise, whose function is to orchestrate and track work, whether automated or carried out by humans. In the same way that databases capture what an organization consumes and produces, workflow systems encapsulate how the organization operates.

Why SAS Workflow?

The integration and interoperability of applications and data has improved with the emergence of the Web, middleware technologies, enterprise application integration (EAI) efforts and adherence to software standards (for example, Java, J2EE, and XML). However, businesses manage by process driving the continued adoption of BPM across many industries. Popular workflow features include the following:

- systematic routing of tasks requiring manual intervention
- automated triggering of basic actions and alerts
- centralized reporting and audit functions

Process management systems like SAS Workflow offer the option to define, automate, audit, and refine business operations by leveraging the Web, middleware and standards to more efficiently and effectively manage by process. The SAS Workflow suite provides the tools to rapidly integrate fundamental process management into business operations and business offerings based on SAS solutions and products.

Chapter 2

SAS Workflow Architecture

Overview

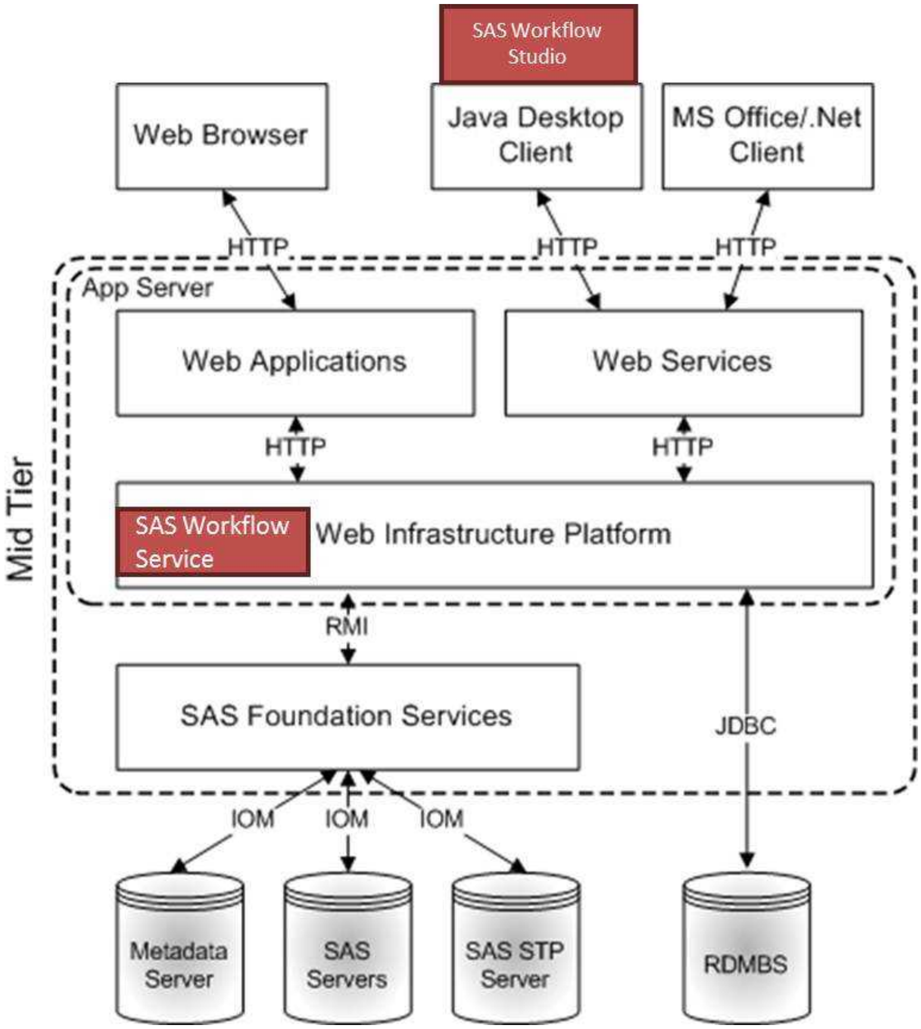
The SAS workflow architecture provides a suite of applications and services in the SAS Web Infrastructure Platform (WIP) that work together to model, automate, integrate, and streamline business processes, providing a platform for more efficient and productive business. SAS workflow consists of the following 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.

The following figure shows how the SAS workflow components fit into the SAS platform architecture:

Figure 2.1 SAS Workflow Architecture



SAS Workflow Studio

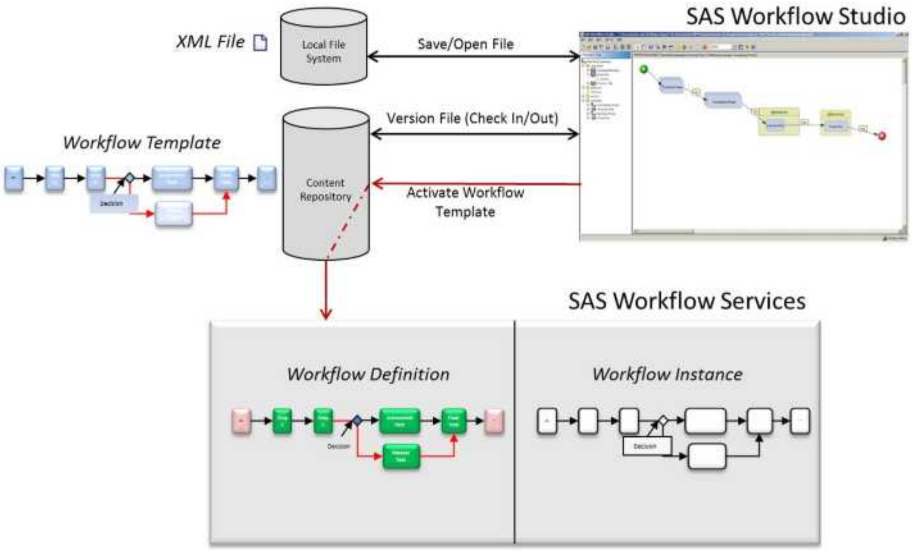
SAS Workflow Studio is designed to quickly build, organize, and reorganize the workflows and business logic at the heart of a process integration application. SAS Workflow Studio handles complex business or application process logic and many common workflow and process modeling patterns. Process designers 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, which can be versioned using SAS Content Services or activated using the SAS Workflow Services for instantiation by workflow-enabled applications. 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 use 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 might 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.

Figure 2.2 *Workflow Definition Management*

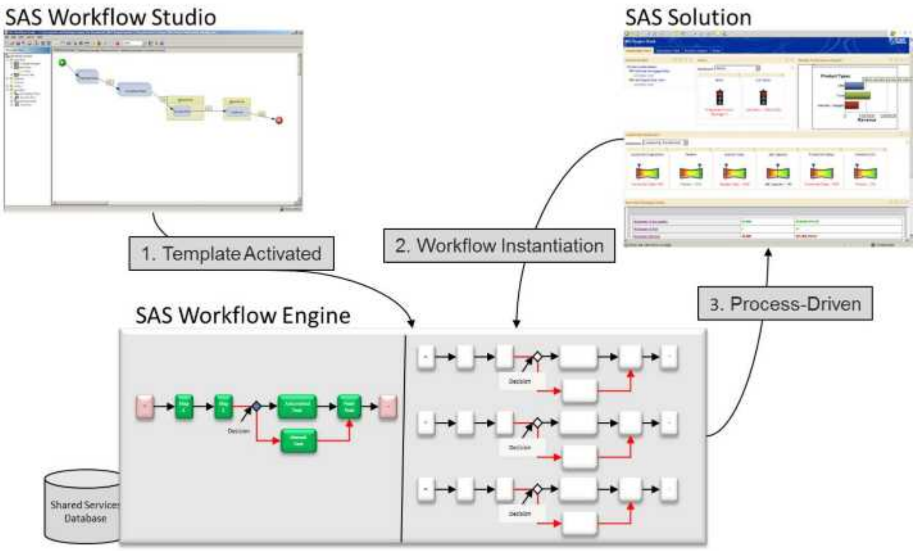


Note: Process changes made in SAS Workflow Studio do not take effect until the new version is uploaded and activated. Once activated, any new processes started use the newly activated version while any currently running process instances continue using their respective version of the workflow definition. Refer to "Deploying and Maintaining Processes" on page 66 for details.

Together, the SAS Workflow Studio and SAS Workflow Services provide the ability to manage workflows that can be leveraged by solutions—including CRM, manufacturing, health care, and manufacturing—as platform components for process automation. Solutions integrate with 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.

The following figure illustrates the lifecycle of a workflow from its creation in SAS Workflow Studio to its instantiation in an application.

Figure 2.3 Workflow Instance Management

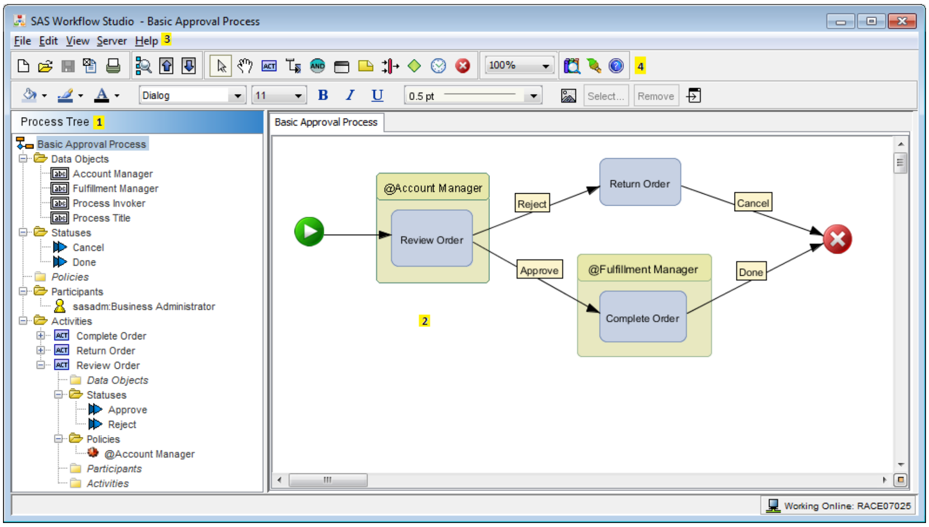


Chapter 3
Navigating SAS Workflow Studio

Overview

The SAS Workflow Studio user interface consists of two main content panes, the process tree and the diagram editor. The process tree acts as a content hierarchy pane, organizing the process elements associated with the current diagram open in the editor. Menus and a toolbar can be used to build workflow diagrams quickly with minimal effort.

Figure 3.1 SAS Workflow Studio Layout



1 Process Tree

2 Diagram Editor

3 Menus

4 Toolbar

Menus and Toolbar

Menus

File Menu

The **File** menu provides commands for file manipulation. Some of these commands are also included on the toolbar for quicker access.

New (Ctrl-N)
creates a new, blank workflow template.

Open (Ctrl-O)
opens a workflow template that is stored locally in a file.

Save (Ctrl-S)
saves the active template to a file.

Save As (F12)
saves the active template to a file and prompts you for a filename.

Close (Ctrl-F4)
closes the active template.

Close All Files
closes all open workflow templates.

Print (Ctrl-P)
prints the active template.

Properties
opens the Properties window to display the properties of the active template.

Options
opens the Options window in which you can set default options for SAS Workflow Studio.

Recent Files
lists recently opened workflow templates.

Exit
closes SAS Workflow Studio.

Edit Menu

The **Edit** menu provides commands for manipulating objects. Some of these commands are also included on the pop-up menus in the project tree and diagram editor for quicker access.

Edit Properties
edits the properties of a selected object.

Cut
copies the selected object to the clipboard and removes it from the drawing pane.

Copy
copies the selected object to the clipboard without removing it from the drawing pane.

Paste
pastes the current clipboard contents to the drawing pane.

Delete
removes the selected object from the drawing pane.

Add to Clip Explorer
adds the selected object to the default folder of the Clip Explorer.

View Menu

The **View** menu provides commands to enable and disable the SAS Workflow Studio utilities: process tree, format toolbar and Clip Explorer. Some of these commands are also included on the toolbar for quicker access.

Process Tree
contains the definitions of the process and all of its data objects, statuses, policies, participants, and activities organized the elements by type and associated activity.

Format Toolbar (F10)
is used to configure graphics properties of the diagram.

Clip Explorer (F11)
stores process elements as a library of symbols and process objects that can be reused using the copy and paste commands.

Server Menu

The **Server** menu provides commands for managing workflow templates between SAS Workflow Studio and the SAS Content Repository. Some of these commands are also included on the toolbar for quicker access.

Log On (F9)
logs in to the SAS Platform.

Log Off (Alt-F9)
ends the connection with the SAS Platform.

Save to Repository (F7)
uploads a process template to the SAS Content Repository.

Open from Repository (F5)
retrieves or checks out a copy of a process template from the SAS Content Repository.

Manage Processes (F3)
displays a list of process templates currently available in the SAS Content Repository and facilitates management of active process definitions.

Help Menu

The **Help** menu provides access to product documentation and configuration information

Help Topics (F1)
opens the online Help for SAS Workflow Studio.

SAS on the Web
provides links to additional resources on the SAS Web site.

About SAS Workflow Studio
displays product information for SAS Workflow Studio.

Drawing Tools

The drawing toolbar provides a set of controls that can be used to construct or edit a workflow template. Select the corresponding button to activate a command. When you want to select an element in the diagram editor (to move it, or for access to the right-click menu), use the Select Tool button in the toolbar.

Select Tool (space bar)
selects an object on the editor pane before moving it or right-clicking it.

Hand Tool (H)
navigates the editor pane and acts as a scroll bar.

Add Activity (A)
adds a new activity to the diagram.

Add Sequence flow Line (C)
connects two activities together with an optional status.

Add Logic Gateway (L)
adds a logic gateway (AND or OR) that controls the process flow behavior of the connected activities.

Add Swimlane (W)
sets the participants of the contained activities.

Add Annotations (N)
adds annotations to the diagram.

Add Merge/Fork Gateway (J)
represents process forking or joining to denote parallel processing paths.

Add Decision Gateway (D)
adds a Decision gateway to the diagram to denote alternate processing paths.

Add Timer (T)
adds a Timer node to the diagram.

Add Stop Node (S)
adds a Stop node to the diagram.

SAS Workflow Studio supports several predefined keyboard shortcuts to create multiple objects of the same type. If a shortcut key is pressed followed by a left mouse click in the diagram editor, then the associated object is created. For example, if you press the A key and then click the mouse in the drawing pane, an Activity node is created to the right of the mouse pointer location. If you press the W key, then you can use the mouse to draw the boundaries of the swim lane.

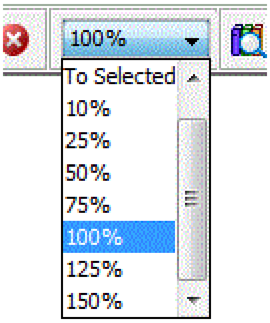
In addition, the following movement key bindings are available:

- The left, right, up, and down arrows move a selected object one pixel in the corresponding direction.
- The same keys in combination with the Shift key move the object 8 pixels in the corresponding direction.
- The same keys in combination with the Alt key move the object 18 pixels in the corresponding direction.

Note: These key bindings cannot be customized.

Zoom Toolbar

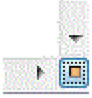
SAS Workflow Studio includes a zoom toolbar that allows users to zoom the current diagram in or out. It consists of a drop-down box with various zooming factors. Choosing a zoom factor of 100% yields a 1:1 magnification ratio. A higher zoom percentage zooms in to the diagram while a lower zoom percentage zooms out from the drawing.



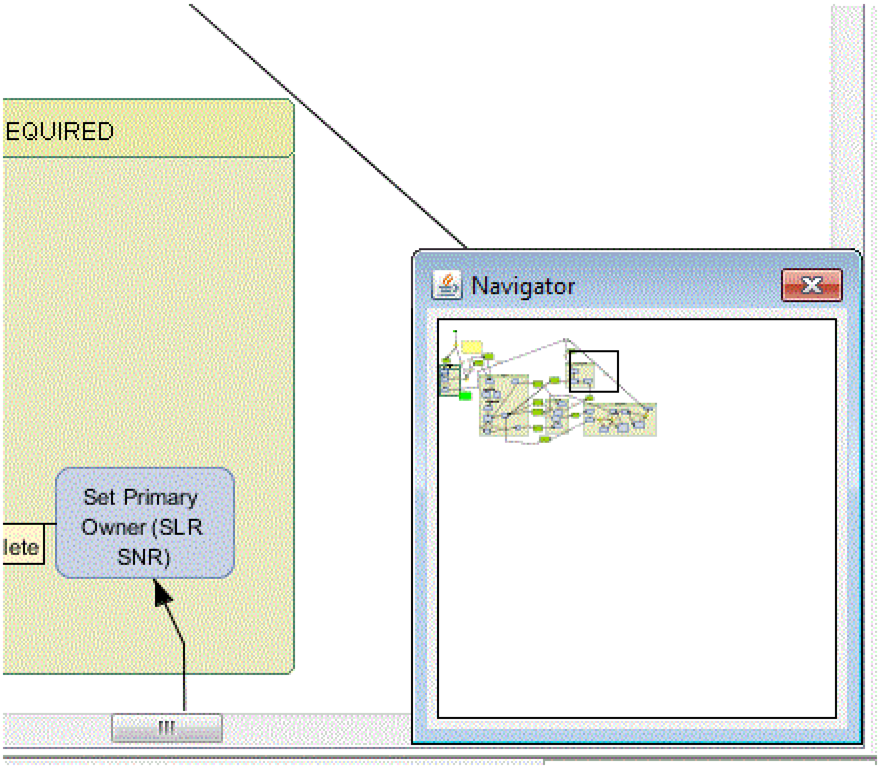
Navigator Pane

In addition to the zoom toolbar, the navigator pane is another useful visual aid. You activate this feature by clicking the small button at the lower right hand corner of the drawing pane.

Figure 3.2 Navigator Pane Button



The navigator pane displays a scaled down version of the entire drawing pane with the currently viewable section of the drawing highlighted. Hold the left mouse button and drag the mouse pointer over the navigator pane to pan across the entire main drawing page.



Using the Drawing Editor

Overview

Using SAS Workflow Studio's diagram editor, the process analyst can graphically design the relevant sequence of activities that comprise the workflow.

Elements of a Process

Overview

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 (ACT) 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 while 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 transition 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

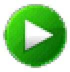
Participants drive process access and authorization by linking the workflow authorization roles to the SAS platform users, groups, and organizational roles.

Process Diagram Elements

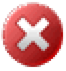
Global data objects, policies, statuses, and participants are associated with the top level folders under the workflow root. Activities can also contain locally defined data objects, policies, statuses, and participants. Local elements exist in the context of a specific activity are accessible only at 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 recreated 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 and Stop node, but a single Stop node might terminate multiple activities. Likewise, the Start node can be used to initiate multiple activities.

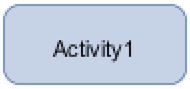
The following elements can be used in a process diagram:



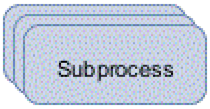
The Start node must precede the first activity in the process.



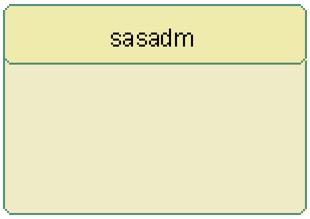
The Stop node must be connected to any activity that leads to process termination.



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

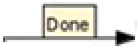


An activity element with a stacked appearance represents a subprocess. A subprocess contains one or more activities that might, in turn, represent subprocesses resulting in a process hierarchy. You can create subprocesses and edit the contained activities from the drawing editor.

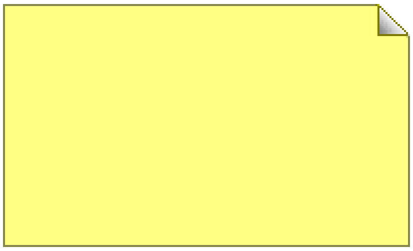


Swimlane elements are used in SAS Workflow Studio to group activities assigned to the same participant definition. They can be explicitly assigned to a Participant object or they can be implicitly assigned via a swimlane policy. The swimlane policy derives the user, group, or role value defined by the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

The Sequence Flow tool is used to connect process elements.



This connection element might also be used to designate process status (that is, state changes or transitions between activities).



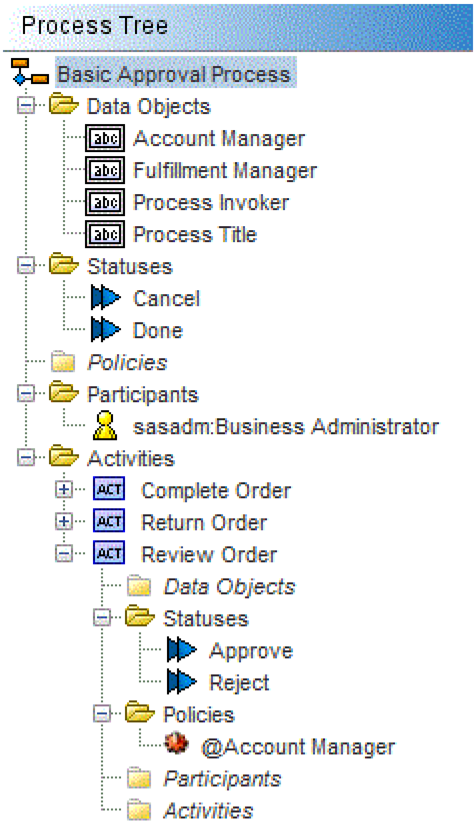
Annotations are used to hold additional information. These notes are for presentation only and are not associated with the run-time process definition.

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 might use the process tree pop-up menus to add activities and other process elements.

Using the Process Tree

The process elements (activities, data objects, statuses, policies, and participants) for the process definition open in the diagram editor are accessible from the process tree. The tree uses a familiar hierarchical folder structure to organize the elements by type and associated activity. You can click on a folder to open it, view the contents and access the attributes for each object.

Figure 3.3 Process Tree



The process folder contains top-level folders for data objects, statuses, policies, participants, and all of the associated activities. Each activity folder can contain local definitions for each of these object types. Whenever you create or edit an object, you have the option to make it available globally (in the subtree) or not.

In the process tree, you can right-click on a folder and then select **New** to create the relevant object based on the folder selected. You can also edit or delete existing elements using pop-up menus.

Using the Clip Explorer

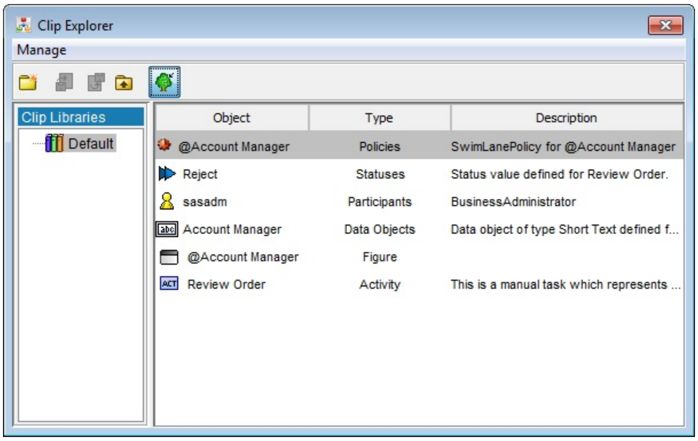
The Clip Explorer utility can store a library of symbols and process objects that can be shared using the copy and paste commands. Copying figures and process tree objects to the clipboard creates a reusable copy of these objects. That is, if an object from the Clip Explorer is copied into a drawing or a process tree, the properties of the new instance can be edited independently, thus maintaining the properties of the original. The object library can be organized within the clip explorer by copying or moving specified objects to different folders. Double-clicking an object opens the corresponding property editor dialog box. In addition, you can share library elements by exporting an entire folder of objects as an XML file or can access another process analyst's library by importing an existing Clip Explorer file into a folder of objects.

The Clip Explorer has two main panels:

- a folder tree, which is a hierarchical representation of the clip library folders

- a content panel that displays the contents of the currently selected folder in the folder tree

Figure 3.4 *Clip Explorer*



Object	Type	Description
@Account Manager	Policies	SwimLanePolicy for @Account Manager
Reject	Statuses	Status value defined for Review Order.
sasadm	Participants	BusinessAdministrator
Account Manager	Data Objects	Data object of type Short Text defined f...
@Account Manager	Figure	
Review Order	Activity	This is a manual task which represents ...

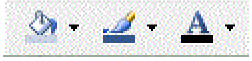
The Clip Explorer can be toggled on and off by pressing F11, or, by selecting **View** ⇨ **Clip Explorer**. To create a new subfolder in the folder tree, right-click the parent folder and choose **New Subfolder** from the resulting pop-up menu. To delete a folder, right-click on the target folder and then select **Delete**. Alternatively, you can select the relevant folder and press the Delete key from the keyboard.

To edit an object, right-click on the object and then select **Edit** from the resulting pop-up menu that opens the corresponding property dialog box to make the necessary changes.

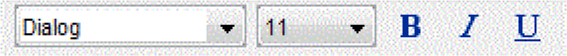
Using the Format Toolbar

The format toolbar provides additional formatting properties for process definitions. It contains the following four main control groups:

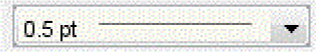
Color Property Controls



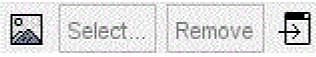
Text Property Controls



Outline Property Controls

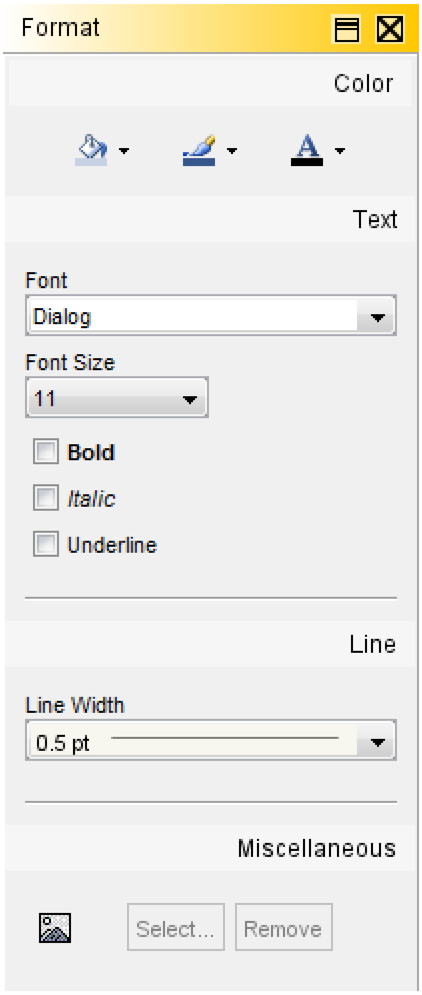


Miscellaneous Property Controls



The format toolbar can be toggled on and off by pressing F10, or, by selecting the **Format Toolbar** check box in the **View** menu. The format toolbar can also be displayed as a panel by selecting the Switch to panel mode icon ( ) in the Miscellaneous Property Controls group.

Figure 3.5 Format Toolbar in Panel Mode



Font: Dialog

Font Size: 11

□ **Bold**

□ *Italic*

□ Underline

Line Width: 0.5 pt

Note: Standard cut and paste operations do not maintain any special formatting. Formatted objects should be reproduced using the Clip Explorer.

Chapter 4
Defining Processes with SAS Workflow Studio

Creating a Process

The basic steps to create a process are:

1. Start SAS Workflow Studio.

   By default an untitled process opens that contains a Start node, an End node, two data objects: Process Invoker and Process Title and several status values: Scheduled, Cancel, Okay, Overdue and Done.

   Note: If SAS Workflow Studio is already started, then you can start a new process definition by selecting **File** ⇨ **New** from the menu bar, selecting the New File icon (🗋) in the toolbar, or pressing Ctrl-N.

2. In the process tree, select the root process node (Untitled1), right-click, and then select **Edit** from the resulting pop-up menu.

   The Edit Process dialog box appears.

3. In the **Process Name** field, replace **`Untitled1`** with the desired name for the new process.

4. Optionally, add a description for the process in the **Description** field.

5. Add and configure the required activities, data objects, statuses, policies, and participants.

6. Connect the process elements.

Note: Any workflow template changes take effect on the server once the revised process definition is uploaded and activated. If an earlier version of the process is already running, then it continues with the corresponding definition, but all new process instances started use the newly activated version.

The following sections provide details about each type of process element.

Working with Activities

Adding Activities

To define a new activity, follow these steps:

1. Start SAS Workflow Studio and either open a saved process, download a process from the content repository, or create a new process.

2. In the Studio toolbar, select the Add Activity icon (ACT).

3. Click anywhere on the drawing panel to create an activity.

4. Move the activity to the desired position on the drawing panel by clicking the selection tool (↖) on the toolbar, selecting the desired activity, and dragging it to the new location.

The default name of the first activity is Activity0. The second activity is named Activity1, and so on.

Connecting Activities in a Sequential Flow

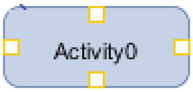
To define a logical sequence of activities, you connect them in the desired order. You can also assign a status. Refer to “Working with Statuses” on page 26 for details.

To link two activities together, follow these steps:

1. Select the Add Sequence Flow icon ( ) from the toolbar.

   The cursor changes to a cross (+) and a four-sequence flow anchor blinks when the cursor is in the proper position to make the connection.

   Figure 4.1 *Flow Anchors for an Activity*

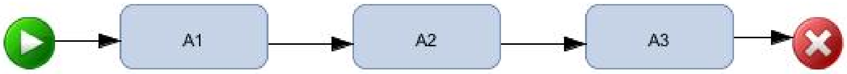
   

2. In the drawing panel, drag the mouse pointer from the first activity figure toward the second activity figure.

   A line and arrow illustrates the direction of the link, indicating the sequence of the process logic.

For example, a process flow containing a set of activities, Activity (A1, A2, A3) that executes such that A1 executes first, followed by A2 upon completion of A1, and so on, until A3 completes to terminate the process, could be modeled as follows:

Figure 4.2 *Sequential Process Flow*



Editing Activities

To edit an activity, follow these steps:

1. Open the Edit Activity dialog box by double-clicking the activity.

   Alternatively, select the activity on the diagram or in the process tree and right-slick to select the **Edit** option in the resulting pop-up menu.

2. Enter the desired name in the **Activity Name** field.
3. Optionally, add a description for the activity in the **Description** field.
4. Optionally, enable the desired notifications, which are generated using the SAS Notification Service.
5. Optionally, add localized versions of the name and description.

   For more information, see “Text Localization” on page 52.

6. Optionally, add custom attributes.

For more information, see “Custom Attributes” on page 52.

7. Select **OK** to save the updated activity definition.

Deleting Activities

To delete an activity, follow these steps:

1. Right-click on the desired activity figure in the drawing panel or process tree and then select **Delete** from the resulting pop-up menu.

   Alternatively, select the activity directly in the drawing panel or process tree and press the Delete key or Ctrl-X.

2. Select **Yes** in the confirmation dialog box to permanently remove the selected activity.

The deleted activity is no longer visible in the drawing panel or the process tree.

TIP To delete multiple activities and the links between them, drag a selection box around the desired elements or use the Shift-click selection technique and then press the Delete key.

Note: All locally defined data objects, statuses, participants, and policies are also deleted when the containing parent activity is deleted. If you attempt to delete a global element that is referenced elsewhere in the process, then you are notified that it should not be deleted until all references are removed.

Note: SAS Workflow Studio does not currently support the undo or Ctrl-Z operation.

Adding a Subprocess

Processes contain one or more tasks that each represent a step or unit of work in the workflow structure. Grouping elements together into a process can also help to organize and reuse business logic. A subprocess is a process contained within a parent process that can be used to refactor a larger process into smaller components. Using subprocesses improves readability of the workflow definition and promotes consistent reuse of process logic within the organization.

To create a subprocess, follow these steps:

1. In the drawing panel, right-click on the activity that you want to convert to a subprocess and then select **Create Subprocess** from the resulting pop-up menu.

2. When prompted, select one of the following options:

   - **Create New**
   - **Load from file**
   - **Open from workflow repository**

3. Select **OK** to save the subprocess definition.

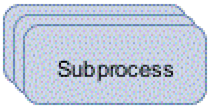
For each option, the relevant diagram opens. For the **Create New** selection, the diagram is blank and untitled. For the other options, the diagram is a copy of the designated existing template.

The original parent process elements remain visible in the process tree.

Once an element is added to the new diagram, the relevant activity becomes denoted as a process ( ) in the process tree. The subprocess data objects, statuses, policies, and participants are also visible in the process tree hierarchy.

After adding a subprocess, the activity symbol in the parent process becomes a stacked icon representing a set of activities rather than a single step in the process. The following figure shows an example of subprocess notation:

Figure 4.3 *Subprocess Symbol*

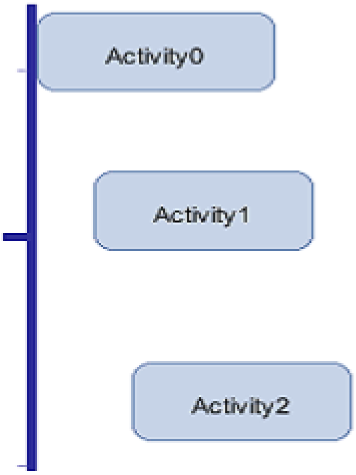


Aligning Activities

Multiple activities can be aligned using the alignment tool.

To activate the alignment tool, follow these steps:

1. Hold down the Ctrl key and click the mouse. A blue cross (+) appears on the drawing pane.
2. Drag the cross vertically or horizontally to form an alignment bar.



3. Toggle between the following alignment modes by pressing the space bar:
   1. Space evenly
   2. Pack tightly
   3. Spread out
   4. Original

Working with Statuses

Overview

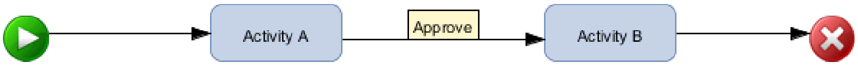
Statuses are used to denote the outcome of a process step as values associated with a logical transition from one activity to another. These values appear as labels on the flow connections and they represent the condition, which must be met in order to realize the transition.

By default, when no status is added, the task completes and the subsequent task is automatically started. This default value is represented as the (FINISHED) state in the status assignment menu, but does not explicitly appear on the transition as a label. Predefined values are **`Cancel`**, **`Done`**, **`Okay`**, **`Overdue`**, and **`Scheduled`**. You can also define custom status values.

If a status is specified on the connection, then the process advances to the next activity in the sequence only when that status changes to the associated value causing the current activity to complete.

For example, the process contains two activities, A and B, with a connection from Activity A to Activity B. If Activity B is initiated once Activity A has been approved, then we can represent it by assigning an Approve status to the connection between Activity A and Activity B, as shown in the following figure:

Figure 4.4 *Status Example*



Adding Statuses

To define a new status, follow these steps:

1. In the process tree, right-click on the **Statuses** folder and then select **New Status** from the resulting pop-up menu.

   Alternatively, you can right-click an activity in the diagram editor and then select the **New Status** option from the resulting pop-up menu.

2. In the New Status dialog box, enter a name for the status in the **Status Name** field.
3. Optionally, add a description for the status in the **Description** field.
4. Optionally, add localized versions of the name and description.

   For more information, see “Text Localization” on page 52.

5. Optionally, add custom attributes.

   For more information, see “Custom Attributes” on page 52.

6. Choose the scope of the value using the **Visible in entire subtree** check box. This makes this status definition accessible to other activities in this process.
7. Click **OK** to save the new status definition.

Editing Statuses

To edit an existing status definition, follow these steps:

1. In the process tree, open the **Statuses** folder and then right-click on the desired status node.
2. Select **Edit** from the resulting pop-up menu.
3. Change the desired values in the Edit Status dialog box.
4. Select **OK** to save the updated status definition.

Deleting a Status

To completely remove a status value from the workflow, follow these steps:

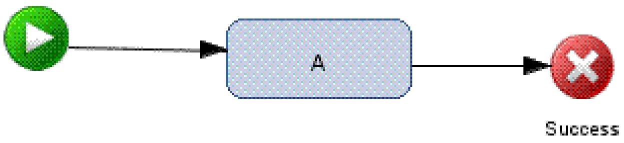
1. Right-click on the status in the process tree and then select **Delete** from the resulting pop-up menu

   Alternatively, you can select the status directly in the process tree and press the Delete key or Ctrl-X.
2. Select **Yes** in the confirmation dialog box to permanently remove the selected status.

Assigning a Status

To assign a status, right-click on a connection between activities and then select **Change status** from the resulting pop-up menu. This opens a submenu containing the status values defined for the workflow. If the desired status is not defined, then you can select the **New status** menu option to define the new status value.

The **Change status** menu does not appear on the pop-up menu for the connection between the Start node and the first activity. However, a status value can be assigned to a connection to the Stop node, as shown in the following figure:



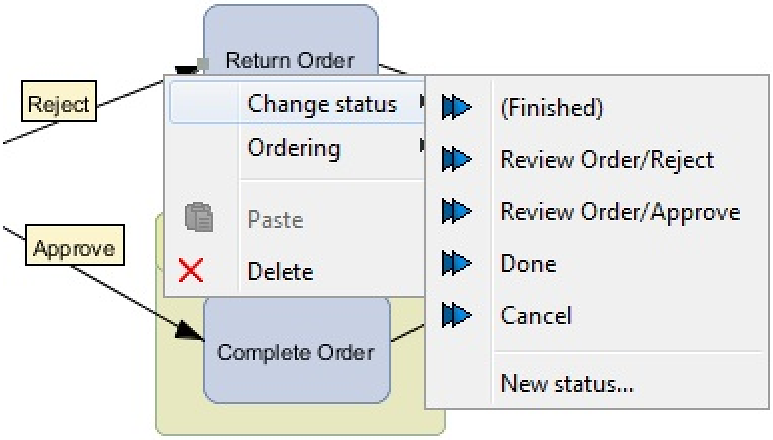
TIP You can use the Selection tool ( ) to reposition the activity figures in the diagram editor to ensure that the statuses are visible. Also, you can reposition the connection endpoints by selecting the target connection. Then, select the endpoint and drag it to the new location in the figure by releasing the mouse.

Local Statuses

When you define a status locally for an activity, not for the whole process, SAS Workflow Studio displays the relative label path. A label path is an alternative way to refer to an object when using the object ID is not feasible. A label path contains a sequence of strings. Each string is the label of a parent or ancestor of the object except for the last label, which corresponds to the object itself .

The following example shows the global statuses Done and Cancel, which are defined for the Approval process. The Review Order activity includes the statuses Approve and Reject, and their label paths include the object's ancestors.

Figure 4.5 Local Status Values



Working with Participants

Overview

As mentioned previously, a workflow often requires user interaction with the system to complete a task. In addition, automation of task assignment within the process to a particular set of users might be desired. These tasks are assigned to the appropriate members within the organization using specific roles. SAS Workflow uses participants to assign the persons (users, groups and organizations) potentially involved with the workflow, or a specific activity, as role-based permissions for accessing workflow data. The following standard workflow roles can be used in either participant or swimlane definitions:

Task Initiator
: is the person who starts the task instance. By default the person who starts the workflow instance is included as the task initiator.

Potential Owner
: is a person who can access the task to claim and complete it. A potential owner becomes the actual owner of a task by explicitly claiming it.

Excluded Owner
: is someone who cannot become an actual or potential owner and thus cannot claim or perform the task. All excluded owners are implicitly removed from the set of potential owners.

Actual Owner
: is the individual who performs the task. A task can have only one actual owner. When the task is claimed, the actual owner can complete the task, release the claim, or transfer or delegate the task.

Task Stakeholder
: is anyone with an interest in the task (that is, monitoring its progress).

Note: The Task Stakeholder role is not currently used by SAS workflows.

Business Administrator
: is someone who can influence the progress of a task by adding comments, delegating or transferring the task, or releasing the task locked by another user.

Notification
: is someone who receives notifications when events happen (for example, missing a deadline or reaching a milestone). A notification does not have to perform any action when notified. The notification is purely informational and is an execution requirement. This is in contrast to an actual owner or potential owner, who must perform a specific action in order to complete an assigned task.

By default, these workflow roles have predefined permissions and are used to assign items to the worklist for individual users of the system.

Adding Participants

To define a new participant, follow these steps:

1. In the process tree, right-click the top-level or local **Participants** folder and then select **Add Participant** from the resulting pop-up menu.

   Alternatively, right-click an activity in the diagram editor and then select the **New Participant** menu option.

2. In the Edit Participant Properties dialog box, enter the desired values.

3. For **Participant's Role in Process**, choose the relevant role from the predefined workflow roles.

4. For **Identity**, select the **User**, **Group**, or **Organizational Role Name** option and enter the desired value.

5. Optionally, add localized versions of the name.

   For more information, see “Text Localization” on page 52.

6. Select **OK** to save the new participant definition.

Participants can be also assigned using swimlanes. For details, see “Assigning Participants” on page 30.

The process tree displays both the identity name and associated workflow role. For example, if the user name is `admin` and the participant's role is Task Initiator, the tree shows `admin: Task Initiator`.

Editing Participants

To edit an existing participant definition, follow these steps:

1. In the process tree, open the **Participants** folder and then right-click on the desired participant node.
2. Select **Edit** from the resulting pop-up menu.
3. Change the desired values in the Edit Participant Properties dialog box.
4. Select **OK** to save the changes.

Deleting Participants

To completely remove a participant from the workflow, follow these steps:

1. Right-click on the target participant in the process tree and then select **Delete** from the resulting pop-up menu.

   Alternatively, select the participant in the process tree and then press the Delete key or Ctrl-X.
2. Select **Yes** in the confirmation dialog box to permanently remove the selected participant.

Assigning Participants

Participants can be assigned to activities directly or indirectly via swimlane.

You can make a direct assignment by any of the following methods:

- Drag the participant from the process tree to the relevant activity in the diagram editor.
- Copy (to reuse the global definition) or drag (to demote it to a local definition) the participant element into the participant folder for the target activity in the process tree.
- Define a swimlane and associate it with the relevant participant definition. Then assign the activity by dragging it onto the swimlane.

To make an indirect assignment, follow these steps:

1. Define a data object that represents a participant user, group, or organizational role identity.
2. Create a swimlane and assign the value of the data object as the value.

   The data object acts as a placeholder (designated by the `@` preceding the data object name) for the value until assigned at run time.
3. Drag the activity onto the relevant swimlane in the diagram.

Note: Assigned participants are visible only in the project tree and cannot be accessed in the Edit Activity dialog box.

Configuring Participants

Run-time workflow participants 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, then the name must be a SAS platform user.
- If a group type is specified, then the name must be a SAS platform group.
- If an organizational role type is specified, then the name must be a SAS platform role.

Working with Policies

Overview

Often tasks require that actions be automatically triggered by specific process events. SAS Workflow provides a way to define these actions through policies. A policy object encapsulates executable business logic and, because an activity might initiate multiple actions simultaneously, multiple policies can be associated to a single activity. Policies are event-driven executable actions that can be configured to perform useful automation such as the following:

- Indicate how and when notifications of deadlines should be sent.
- Configure the process to notify the owner of a process of arbitrary task events such as task start or task stop.
- Configure the process to notify users of arbitrary task events such as task start or task stop with links to the task that has been changed.
- Integrate the system with other back-end systems.

The following events can be used to trigger policies:

Process Started
Generated when the process state changes to Started.

Process Finished
Generated when process state changes to Finished.

Status Addition
Generated when status is added to workflow process.

When no status is specified (when the default `---` is selected), the associated policy is executed for any status addition.

Status Removal
Generated when status is removed from workflow process

Timer Expired
Generated when a timer associated with workflow process triggers.

Data Object Updated
Generated when a data object is updated.

When no data object is specified (when the default `---` is selected), the associated policy is executed for any data object update.

Participants Updated
Generated when participants list of the process is updated.

Note: Process data should not be defined within a policy. Data objects should be used to hold relevant business data in the workflow and can be updated within policies, but, in general, policies should be limited to actions.

Note: Policies are asynchronous and are subject to timing for execution order. For example, if you use a policy to copy the value from a root data object to local activity data object, then you must ensure that the source object is set and not subject to change.

Policy Actions

Add Status to Process

The Add Status to Process policy is commonly used to add a status to the current workflow, thus automating itself based on the event trigger. This can include the Process Started event trigger, if the activity is designed to trigger other policy activities and then stop itself.

The following parameters can be defined for this policy:

Parameter Name	Description
Process	specifies the label path of the process or subprocess to which the status is added.
Status	specifies the label path of the status to be added.

Add Status to Process with ID

The Add Status to Process with ID policy is used to add the selected status to any process (including other process instances and activities) based on the ID value. The referenced data object can be updated while the workflow process is active.

The following parameters can be defined for this policy:

Parameter Name	Description
Process ID Data Object	specifies the label path to a data object that contains the process ID value of the target process.
Status Label	specifies the label path to the status that is added to the target process.

Copy Participants to Process

The Copy Participants to Process policy is used to copy participant values between processes.

The following parameters can be defined for this policy:

Parameter Name	Description
Source Process	specifies the label path of the activity from which the participant is copied.
Source Role	specifies the workflow role that is copied from the source activity.
Target Process	specifies the label path of the activity to which the participant is copied.
Target Role	specifies the workflow role for the target activity.

Note: Swimlanes also affect participant values (via Set Process Participant policy). So, if the activity that triggers the Copy Participant to Process policy transitions into a swimlaned activity, then a race condition occurs because all policies are asynchronous. In this case, the value is either the copied one or the swimlane one. You should avoid this situation or ensure the participant values match.

Copy Data Object

The Copy Data Object policy is used to copy values between data objects.

The following parameters can be defined for this policy:

Parameter Name	Description
Source Data Object	specifies the label path of the data object whose value is to be copied.
Target Data Object	specifies the label path of the target data object where the value is to be copied.

Copy Data Object to External Process

The Copy Data Object to External Process policy is used to copy data object values between processes.

The following parameters can be defined for this policy:

Parameter Name	Description
Source Data Object	specifies the label path of the local data object whose value is to be copied.
Target Process ID Data Object	specifies the label path to a data object that contains the process ID value of the target process.
Target Data Object Label	specifies the label path of the target data object where the value is to be copied.

Copy Data Object from External Process

The Copy Data Object from External Process policy is used to copying a data object from another process (remote or external) into the current process.

The following parameters can be defined for this policy:

Parameter Name	Description
Source Process ID Data Object	specifies the label path to a data object that contains the process ID value of the source process.
Source Data Object Label	specifies the label path of the external data object whose value is to be copied.
Target Data Object	specifies the label path of the target data object in the current process where the value is to be copied.

Extract from XML Data Object

The Extract from XML Data Object policy is used to extract the values from an XML type data object and assign it to another data object. XML data objects are commonly used as the output of a Web service invocation. Refer to “Invoke Web Service” on page 37 for details.

The specified XPath expression is applied to the XML value stored in the data object and the result is returned as text. If the result of XPath extraction is an array, then the first member in the array is returned.

The following parameters can be defined for this policy:

Parameter Name	Description
XML Data Object	specifies the label path to the data object that contains the XML value from which the value is extracted.
Output Data Object	specifies the label path to the data object to which the extracted value is stored.
XPath Statement	specifies the statement that is used to extract the desired text from the data object specified in the **XML Data Object** parameter.

In the following example, the XML data object has the value:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bookstore>
   <book>
      <title lang="en">Harry Potter</title>
      <author>J K. Rowling</author>
      <year>2005</year>
      <price>29.99</price>
   </book>
</bookstore>
```

If the policy specifies **`/bookstore/book/title`** for the **XPath Statement** parameter, then the policy stores the value **`Harry Potter`** in the target data object.

If the XML schema includes namespace prefix notations, then you should use the following format for the XPath expression:

```
declare namespace n =
 "http://support.sas.com/xml/namespace/biwebservices/webservicemaker-9.2";
/n:ListWebServicesResponse/n:ListWebServicesResult/n:string
```

HTTP Request

The HTTP Request policy is used to invoke an HTTP method using the information specified in the policy definition.

The following parameters can be defined for this policy:

Parameter Name	Description
Directive Name	specifies the name of the directive. The value can be specified using data object substitution. For more information, see "Data Object Substitution" on page 52. *Note:* If you are logged in to the SAS platform, then the ellipsis button (...) launches a directive picker.
URL	specifies the target URL to invoke, execute, or access. The value can be specified using data object substitution.
Parameters	specifies additional HTTP parameters. The value can be specified using data object substitution.
Method	specifies the HTTP method to execute. *Note:* POST is the only method that is currently supported.
Status Code Data Object	specifies an optional data object to bind to the return code of the HTTP request.

This URL can be an absolute (or complete) URL that includes the host and port, or it can be a relative URL. If a relative URL is specified, then the policy uses the same host and port that the Workflow Service is configured to use.

However, if the URL is not specified or if it identifies a data object that contains an empty value, then the **Directive Name** parameter is used to create the appropriate URL. The policy retrieves the named directive from the SAS Directive Service, and uses it to build the URL. If the parameters field is specified, then it is appended to the URL value. The policy execution always uses the trusted user identity. Refer to Appendix 2, "Policy Usage Examples," on page 79 for a detailed example.

Increment Data Object

The Increment Data Object policy is used to increment a numeric data object.

The following parameters can be defined for this policy:

Parameter Name	Description
Data Object	specifies the label path to a data object to be incremented.
Increment Value	specifies the numeric value by which the data object should be incremented.

Invoke SAS Code

The Invoke SAS Code policy is used to execute SAS code stored in a SAS program. The SAS program can return only a single value to the process.

The following parameters can be defined for this policy:

Parameter Name	Description
Source URI	specifies either a file URI or a URL that is the name of the file that contains the SAS code to execute. The value can be specified using data object substitution. For more information, see "Data Object Substitution" on page 52.
Logical Server Name	specifies the name of the Pooled Workspace Server to use to execute the SAS code. The value can be specified using data object substitution.
Repository Name	specifies the name of the metadata repository in which the Pooled Workspace Server is defined. The value can be specified using data object substitution.
Result Macro Variable	specifies the name of a global macro variable defined in the SAS code. The value can be specified using data object substitution.
Return Data Object	specifies the data object that is used to store the returned value.
Pass all root process data objects?	controls the scope of root data objects when executing the SAS code. If this parameter is checked, then each root-level data object is converted to a macro variable. Each macro variable is prepended to the SAS code before the code is submitted to the server for processing.

Parameter Name	Description
Parameter *n* Data Object Parameter *n* Macro Variable	specify up to five optional additional pairs of data object and macro variable values, where *n* is the sequence number for a list of values. The macro variable name is limited to 32 maximum characters. Values can be specified using data object substitution.

If you pass all the root data objects, then the macro variables are available for use within the SAS code. The macro name generation rules are as follows:

- All macro variable names are case-insensitive.
- A single underscore is defined as the default prefix character when forming macro variable names.

For example, if a root data object is defined with the name category and a value of sales then the following macro variable definition is submitted:

```
%let _category=sales;
```

By default, no prefix is added to the name and no default value is specified. To define a default prefix, follow these steps:

1. Open SAS Management Console and navigate to **Configuration Manager** ⇨ **SAS Application Infrastructure** ⇨ **Workflow Services 9.3** ⇨ **Properties**.
2. Select the **Advanced** tab and add the Workflow.SASCodeOperandPrefix property.
3. Assign the desired value to the new property.

For example, if the Workflow.SASCodeOperandPrefix is defined as **`wf_`**, then the following macro variable definition is submitted:

```
%let wf_category=sales;
```

If underscores exist in the data object name, then the resulting macro variable name retains each underscore even if an underscore is used as the first character. Thus, **`_quarterly_sales`** becomes **`wf__quarterly_sales`**. Spaces in data object names are replaced by double underscores. **`Quarterly Sales`** becomes **`_quarterly__sales`**. The total length of the macro variable name, including the prefix, cannot exceed 32 characters. Longer names are truncated to 32 characters.

The SAS code execution always uses the trusted user identity.

Refer to Appendix 2, “Policy Usage Examples,” on page 79 for a detailed example.

Invoke Web Service

The Invoke Web Service policy is used to invoke a Web service over SOAP/HTTP.

The following parameters can be defined for this policy:

Parameter Name	Description
Endpoint URL	specifies the URL of the Web service endpoint. *Note:* If you are logged in to the SAS platform, then selecting the ellipsis button (...) provides a list of all the registered SOAP Web services.
Service Name	specifies the name of the target SOAP service. If you specify a value for the **Endpoint URL** parameter, then this field should be left blank.
Action	specifies the SOAP action header value for the Web service.
Input Data Object	specifies the XML data object used for input. The associated schema parameter should be set from the WSDL for the Web service. The value of the data object should contain the body of the SOAP request message.
Output Data Object	specifies the XML data object used for output. The associated schema parameter should be set from the WSDL for the Web service. The value of the data object should contain the body of the SOAP response message.
Username	specifies the user name (optional). If a value is specified, then security level headers are added to the request at both transport level and message level.
Password	specifies the password (optional). If this parameter is specified, then it is used in the security level headers on the Web service request.

Note: This policy supports a single request object, so the value must be set as a single XML string. Data object substitution is not supported.

Note: This policy requires a single output object for storage of the Web service response. If the Web service does not define a response, then there is no way to verify that the invocation succeeded. In addition, if a fault occurs, then the response object is not set and the fault message is logged only on the server and is not delivered to the client.

Refer to Appendix 2, “Policy Usage Examples,” on page 79 for a detailed example.

Notify Participant

The Notify Participant policy is used to send a notification via the SAS Alert Notification Service to a participant as defined by their preferences. The notification can be sent when an activity starts or finishes.

The following parameters can be defined for this policy:

Parameter Name	Description
Subject	specifies the subject of the notification.
Message	specifies the body of the notification.

To create a policy that notifies a participant that an activity has started, follow these steps:

1. Select the target activity for the policy definition.
2. Right-click and select **Edit** to open the Edit Activity dialog box
3. Select the **Notify participant when activity starts** check box.

If you expand the activity's **Policies** folder in the process tree, then you should see the Workflow:Notify Participant policy definition. The notification is triggered by the Process Started event that is generated when the associated activity starts. When the policy executes, it evaluates the process to see whether the Actual Owner workflow role exists on the process. If the Actual Owner role is found, then the notification is sent to all users associated with that workflow role. If no Actual Owner workflow role is defined, then the policy execution then looks for the Potential Owner workflow role. If the Potential Owner role is found, then the notification is sent to all potential owners. In either case, the policy execution obtains the person or group defined for the workflow role, looks up that person or group in SAS metadata, and obtains the relevant notification settings as defined in the metadata.

To create a policy that notifies a participant that an activity has ended, follow these steps:

1. Select the target activity for the policy definition.
2. Right-click and select **Edit** to open the Edit Activity dialog box.
3. Select the **Notify owner when activity ends** check box.

If you expand the activity's **Policies** folder in the process tree, then you should see the Workflow:Notify Owner policy definition. The notification is triggered by the Process Finished event that is generated when the associated activity completes. The notification is sent to the owner of the process—the person in the Task Initiator workflow role. When a process instance starts, SAS workflow automatically adds the Task Initiator workflow role to the process, identifying that person as the original owner (or task initiator).

In addition to the predefined Notify Participant and Notify Owner policies, a process designer can create the policy directly and configure the policy using the Edit Policy dialog box accordingly.

Remove Status from Process

This policy is used to remove a status from the specified process.

The following parameters can be defined for this policy:

Parameter Name	Description
Process	specifies the label path for the target process or activity.
Status	specifies the label path for the status that is removed from the target process.

Schedule Process

This policy is used to schedule the initiation of a process.

The following parameters can be defined for this policy:

Parameter Name	Description
Process to Schedule	specifies the label path for the scheduled process.
Schedule Timer Data Object	specifies the data object with the expression that defines the start time.
End Timer Data Object	specifies the data object with the expression that specifies the end time (optional).

The Schedule Timer Data Object expression identifies when the scheduled process or activity should start. The timer value can be an exact date (Date or String) or a relative expression (String). An exact date must be specified in the form of **`MM`**/*dd*/*YYYY hh*:*mm*:*ss a* (for example, `12/31/2010 11:59:00 PM`). A relative expression can be used to configure the process to be started at a moment in time, relative to when the policy is triggered. The expression syntax supports seconds, minutes, hours, days, and weeks, as follows:

Expression	Interpretation
Now	Timer expires immediately
+1s	Timer expires in one second
+1m	Timer expires in one minute
+1h	Timer expires in one hour
+1d	Timer expires in one day
+1w	Timer expires in one week
+1h+1m+1s	Timer expires in one hour + one minute + one second

The optional **End Timer Data Object** parameter that identifies when the schedule timer should be stopped.

Send E-mail

This policy is used to send e-mail notifications via the SAS Mail Service using the information provided in the policy definition.

The following parameters can be defined for this policy:

Parameter Name	Description
To	specifies one or more recipient e-mail addresses. Separate multiple recipient addresses with commas.
From	specifies the sender e-mail address.
Subject	specifies the subject of the e-mail message. The value can be specified using data object substitution. For more information, see "Data Object Substitution" on page 52.
Message	specifies the body text of the e-mail message. The value can be specified using data object substitution.

Send Notification by Data Object

This policy is used to send notifications via the SAS Alert Notification Service based on the information stored in various workflow data objects.

The following parameters can be defined for this policy:

Parameter Name	Description
User Data Object	specifies a user defined in SAS metadata.
Subject	specifies the subject text of the notification message. The value can be specified using data object substitution. For more information, see "Data Object Substitution" on page 52.
Message Data Object	specifies the data object that contains the body text of the notification message.

When the policy executes, it retrieves the values from the specified data objects. The User data object value must identify the name of a user defined in the SAS metadata. The policy searches SAS metadata for this user and retrieves the relevant address defined for that user to send the notification.

Note: This policy is supported for legacy usage and does not leverage the SAS platform user preference settings regarding alert and e-mail support. Therefore, one of the other notification policies should be used.

Send Notification by Workflow Role

This policy is used to send notifications via the SAS Alert Notification Service based on the workflow role of a user.

The following parameters can be defined for this policy:

Parameter Name	Description
Role Data Object	specifies one of the supported workflow roles
Subject	specifies the subject text of the notification message The value can be specified using data object substitution. For more information, see "Data Object Substitution" on page 52.
Message Data Object	specifies the notification message body

When the policy executes, it retrieves the values from the specified data objects. The Role data object value must map to a valid workflow role: Actual Owner, Potential Owner, or Business Administrator. When the policy executes, it obtains the process' workflow roles, searches for the role identified by the data object value, and extracts the relevant user for that role. The policy searches SAS metadata for this user and retrieves the relevant address defined for that user to send the notification.

Note: This policy is supported for legacy usage and does not leverage the SAS platform user preference settings regarding alert and e-mail support. Therefore, one of the other notification policies should be used.

Send Workflow Group Notification

This policy is used to send a workgroup event notification, which triggers the SAS Alert Notification Service to generate end-user notification messages. This policy is used to send notifications to a group or set of recipients and where all recipients are required to be addressed together in a single message. This capability is useful in collaboration scenarios or where the event should be copied to all interested parties. E-mail is the only delivery channel supported using the group notification policy.

The following parameters can be defined for this policy:

Parameter Name	Description
Recipients(s)	specifies as primary recipients one or more users defined in SAS metadata. Separate multiple user names with commas.
CC Recipient(s)	specifies as copied recipients one or more users defined in SAS metadata. Separate multiple user names with commas.

Parameter Name	Description
BCC Recipient(s)	specifies as blind-copied recipients one or more users defined in SAS metadata. Separate multiple user names with commas.
Group Recipient(s)	specifies as primary recipients one or more groups defined in SAS metadata. Separate multiple group names with commas.
Group CC Recipient(s)	specifies as copied recipients one or more groups defined in SAS metadata. Separate multiple group names with commas.
Group BCC Recipient(s)	specifies as blind-copied recipients one or more groups defined in SAS metadata. Separate multiple group names with commas.
Description	specifies the description (display name) for the event.
Template	specifies the SAS notification template to use. If this parameter is not specified, then the policy defaults to the SAS_Email_Message template.
Directive	specifies the target page to which you are directed upon notification. The value must be a registered SAS directive name. If you are logged on to the SAS platform, then selecting the ellipsis button (...) presents a list of all registered directives.
Notification Variables	specifies one or more label paths for data objects that represent name-value pairs that are used as merge variables for the template. The data object name corresponds to the notification variable name while the data object value is used as the variable value. Separate the label path values with commas.
HTTP Parameters	specifies one or more label paths for data object that represent name-value pairs that are used as HTTP parameters. The data object name corresponds to the HTTP parameter name while the data object value is used as the parameter value. Separate the label path values with commas.

Parameter Name	Description
Action on Expiry	specifies the action to perform if the process is not completed by the date specified by the **Expiration Date Data Object** parameter. The available expiry options are None, Remove, Reroute, Resend, and Start Workflow. For more information, see "Expiry Options for Notification Policies" on page 44.
Expiration Date Data Object	specifies the data object that defines the expiration date of the policy.
Expiry Recipients(s)	specifies one or more users defined in SAS metadata to whom notifications are sent when the policy reaches its expiration date. Separate multiple user names with commas. *Note:* This parameter is available only when the **Action on Expiry** parameter is set to **`Reroute`**.
Expiry Group Recipients(s)	specifies one or more groups defined in SAS metadata to which notifications are sent when the policy reaches its expiration date. Separate multiple group names with commas. *Note:* This parameter is available only when the **Action on Expiry** parameter is set to **`Reroute`**.

All policy parameters except for **Expiration Date Data Object** are text fields representing a specific value or a dynamic value using data object substitution. For more information, see "Data Object Substitution" on page 52. The **Expiration Date Data Object** parameter is a label path to a data object and does not support object substitution.

When the Alert Notification Service receives the event, an end-user notification message is generated using the Template Service to specify the message format. Notification variables are name-value pairs that are used as merge variables and are applied to the template. If the **Directive** parameter is set, then the Directive Service is used to generate a URL based on the directive name and any value specified for the **HTTP Parameters** values. The HTTP parameters are optional name-value pairs specified on the event.

Note: Data objects that are used as notification (merge) variables must follow the naming conventions defined for the SAS Template Service.

Expiry Options for Notification Policies

For policies that have expiration date properties, one of the following expiry options can be specified to determine the action that is taken when the expiration date is reached.

None

no expiration for this notification.

Note: The **Expiration Date Data Object** parameter is not required for this option because the notification does not expire.

Remove

removes the policy once the date is passed.

Reroute
: sends the notification to alternative recipients or group recipients as specified.

Resend
: re-sends the notification to the original recipients defined.

Workflow
: starts a new process with the name specified.

Send Workflow Notification

This policy is used to send a directed workgroup notification via the SAS Alert Notification Service. Workflow notifications can be sent to users via e-mail, SMS message or displayed in a portlet.

The following parameters can be defined for this policy:

Parameter Name	**Description**
Recipients(s)	specifies one or more users defined in SAS metadata. Separate multiple user names with commas.
Group Recipient(s)	specifies one or more groups defined in SAS Metadata. Separate multiple group names with commas.
Description	specifies the description (display name) for the event.
Template	specifies the notification template to use. If this parameter is not specified, then the policy defaults to the SAS_Email_Message template.
Directive	specifies the target page to which you are directed upon notification. The value must be a registered SAS directive name. If you are logged on to the SAS platform, then selecting the ellipsis button (...) presents a list of all registered directives.
Notification Variables	specifies one or more label paths for data objects that represent name-value pairs that are used as merge variables for the template. The data object name corresponds to the notification variable name while the data object value is used as the variable value. Separate the label path values with commas.

Parameter Name	Description
HTTP Parameters	specifies one or more label paths for data objects that represent name-value pairs that are used as HTTP parameters. The data object name corresponds to the HTTP parameter name while the data object value is used as the parameter value. Separate the label path values with commas.
Action on Expiry	specifies the option to perform if the process is not completed by the date specified by the **Expiration Date Data Object** parameter. The available expiry options are None, Remove, Reroute, Resend, and Start Workflow. For more information, see "Expiry Options for Notification Policies" on page 44.
Expiration Date Data Object	specifies the data object that defines the expiration date of the policy.
Expiry Recipients(s)	specifies one or more users defined in SAS metadata to whom notifications are sent when the policy reaches its expiration date. Separate multiple user names with commas. *Note:* This parameter is available only when the **Action on Expiry** parameter is set to `Reroute`.
Expiry Group Recipients(s)	specifies one or more groups defined in SAS metadata to which notifications are sent when the policy reaches its expiration date. Separate multiple group names with commas. *Note:* This parameter is available only when the **Action on Expiry** parameter is set to `Reroute`.

All policy parameters except for **Expiration Date Data Object** are text fields representing a specific value or a dynamic value using data object substitution. For more information, see "Data Object Substitution" on page 52. The **Expiration Date Data Object** parameter is a label path to a data object and does not support object substitution.

When the Alert Notification Service receives the event, end-user notification message is generated using the SAS Template Service to specify the message format. Notification variables are name-value pairs that are used as merge variables and are applied to the template. If the **Directive** parameter is set, then the Directive Service is used to generate a URL based on the directive name and any value specified in the **HTTP Parameters** parameter. The HTTP parameters are optional name-value pairs specified on the event.

Note: Data objects that are used as notification (merge) variables must follow the naming conventions as defined by the SAS Template Service.

Note: This notification is directed and cannot be opted out based on SAS platform user preferences.

Set Process Participant

This policy is used to set a participant, which specifies access control information for the activity.

The following parameters can be defined for this policy:

Parameter Name	Description
Process	specifies the label path for the target process or activity.
Role	specifies the workflow role for the participant.
Type	specifies the access control type: User, Group, or Role
Name Data Object	specifies the data object that defines the name of the participant.

Note: The policy must be defined in the same process where the data object is defined. Otherwise, the policy does not execute.

Set Multiple Process Participants

The Set Multiple Process Participants policy is used to add multiple participants which specify access control information for the activity.

The following parameters can be defined for this policy:

Parameter Name	Description
Process	specifies the label path for the target process or activity.
Role	specifies the workflow role for the participant.
Names Data Object	specifies the data object that specifies the names of the participants.
Delimiter	specifies the delimiter used if multiple values are specified for the **Names Data Object** parameter

Note: The policy must be defined in the same process where the data object is defined. Otherwise, the policy does not execute.

Set Overdue Status

The Set Overdue Status policy is used to add the Overdue status to a process.

The following parameters can be defined for this policy:

Parameter Name	Description
Process	specifies the label path for the target process or activity.

Start Process

The Start Process policy is used to start a subprocess or activity.

The following parameters can be defined for this policy:

Parameter Name	Description
Process	specifies the label path for the target process or activity.

Start Process with Label

The Start Process with Label policy is used to start a new separate process.

The following parameters can be defined for this policy:

Parameter Name	Description
New Process Label Data Object	specifies the label path to a data object that contains the name of the process definition to clone and start.
New Process ID Data Object	specifies the label path to a data object that specifies the value of the instance ID of the newly created process instance.
Current Process ID Data Object	specifies a data object that specifies the instance ID of the process executing this policy. The data object must exist in the newly created instance.

When the policy executes, the process is started. The instance ID of the newly started process instance is stored in the New Process ID data object. In addition, the policy execution looks for the Current Process ID data object in the newly started instance and stores the instance ID of the activity associated with the policy.

Stop Process

The Stop Process policy is used to stop a subprocess or activity.

The following parameters can be defined for this policy:

Parameter Name	Description
Process	specifies the label path for the target process or activity.

Stop Process with ID

The Stop Process with ID policy is used to stop a process as defined by the process identifier that is stored in a data object.

The following parameters can be defined for this policy:

Parameter Name	Description
ID of Process to Stop	specifies the label path to a data object with the ID value of the process to stop.

Adding Policies

Because the policy definition varies significantly depending on the action supported, the Edit Policy dialog box configuration adapts according to the type of policy.

To define a new policy, follow these steps:

1. 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.

   Alternatively, right-click an activity in the diagram editor and then select the **New Policy** menu option.

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

   Event
   corresponds to the workflow event that triggers the policy.

   Action
   corresponds to the policy type. The selection controls what fields appear in the Properties section.

3. Some policy types are parameterized policies, which require parameters to configure the policy instance. If applicable, enter the parameter values.

   Note: Some of these parameters provide a text editing tool. Others use the selection button to access global data objects

4. Select **OK** to save the policy definition.

Editing Policies

To edit an existing policy, follow these steps:

1. In the process tree, open the **Policies** folder.
2. Right-click on the desired policy node and then select **Edit** from the resulting pop-up menu.
3. Changed the desired values in the Edit Policy dialog box.
4. Select **OK** to save the changes to the policy definition.

Deleting Policies

To completely remove a policy from the workflow, follow these steps:

1. Right-click on the target policy in the process tree and then select **Delete** from the resulting pop-up menu

Alternatively, select the policy directly in the process tree and press the Delete key or Ctrl-X.

2. Select **Yes** in the confirmation dialog box to permanently remove the selected policy.

Assigning Policies

To assign a policy to an activity, drag the desired policy from the process tree to the relevant activity in the diagram editor.

Alternatively, copy (to reuse the global definition) or drag (to demote it to a local definition) the policy element into the **Policies** folder for the target activity in the process tree.

Working with Data Objects

Overview

Data objects are similar to variables in a computer program and they hold business data required by the process. 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.

Note: Do not replicate application data structures in the data objects. Instead, define only the data required to evaluate decision points or for use in policies. This practice results in more efficient process definitions with less coupling, which minimizes impact from potential application data model changes.

Note: The number of data types supported in SAS 9.3 has been reduced and strong typing is now enforced. Workflow templates from SAS 9.2 are migrated when they are opened in SAS Workflow Studio and you are prompted to verify the migration and replace data objects as needed to support the new explicit types. Workflow services will evaluate older object definitions to determine the data type at run time, but this function will be deprecated in a future release, so you should migrate all data types accordingly.

Adding Data Objects

To define a new data object, follow these steps:

1. In the process tree, right-click the top-level or local data object folder in the process tree and then select **Add Data Object** from the resulting pop-up menu.

   Alternatively, right-click an activity in the diagram editor and then select the **New Data Object** menu option.

2. In the New Data Object dialog box, specify a label for the object in the **Data Object Label** field.

3. Select one of the following values for `Type`:

   Date
   E-mail
   Item List

Long Text (unlimited length)

Short Text (4000 bytes or less and searchable)

Number

URL

XML Object

4. Specify the relevant values in the **Properties** fields based on the type selected.
5. Optionally, add a description for the data object in the **Description** field.
6. Optionally, add localized versions of the label and description.

   For more information, see “Text Localization” on page 52.
7. Optionally, add custom attributes.

   For more information, see “Custom Attributes” on page 52.
8. For **Scope**, check **Make visible in entire subtree** for global data objects.
9. Select **OK** to save the data object definition.

Editing Data Objects

To edit an existing data object, follow these steps:

1. In the process tree, open the **Data Objects** folder.
2. Right-click on the desired data object node and then select **Edit** from the resulting pop-up menu.
3. In the Edit Data Object dialog box, make the desired changes.
4. Select **OK** to save the data object definition.

Deleting Data Objects

To completely remove a data object from the workflow, follow these steps:

1. Right-click on the target data object in the process tree and then select **Delete** from the resulting pop-up menu.

   Alternatively, select the data object directly in the process tree and press the Delete key or Ctrl-X.
2. Select **Yes** in the confirmation dialog box to permanently remove the selected data object.

Assigning Data Objects

To assign a data object to an activity, drag the data object from the process tree to the relevant activity in the diagram editor.

Alternatively, copy (reuse the global definition) or drag (demote it to a local definition) the data element into the **Data Objects** folder for the target activity in the process tree.

Additional Process Features

Text Localization

SAS Workflow Studio supports localization of name and description text fields for processes, data objects, statuses, and activities. Localizable fields are designated by an ellipsis button (...). Clicking the button opens the Localized Text dialog box. In that dialog box, you can specify Locale and Localized Text value pairs to define localized versions of the text for the corresponding field.

Note: The value entered in the original text field is associated with the English locale.

Note: Localization of date and number formats is supported internally in SAS 9.3 and defaults to the locale used in SAS Workflow Studio when the workflow is defined. If the workflow template is imported into a run-time environment that supports a different locale, then the date and number formats remains aligned with the locale of the workflow definition.

Custom Attributes

Custom attributes are application-specific values that are not related to the workflow business data. You can define custom attributes for processes, data objects, statuses, and activities. The editing windows for these objects include an **Attributes** button that opens the Attributes dialog box. In the Attributes dialog box, you can add Key and Value pairs to define the custom attributes.

These custom attribute values are not available within workflow policies or gateway expressions, but can be accessed by applications using the workflow client APIs. If the value is required within the workflow instance, then it should be defined as a standard data object rather than as an attribute.

Note: The use of application-specific attributes is not recommended to minimize coupling thereby maximizing process agility.

Data Object Substitution

As noted, some policy parameters can be set dynamically using data objects known as data object substitution. When the policy executes, it uses the value of the relevant data object for the policy parameter. The variable syntax used to indicate this value substitution is **${*relative-label-path*}**. SAS workflow replaces all occurrences with the value contained in the data object specified by relative label path.

For example, given the following data object relative paths

```
../CMD_ID
../PROCESS_ID
../HOST
```

and the following associated values

```
1212
345
T67890
```

for the text data object representing a URL specified as

```
http://${../HOST}/SASapp/Workflow?command=${../CMD_ID}&processid=${../PROCESS_ID}
```

resolves to

```
http://T67890/SASapp/Workflow?command=1212&processid=345
```

Chapter 5

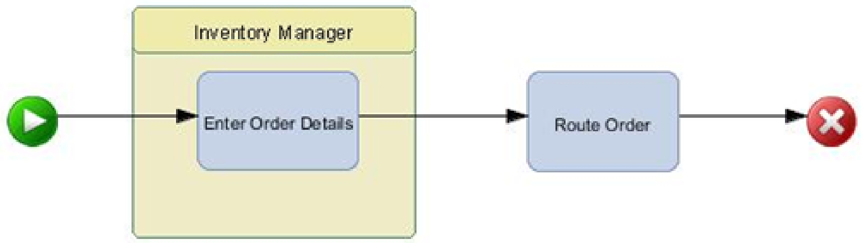
Advanced Topics

Workflow Patterns

Overview

Only the most basic processes can be represented as a single, sequential flow (Workflow Pattern: Sequence). More realistic processes generally contain varying combinations of activities based on business decisions derived from the business data. These variations can be initiated by specific outcomes from the previous task or by expression evaluation at decision points to control the method of selecting a path through the process at run time.

Figure 5.1 Sequential Process Example



Another common flow pattern is parallel processing, when two or more process paths execute in parallel in contrast to alternate path selection supported by statuses and decisions. SAS Workflow Studio supports both alternate and parallel flow patterns, as detailed in the following sections.

Defining Alternate Paths

Overview

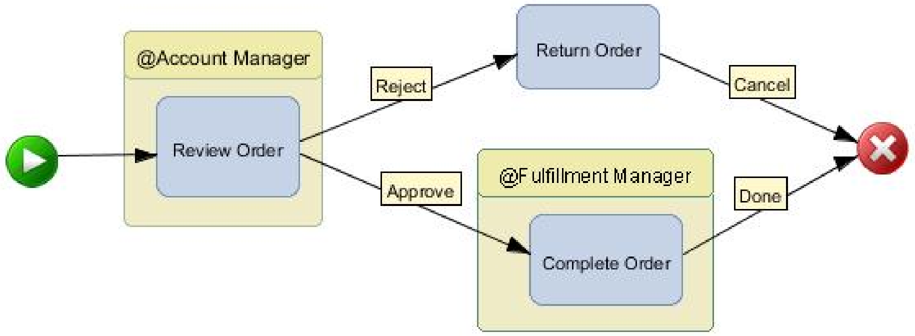
Many processes contain multiple paths where each path represents one potential case or process instance. The specific path taken is based on the relevant business logic evaluated at run time.

Path Selection Based on Status

If the end user has multiple choices when responding to a simple activity in their to-do list, then it can be modeled by defining a status for each choice to indicate unique outcomes.

For example, assume that you are defining a basic approval process with two possible outcomes: approved or denied. You can define two status values: Approve and Reject and then add the relevant status to the appropriate case in the workflow by assigning it to the connection for that path. At run time, the process continues to the next activity based on the status corresponding to the action.

Figure 5.2 Path Selection Based on Status



The status values are unique, so there is only a single choice possible for this example (Workflow Pattern: Exclusive Choice). SAS Workflow also supports multiple paths for the same status value (Workflow Pattern: Parallel), which behaves in a manner similar to using the merge/fork gateway, which might not have status assignments.

Refer to Appendix 4, “Basic Workflow Examples,” on page 95 for a detailed example.

Note: Most SAS products that leverage workflow use status values for every transition. Without status assignments, the activity execution is controlled purely by sequence flow—the order in which elements are connected—and gateways. SAS Workflow allows significant flexibility for defining process flow logic so process designers should carefully validate that the workflow behaves as expected.

Path Selection Based on Expression Evaluation

In contrast to status evaluation, decisions can be used to route the process flow by means of evaluated expressions based on real-time business data values. Decision gateways in SAS Workflow Studio contain Boolean expressions that drive actions based on the calculated value. Thus, decision nodes can be used to route process execution to one (Workflow Pattern: Exclusive Choice) or more (Workflow Pattern: Inclusive Choice) of several alternate outgoing paths, depending on the condition.

To add and configure decisions, follow these steps:

1. Select the Add Decision Gateway icon (◆) on the toolbar.
2. Click on an empty space in the drawing area to add a decision gateway to the diagram.

To define the properties of a decision node, follow these steps:

1. Right-click on the figure in the drawing area and then select **Edit** from the resulting pop-up menu.

   Alternatively, double-click on the decision node.

2. In the Edit Decision Gateway dialog box, enter the desired values.
3. Enter a name for the decision in the **Label** field. The default name of the first decision is **`Decision0`**. The second decision is **`Decision1`**, and so on.
4. To define Boolean expressions for a decision node, follow these steps:

   a. Select **Add** to open the Add Boolean Expression dialog box.

   b. Enter the desired logical expression.

   c. Optionally, provide a name for the expression in the **Label** field.

      (The default value is the expression itself.)

   d. Select **OK** to save the current expression.

      The newly defined Boolean expression should be visible in the list box.

   Repeat these steps to associate additional Boolean expressions with the current decision figure.

5. Select **OK** to save the decision definition.

All expressions must evaluate to a Boolean value and must conform to the Java language syntax. In summary, each expression:

- must evaluate to true or false
- can use comparison operators (`<`, `=`, `==`, or `>`)

Note: The `=` and `==` operators are evaluated in an identical manner.

- can contain logical combinations (**`&&`**, `||`, **`!`**) of some set of comparisons
- can contain arithmetic expressions on each side of the comparison. The arithmetic expression can be any combination of the following operators: **`+`**, **`-`**, **`*`**, **`/`**, or **`%`**. Arithmetic expressions can also contain functions, integers, floating point numbers, and data objects.

Data objects are specified by their label, and might contain spaces.

Refer to Appendix 1, "Decision Expression Examples," on page 75 for detailed operator and function support and Boolean expression examples.

TIP To use data objects when writing Boolean expressions, press the F3 key to access a menu of valid data objects for the current process.

TIP Use the **Edit** and **Delete** buttons to modify the contents of the list of expressions.

To assign the expressions to the relevant process paths, follow these steps:

1. Ensure you have defined all the required Boolean expressions for the decision gateway.
2. Define the necessary process paths.
3. Connect the decision to the relevant activities for each path.
4. For each connection, right-click and associate the appropriate expression from the resulting **Change status** pop-up menu.

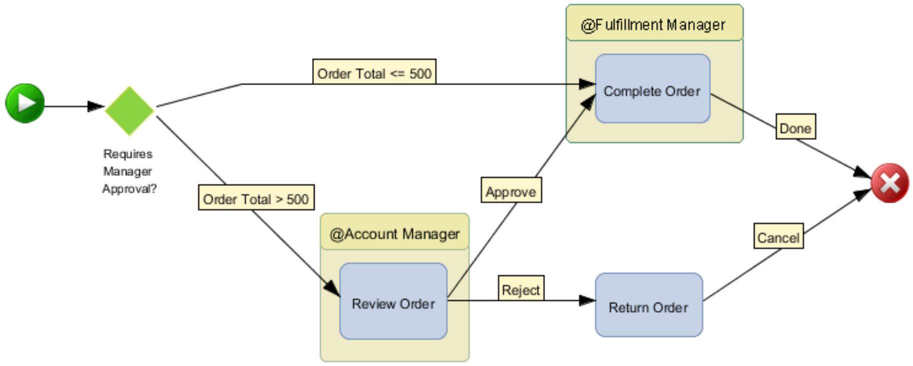
Each expression corresponds to a calculated value or outcome and is represented as a label in a similar manner to status values. In addition to these user-defined calculations, the special status of **`Otherwise`** can be assigned. Use this value to designate the logical path that should be traversed when all defined expressions evaluate to false. In summary, the **`Otherwise`** path represents the default execution path when none of the expression values are true.

Note: All paths with expressions that evaluate to true are executed.

For the following example, any order total that exceeds $500 requires manager approval before it is fulfilled. The decision gateway is based on an Order Total data object of type Number where the **`Yes`** path corresponds to a value exceeding the threshold and **`No`**, where the order total is below the threshold, is approved automatically.

Expression	**Resulting Action**
Order Total <= 500	Automated approval
Order Total > 500	Requires Manager approval

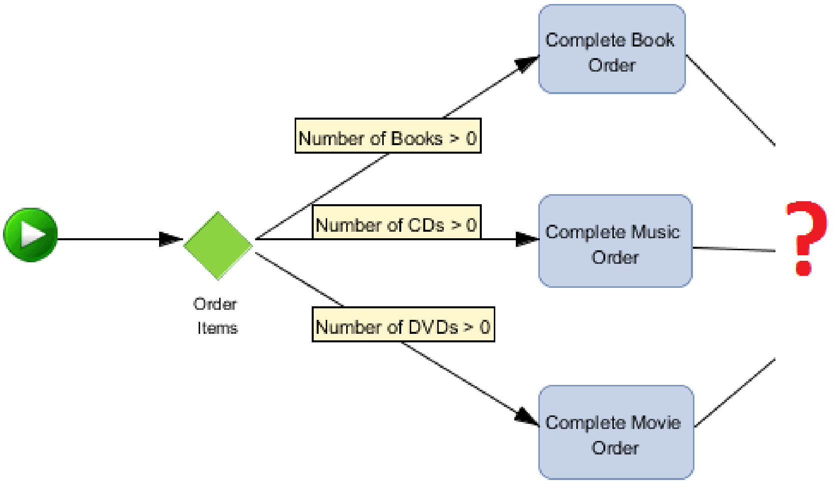
Figure 5.3 *Example of Path Selection Based on Expression (Exclusive Choice)*



Refer to the examples provided with SAS Workflow Studio for details of this example.

For a process flow where one or more paths from a decision can be followed at the same time, an inclusive choice decision might be used. For example, a mail order company receives orders with combinations of items – books, music (CDs), movies (DVDs) – with distinct fulfillment activities initiated for each item type. A typical order contains quantities of more than one product type.

Figure 5.4 *Example of Path Selection Based on Expression (Inclusive Choice)*

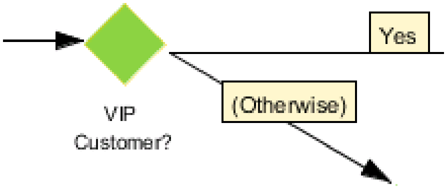


However, this divergence pattern (Workflow Pattern: Inclusive Choice) has two potential problems:

- Deadlock if none of the choice expressions are realized.
- Convergence requirements might be complex resulting in deadlock or surplus executions.

The first issue can be resolved by providing a default path. SAS Workflow supports this with the (Otherwise) choice, which is available with all decision gateways. Most importantly, the business logic must be captured accurately and should cover all possible outcomes.

Figure 5.5 Avoiding Deadlock with a Default Path



Defining Parallel Paths

Overview

Frequently, processes focus on sequences of activities where each task is activated upon completion of the prior task. At other times, a process might require multiple tasks or process paths to execute in parallel after a specific activity has completed. SAS Workflow Studio supports parallel paths via the merge/fork gateway element.

Adding Merge/Fork Gateways

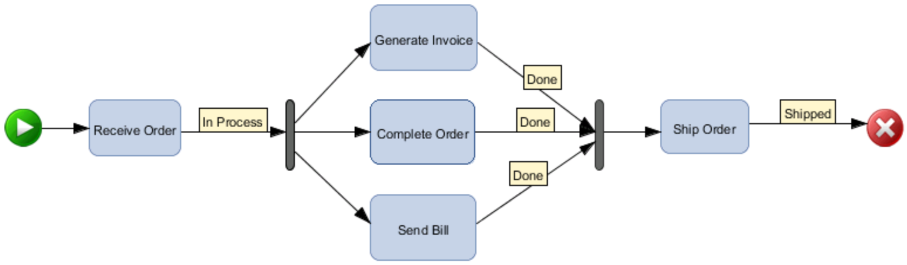
To add a merge/fork gateway, follow these steps:

1. Select the Merge/Fork Gateway icon ( ) on the toolbar.
2. Click on an empty space in the drawing area to add a merge/fork gateway to the diagram.
3. On the left side of the bar, draw the desired activity structure and connect to the merge/fork gateway.
4. On the right side, create the desired activity set and create a sequence flow line from the merge/fork gateway figure to each activity.

   If more than one activity exists prior to the bar, then all activities must complete before any activities on the other side are initiated.

Here is an example of a process flow that splits into three parallel activities (Generate Invoice, Complete Order, Send Bill), which start only after the Receive Order activity completes.

Figure 5.6 Example of Parallel Processes



Refer to Appendix 4, "Basic Workflow Examples," on page 95 for a more detailed parallel process example.

Note: Statuses cannot be assigned to connections leading from a merge/fork gateway. All subsequent paths are executed.

TIP To change the orientation of the merge/fork gateway from vertical to horizontal (and vice versa), right-click the bar and then select **Change orientation option** from the resulting pop-up menu.

The following table summarizes common divergence workflow patterns:

Workflow Pattern	Workflow Element	Description
Sequence	Sequence Flow (Connection)	Simple sequence where the subsequent activity is triggered once the current activity preceding it is competed.
Exclusive Choice	Decision Gateway	Alternate paths where only one is taken (logical XOR).
Inclusive Choice	Decision Gateway	Alternate paths where more than one might be taken, but not necessarily all paths (logical OR).
Parallel	Merge/Fork Gateway	Parallel structure where all paths are executed concurrently (logical AND).

Defining Convergent Paths

Overview

In general, divergent process flow logic eventually converges either by processing the inputs as they are received (no synchronization) or by coordinating and consolidating the inputs (synchronization) into a single execution path. These convergence pattern types are detailed in the following sections.

Merging Paths without Synchronization

As a rule, alternate paths should converge without synchronization to prevent deadlocks. This means that exclusive choice decisions (exactly one path is selected) should require only a single input upon convergence (Workflow Pattern: Exclusive Merge). This can be accomplished in two ways:

- explicitly specify an OR-type logic gateway (OR)
- configure the inputs to flow into a single task directly (no gateway).

SAS Workflow Studio currently supports logical XOR (exclusive) for the first option and logical OR (inclusive) for the second option. So, for a single execution of the convergent path, the logical OR gateway should be used. If multiple executions are desired (one for each path), then the paths should converge directly into an activity.

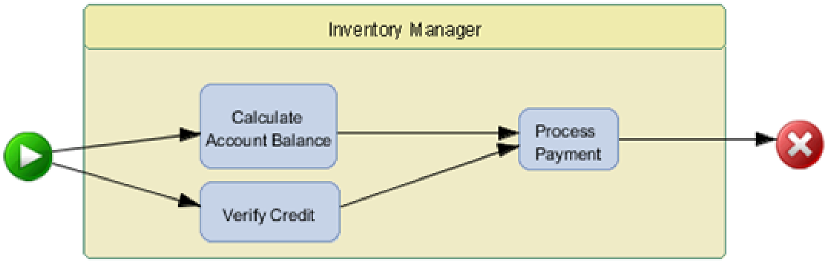
To add and configure a logic gateway, follow these steps:

1. Select the Logic Gateway icon (AND) on the toolbar.
2. Click on an empty space in the drawing area to add a logic gateway to the diagram.

   Note: The gateway is AND by default.
3. Configure the logic gateway as OR by right-clicking the gateway and then selecting the **OR** option from the **Change logic** resulting pop-up menu.
4. On the left side of the bar, create the desired activity set and connect each path to the logic gateway.
5. On the right side, draw the desired activity structure and create a sequence flow line from the logic gateway figure to the first activity in the converged path.

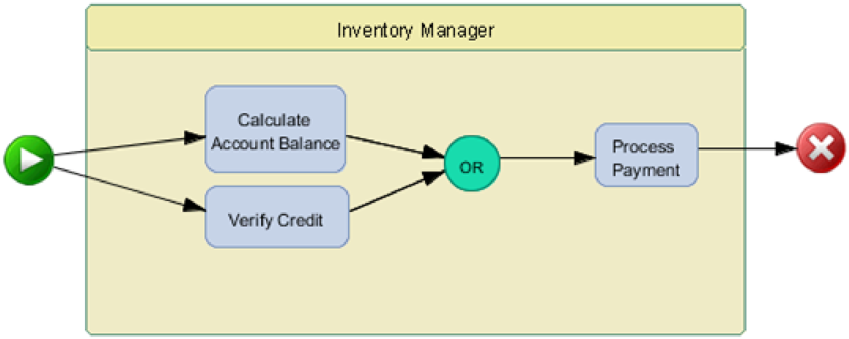
The following example results in only a single work item (task instance) for Process Payment once both Calculate Account Balance and Verify Credit have been completed. However, any actions associated with the Process Payment activity are executed twice.

Figure 5.7 *Example of Inclusive Merge (OR)*



The following example executes a single action triggered by the first of the preceding tasks that completes. If the other task completes before Process Payment is performed, then no additional action is taken.

Figure 5.8 *Example of Exclusive Merge (XOR)*

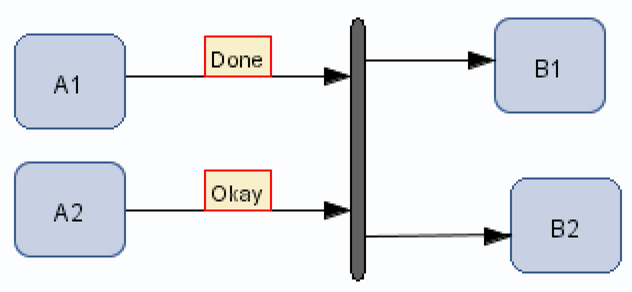


Merging Paths with Synchronization

Finally, parallel execution paths that require all tracks to complete should be joined and synchronized before initiating the convergent path. This can be accomplished by either using a merge/fork gateway or an AND-type logic gateway.

The parallel process in Figure 5.6 on page 60 illustrates merge/fork gateway usage where the Ship Order task is not executed until all three preceding tasks (Generate Invoice, Complete Order, and Send Bill) complete. If more than one activity exists after the gateway, then they are started together and run independently. The following is an example of a process flow where a set of activities (B1, B2) starts only after another set of activities (A1, A2) finishes:

Figure 5.9 *Example of a Complex Parallel Process*



Another variation on the inclusive merge convergence pattern, which controls partial synchronization, is sometimes referred to as the Discriminator model. This pattern supports multiple inputs that trigger multiple executions, but not necessarily one-to-one (selective execution). In other words, there can be three inputs leading to the converged path, but only two trigger actions while the remaining input is ignored, if present. This pattern is not supported by SAS Workflow Studio as an explicit gateway.

In summary, the common convergence workflow patterns are as follows:

Workflow Pattern	Workflow Element	Description
Exclusive Merge	OR Logic Gateway	Merging of alternate paths where only the first input is used to trigger the converged path Single execution (logical XOR)
Inclusive Merge	No gateway	Merging of alternate paths where more than one input can be used to trigger the converged path Multiple executions with no synchronization (logical OR)
Join	Merge/Fork Gateway AND Logic Gateway	Parallel structure where all inputs are required before proceeding on converged path complete synchronization (logical AND)

References

Nick Russell, Arthur H.M. ter Hofstede, Wil M.P. van der Aalst and Nataliya Mulyar "Workflow Control-Flow Patterns: A Revised View." 2006. Available http://www.workflowpatterns.com/documentation/documents/BPM-06-22.pdf.

W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski and A.P. Barros "Workflow Patterns." 2003. Available http://www.workflowpatterns.com/documentation/documents/wfs-pat-2002.pdf.

"Workflow Patterns Initiative." 1999. Available http://www.workflowpatterns.com.

Using Timers

Overview

Timers can be used in a workflow to trigger process steps at particular times or after specified intervals. Common timer-triggered actions include:

- start a process
- stop a process
- automatically traverse a transition
- deliver a notification

Adding Timers

There are two ways to define timers:

- use the timer tool (🕒) on the toolbar, which generates an associated Timer Tool Policy
- explicitly define a policy that is triggered by a Timer Expired event

Either way, a timer element is incorporated into the process with the specified timer settings.

Timer Considerations

Timers can be configured to reset and fire any number of times. Here is a list of important considerations for determining the approach that is best for your process:

- timers are initialized when the innermost process or activity containing the timer is started
- all outgoing connections from a timer element are traversed each time it fires.

  The timer execution varies depending on its orientation in the process diagram. A timer can assume the following configurations:

  - on the border of an activity—stops the activity when triggered

- inside the border of an activity—executes an associated action while the activity remains active

  Note: The timer starts when the parent activity begins.

- outside an activity in the main process—executes an associated action

Note: All transitions leading from a stand-alone timer share the same timer interval. That is, different timer intervals cannot be specified for each transition from the timer. If this behavior is desired, then a separate timer is required for each interval.

Advanced Timer Parameters

The advanced timer properties dialog box can be viewed by selecting the **Advanced** button to the right of the **Schedule Expression** field. The advanced timer properties dialog box defines the following information:

- Schedule Expression
- Recurrence
- Timer End

Schedule Expression

The **Schedule Expression** field is used to specify the time at which the timer triggers. The value should be a relative expression or cron-like syntax based on when the parent element starts. The expression supports second, minute, hour, day, and week time intervals. The supported syntax is either the special reserved word `Now` or a plus sign followed by one or more digits followed by a unit of measure. A combination of one or more sequences of plus sign followed by a unit can be specified.

The following units are supported:

Unit	Description
s	seconds
m	minutes
h or H	hours
d	days
w	weeks

The following examples illustrate valid expressions:

Expression	Interpretation
Now	the timer expires immediately
+1s	the timer expires in one second
+1m	the timer expires in one minute

Expression	Interpretation
+1h	the timer expires in one hour
+1d	the timer expires in one day
+1w	the timer expires in one week
+1h+1m+1s	the timer expires in one hour + one minute + one second

Refer to Appendix 3, "Timer Examples," on page 89 for additional examples of timer expressions, including cron syntax.

Recurrence

The **Specify recurrence** section includes the following fields:

Number of repetitions
specifies how many times the timer should fire.

Interval of repetitions
is a relative expression that identifies the time interval between repeated executions. By default, timers fire only once.

Note: This field is ignored when a cron expression is specified for the **Schedule Expression**. The syntax for cron expressions implicitly supports repeated firings of the timer object.

Timer End

The **Specify timer end** section includes the following fields:

End timer
specifies when the timer should stop. The end value is a relative or cron expression that identifies when the timer should stop. The syntax supported is similar to that for the **Schedule Expression** field.

Interval of repetitions
is a relative expression that identifies the time interval between repeated executions. By default, timers fire only once.

Deploying and Maintaining Processes

Overview

Process diagrams defined using SAS Workflow Studio can be stored as flat files in XML format. Alternatively, SAS Workflow Studio supports persistence and versioning of the workflow templates using SAS Content Services. Users can also activate the specified template stored in the workflow repository. Activation results in uploading the specified version as a formal workflow definition into the SAS Workflow Engine for instantiation by end-user client applications. Refer to "Workflow Lifecycle" on page 7 for more details.

Note: Process changes made in SAS Workflow Studio do not take effect until the new version is uploaded and activated. Once activated on the workflow server, any new processes that are started use the newly activated version. Any currently running processes continue using their respective version of the workflow definition.

Saving a Workflow Template

To a save the current workflow template to the local file system, follow these steps:

1. From the File menu, select **Save** or **Save As**.

   Alternatively, select the Save icon (💾) on the toolbar.
2. Navigate to the desired directory in a dialog box.
3. Enter the desired filename for the process definition, or select an existing file to replace.
4. Select **Save** to close the dialog box and return to the drawing panel.

Restoring a Workflow Template

To restore a workflow template from the local file system, follow these steps:

1. Select **File** ⇨ **Open** from the menu bar.

   Alternatively, select the Open icon (📂) on the toolbar.
2. Navigate to the desired directory in the dialog box.
3. Select the desired file.
4. Select **OK** to open the template in the drawing panel.

Logging On to the Server

With SAS Workflow Studio, you are limited to managing locally stored workflow templates on your system until you have logged on to the SAS platform. Once connected, you can access additional process templates stored in the SAS metadata repository.

To log on, follow these steps:

1. Select **Server** ⇨ **Logon** from the menu bar, or press F9.
2. In the Log On dialog box, enter the host name.

   The address used is prepopulated based on the SAS platform environment properties file.
3. Enter your user name and password.
4. Select **Log On**.

Note: The available host parameters are configured in the environments.xml file.

```
<environment name="localhost" default="false">
   <desc>SAS Environment: localhost</desc>
   <service-registry>
```

```
      http://localhost:8080/SASWIPClientAccess/remote/ServiceRegistry
   </service-registry>
   <service-registry interface-type="soap">
      http://localhost:8080/SASWIPSoapServices/services/ServiceRegistry
   </service-registry>
</environment>
```

For details about this configuration, refer to the *SAS Intelligence Platform: Web Application Administration Guide*.

Versioning Process Definitions

Any user configured using the SAS platform metadata can upload, download, or activate process definitions using SAS Workflow Studio.

Uploading a Process

Uploading a process file stores it in the SAS Content Server.

To upload a process, follow these steps:

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

   Alternatively, you select the Save a Process Definition to the Workflow Repository icon (⇧) in the toolbar or press F5.

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

2. Enter relevant comments in the Save to Workflow Repository dialog box.

   Note: Comments are optional, but highly recommended because the version history displays them. Summarize the high-level differences for each version so it is easier to identify the changes at a glance.

3. Select the **Activate** option if you would also like to activate the current version in workflow.

4. If the definition has not been previously saved locally, then the Save dialog box appears. See “Saving a Workflow Template” on page 67 for details.

You can also use the Workflow Loader utility to load existing process templates (a single file or a directory of templates) directly into the content repository from the command line. The utility also supports activation of the process (that is, uploading the definition to the run-time platform).

Use the following syntax to invoke the Workflow Loader utility:

java

-Dmulticast.address=*nnn.nnn.nnn.nnn*

-Dmulticast.port=*nnnn*

com.sas.workflow.util.checkin.WorkflowLoader

user-ID

password

path

-outputDir *directory-name*

<-replace>

<-activate>

where

nnn.nnn.nnn.nnn
is the multicast address for the environment where the middle tier is deployed.

nnnn
is the multicast port number of the middle tier.

user-ID
is the ID of a user who is authorized to load content. In SAS 9.3 any user defined in the metadata server can load content.

password
is the password for the specified user.

path
is the fully qualified location of the processes to load. If a single filename is specified, then the content from that single process template file is loaded. If the path resolves to a directory, then the contents of all the files in that directory are loaded.

-outputDir
specifies the directory where modified XML is written if any data types are fixed.

-replace
replaces the definition if already exists in the engine. In the SAS Content Server, this option updates the version.

-activate
activates or loads the template into engine after template is loaded in to the repository.

The Workflow Loader automatically fixes any invalid data types by mapping them to appropriate supported types for SAS 9.3 (for example, Short Text). If any data type migration occurs in the definition file, then a new file with the corrected data type is created in the directory specified with -outputDir option and the migrated file is automatically loaded into content repository.

Downloading a Process

Downloading a process retrieves the workflow definition from the Workflow Repository, so that it can be viewed or revised in SAS Workflow Studio.

To download a process, follow these steps:

1. Select **Server** ⇨ **Open from Repository** from the menu bar.

   Alternatively, you can select the Open a Process Definition from the Workflow Repository icon (⇩) in the toolbar or press F7.

   Note: These options are disabled until you log on to the server.

   The Open from Workflow Repository dialog box appears and displays a list of processes available for download along with the relevant version information (repository and activated) and check-out status.

2. Select the desired process.

3. Choose one of the following download options:

   Get copy
   retrieves and views as a local copy

   Check out
   checks the selected template out of the repository. This selection is required if you want to revise and save any changes to the workflow template.

Note: The checked out version is locked for editing by other users until the template is checked back in or the check out operation is undone.

4. Select **OK** to load the specified process definition in the diagram editor.

Overriding a Process Template

If you forget to check out a workflow template before making changes to a local version open in SAS Workflow Studio, then you can overwrite the repository version.

To override the version of a template in the repository, follow these steps:

1. Download and check out the workflow template.

   The current repository version opens in a new tab.

2. Navigate to the tab for the revised workflow diagram.
3. Upload the revised template to the repository.

Process Management

SAS Workflow Studio includes the following process management functions from the Manage Processes dialog box:

Get a copy of a workflow template
: retrieves a local copy for viewing

Check out a workflow template
: retrieves a local copy for revision in SAS Workflow Studio and locks the repository version from access by other users

Check in a workflow template
: checks in the current local template as the next incremental version in the repository and unlocks it

Undo a check out for a workflow template
: unlocks the repository version for access by other users

Activate a workflow template
: uploads the workflow template as a formal definition in the SAS Platform for instantiation by end-user client applications

  Note: Only one version of a workflow definition is active at any time. Any process instances based on previous versions of the definition are unchanged. However, any new instances are generated using the current activated version.

Show version history for a workflow template
: displays a version history for the specified template version

Delete a workflow template
: removes the selected version from the workflow repository

  Note: The delete function does not remove the activated workflow definition from the SAS platform.

To access these functions, follow these steps:

1. Select **Server** ⇨ **Manage Processes** from the menu bar.

   Alternatively, select the Manage Process Definitions icon ( ) or press F3.

The Manage Processes dialog box appears. The dialog box displays a list of processes in the workflow repository along with the relevant version information (repository and activated), modification status, check-out status, comments and tags.

If a green check mark is visible, then the designated template has been checked out and locked by the specified user.

2. Select a workflow template and right-click on the selected item.
3. Select the desired management function from the resulting pop-up menu.

   If the **Show Versions** option is selected, then the version history is displayed in a separate dialog box. You can get a copy, check out, or activate specific unlocked versions from this dialog box.

Workflow Template Categorization

SAS Workflow supports isolation of workflow definitions using tags. Tags act as labels for workflow templates to provide categorization or grouping of related definitions on a user-defined basis (for example, by application context, organization, and so on). These alphanumeric labels support filtering or searching capabilities for more efficient management of workflow templates. Tag definitions and assignments are specified at design time and cannot be modified at run time. Tags provide basic categorization capabilities, but are not used for access control. Therefore, tags do not prevent sharing of workflow definitions across applications.

To add a tag to a workflow template, follow these steps:

1. Open the workflow in SAS Workflow Studio.
2. Log on to the server. For more information, see "Logging On to the Server" on page 67.
3. Select **File** ⇨ **Properties** from the menu bar.
4. Click the **Add** button in the **Tags** section of the Properties dialog box.
5. Enter the tag value.
6. Select **OK** to save the tag definition.

The new tag value now appears in the Properties dialog box. You can assign it to the current workflow by selecting the associated check box.

Note: Tags are defined at the server level and can be assigned to any workflow template for that system. If the workflow is moved to another system, the user is prompted to create the relevant tags on the new system before uploading and activating the workflow template.

SAS Alert Notification Templates

Editing Alert Notification Templates

The alert notification templates are deployed to the WebDAV repository on the SAS Content Server.

Note: You should keep a backup copy of the alert notification templates. If you need to refer to the default alert notification templates or to reload them to the WebDAV repository, then you can access them in the appropriate language directory in the `SAS_HOME\SASWebInfrastructurePlatform\9.3\Config\Deployment\Content\Templates\notification` directory.

To edit the alert notification templates using the SAS DAVTree utility, follow these steps:

1. Launch the SAS DAVTree utility from the `SAS_CONFIG\Web\Utilities` directory.

   Note: You must have the JAVA_HOME environment variable defined on your system to run the SAS DAVTree utility.

2. Select **File** ⇨ **Open** to access the DAV Location dialog box.
3. Enter the host name for the SAS Content Server in the DAV Location dialog box (for example, `http://*server*:*port*/SASContentServer/repository/default`)
4. select **OK** to save the update alert notification template.
5. The User Credentials dialog box appears. To connect to the WebDAV repository, enter an authorized administrative user name and password.

   To locate the alert notification templates in the WebDAV repository directory, expand the following folders:

   sasdav
   - ▸ **Templates**
     - ▸ **notification**
       - ▸ **en**

   Note: The English-version of the alert notification templates is located in the `en` directory. Select an appropriate directory for your localized version of the alert notification templates.

6. Select an alert notification file that you want to edit.
7. Drag the file from the DAVTree to your desktop.
8. Edit the file and save your changes.
9. Drag the file back to the appropriate folder in the DAVTree Utility.

For more information about using the DAVTree utility, see “Using the DAVTree Utility to Manage WebDAV Content” in the *SAS Intelligence Platform: Web Application Administration Guide* at `http://support.sas.com/documentation/onlinedoc/intellplatform`.

Adding Alert Notification Templates to the WebDAV Repository

To add an alert notification template in the DAVTree, follow these steps:

1. Select the appropriate directory where you want to add the new alert notification template (for example, **en**).
2. Select **Edit** ⇨ **Add** from the menu bar.

   A confirmation dialog box appears.

3. Select **Yes**.

A new window appears.

4. Select **Resource** and then enter the name for the new alert notification template. The name should include the file extension .st for string template support.
5. Select **OK**.

To add content to the new template, right-click the newly added template in the directory tree and then select **Edit** from the resulting pop-up menu.

Adding Fields to Alert Notification Templates

When you configure the alert notification templates to add a new field, you must also add a corresponding data object and notification variable to the parent workflow template. Note that these fields are case-sensitive.

Note: Fields that are represented as an array of objects are shown in the alert notification as a comma-separated string.

To add fields to an alert notification template, follow these steps:

1. Open the alert notification template for editing.
2. Determine the field or fields that you would like to add to the alert notification template.

For example, to add the issue priority (ISSUEPRI) field to the SAS_Email_Message notification template, do the following:

1. Open the SAS_Email_Message.st notification template for editing.
2. Update the contents to include the issue priority ($ISSUEPRI$) field. Be sure to add the relevant field in all three template areas: html, text, and SMS.

   You can add other business object fields to the templates. You must prefix and suffix these fields with the $ character in the notification templates.
3. In SAS Workflow Studio, open the corresponding workflow template, and add a new data object to reflect the field that you added to the notification templates.

   Note: The data objects are case-sensitive.
4. Add the new data object to the **Notification Variables** list for the Send Workflow Notification policy.

Configuring the Subject for an Alert Notification

To configure an alert notification subject, follow these steps:

1. Start SAS Management Console and connect as a SAS administrator.
2. In the Configuration Manager plug-in, right-click **SAS Application Infrastructure** and then select **Properties** from the resulting pop-up menu.

   The SAS Application Infrastructure Properties dialog box appears.
3. Select the **Settings** tab and then select **Notifications** from the list on the left of the dialog box.
4. Select `Custom` in the **Alert prefix type**field.
5. Clear the value in the **Alert prefix** field and then enter the desired value.

6. Select **OK** to save the notification settings.

Refer to Appendix 2, “Policy Usage Examples,” on page 79 for a detailed example that leverages the SAS Template Service.

Appendix 1
Decision Expression Examples

Operator and Function Support

Expressions in decision gateways can use the following operators:

Table A1.1 *Comparison Operators*

Operator	Operation
`=`	equal
`==`	equal
`<`	less than
`<=`	less than or equal
`>`	greater than
`>=`	greater than or equal
`!=`	not equal

Table A1.2 *Arithmetic Operators*

Operator	Operation
`+`	addition
`-`	subtraction
`*`	multiplication
`/`	division
`%`	modulo

Table A1.3 *Conditional Operators*

Operator	Operation
&&, AND, and	conditional AND
||, OR, or	conditional OR

Table A1.4 *Bitwise and Bit Shift Operators*

Operator	Operation
<<	left shift
>>	right signed shift
>>>	right unsigned shift
&	bitwise AND
|	bitwise inclusive OR
^	bitwise exclusive OR
~	bitwise NOT

Table A1.5 *Unary Operators*

Operator	Operation
+	unary plus
-	unary minus
++	increment value by 1
--	decrement value by 1
!, NOT, not	logical NOT

Decision gateway expressions also support some general functions that can be used in conjunction with data objects. The functions only support data objects. They do not support literal constants. The function is applied to the data object value, before the evaluation or comparison occurs.

Table A1.6 Supported Expression Functions

Function	Description
upcase	Converts the data object value to uppercase.
timepart	Uses only the time part of the data object value when evaluating. *Note:* This function is supported only by Date data objects.
datepart	Uses only the date part of the data object value when evaluating. *Note:* This function is supported only by Date data objects.

Example Data Objects

For the following examples, the process definition includes the following data objects:

Label	Data Type
Name	String data object
ProductId	String data object
Discount	Numeric data object
Cost	Numeric data object

Example 1: Comparing the Value of a Data Object to a String Constant

```
ProductId == "A123B"
```

Example 2: Comparing the Values of Multiple Data Objects to String Constants

```
(ProductId == "A123B") && (Name == "Smith")
```

Example 3: Comparing the Value of a Data Object to a Numeric Constant

```
Discount >= 5
```

Example 4: Comparing the Values of Two Different Data Objects

```
Cost == (Cost - Discount)
```

Example 5: Expression Using the Upcase Function

```
upcase(ProductID) == "A1235CC"
```

Example 6: Expression Using Data Object Substitution

This example uses the **${}** syntax so that the complete path to the data object can be specified. For more information, see "Data Object Substitution" on page 52. This is useful if a local data object is defined that has the same name as a global data object defined on the root process and you want the decision gateway to use the global object. In this example, the decision gateway uses the data object Discount that is defined on the Main process.

```
${/Main/Discount} > 1000
```

Note: Data determination logic is supported to ensure backwards compatibility with previous versions of workflow. All data objects should be defined explicitly using the appropriate data type (for example, Date , Numeric , Short Text , and so on). In subsequent releases, processing will honor only the specified data type and all support for real-time data type determination will be removed.

Appendix 2
Policy Usage Examples

HTTP Request Policy Example

In this example, the GRC_Workflow directive is used to determine the URL at run time because the URL field parameter is not specified. Also, the data object GRC_HANDLE referenced in the Parameters field is defined on the root process, which is two levels above the policy element. When the policy executes, it determines the value of the GRC_HANDLE data object, and substitutes that value into the HTTP parameters value. Finally, the return code value is stored in the ReturnCode data object, which is associated with the GRC activity or subprocess (that is, relative path).

The HTTP Request policy in this example uses the following parameters:

Table A2.1 *HTTP Request Policy Parameters*

Policy Parameter	Value
Event	Process Started
Action	HTTP Request
Description	Execute an HTTP Request
Directive	GRC_Workflow
URL	
Parameters	`Command=setFields&tpHandle=${../../GRC_HANDLE}`
Method	POST

Policy Parameter	Value
Status Code Data Object	GRC/ReturnCode
Policy Label	Process Started->Http Request

Invoke SAS Code Policy Example

This example demonstrates the use of data objects passed as macro variables used in the following simple SAS program:

```
data shoeSales;
   set sashelp.shoes end=final;
   retain total 0;
   total = total + &var;
   if final then call symputx('result', total);
run;
```

This example defines the following variables:

shoeSales
   is the name of the column in the shoes data set.

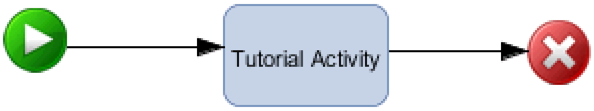
var
   is a macro variable whose value is passed to the SAS code from a source workflow data object.

result
   is the macro variable that holds the return value used to populate the target workflow data object.

If this code is hosted by a Web application server and the file is named invokeSASCodeTutorial.sas, then the file is accessible at **http://*host-name*/invokeSASCodeTutorial.sas**

Note: Versions 7.0 and higher of Microsoft Internet Information Services do not handle unrecognized file types, so a MIME type must be added for the .sas file extension.

First, define the relevant workflow and data objects. In the following example, the data objects are associated with the activity and are visible in the entire subtree:



Data Object Label	Type	Value
Total	Short Text	0
Total Category	Short Text	**sales**

Next, associate the policy definition with the activity.

Policy Parameter	**Value**
Event	Process Started
Action	Invoke SAS Code
Description	
Source URI	`http://localhost/InvokeSASCodeTutorial.sas`
Logical Server Name	SASApp – Logical Workspace Server
Repository Name	Foundation
Result Macro Variable	result
Return Data Object	InvokeSASCode Tutorial/Tutorial Activity/Total
Pass all root process data objects	disabled
Parameter 1 Data Object	InvokeSASCode Tutorial/Tutorial Activity/Total Category
Parameter 1 Macro Variable	var
Policy Label	Process SAS Code->Invoke SAS Code

The parameters for this policy example include the following:

Source URI
: is the location of the SAS code as served by the Web application server.

Logical Server Name
: is the name of the Logical Workspace Server that executes the SAS code.

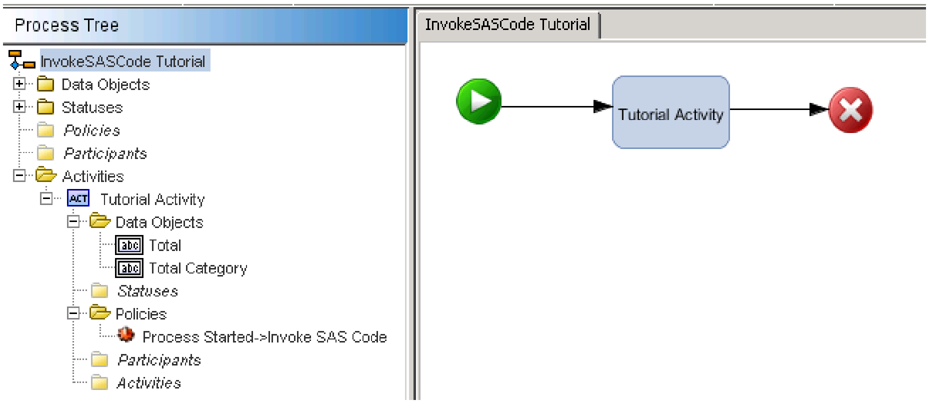
Repository Name
: is the name of the metadata repository in which the specified logical server is defined.

Result Macro Variable
: is the macro variable (result) in the SAS code that contains the value that is returned to the workflow in the specified data object.

Return Data Object
: is the workflow data object (Total) that holds the value from the Result Macro Variable.

Parameter 1 Data Object
: is the workflow data object (Total Category) whose value is passed to the SAS code in the macro variable specified by Parameter 1 Macro Variable (var).

The final workflow definition is as follows:



Up to five data objects can be passed to the SAS code via corresponding macro variables. Since this example only requires one, the following line of code is prepended to the SAS code prior to execution:

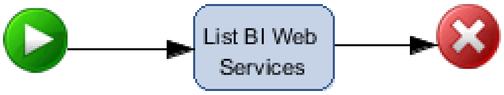
```
%let var=sales;
```

Refer to the examples provided with SAS Workflow Studio for details of this example.

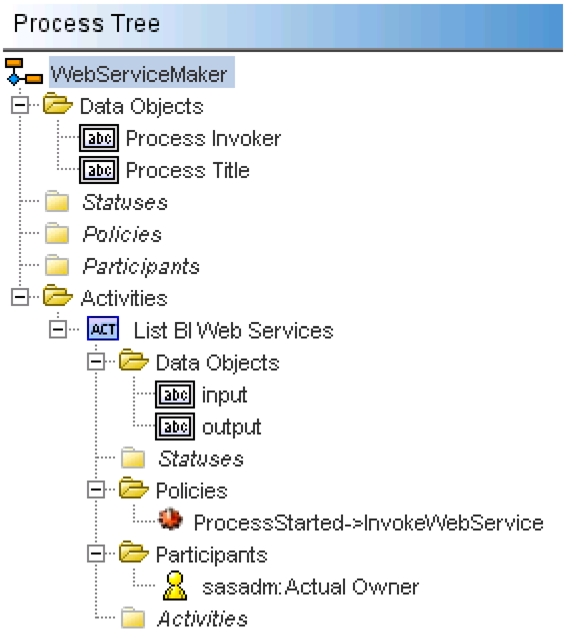
Invoke Web Service Policy Example

This example illustrates a simple sequence with a single task. Once started, the activity executes the invokeWebService policy sending a ListWebServices SOAP request to the SAS BI Platform WebServiceMaker Web service. The XML for the request is provided in the input data object and the response is stored in the output data object. The activity is assigned to sasadm as actual owner, so you can perform the task to complete, and stop the workflow.

First, define the relevant workflow and data objects. The following figure shows a simple example where the policy data objects are associated with the activity and are visible in the entire subtree:



The following figure shows the process tree for the example process:



Process Tree

- WebServiceMaker
  - Data Objects
    - Process Invoker
    - Process Title
  - *Statuses*
  - *Policies*
  - *Participants*
  - Activities
    - List BI Web Services
      - Data Objects
        - input
        - output
      - *Statuses*
      - Policies
        - ProcessStarted->InvokeWebService
      - Participants
        - sasadm:Actual Owner
      - *Activities*

The data objects in the List BI Web Services activity have the following properties:

Data Object Label	Type	Value
input	XML Object	**`<web:ListWebServices xmlns:web="http://support.sas.com/xml/namespace /biwebservices/webservicemaker-9.2"/>`**
output	XML Object	None *Note:* The Web service response is stored as the value for this data object during policy execution.

Both data objects use the following schema:

```
<schema elementFormDefault="qualified"
        targetNamespace="http://support.sas.com/xml/namespace/biwebservices/webservicemaker-9.2"
        xmlns="http://www.w3.org/2001/XMLSchema"
        xmlns:tns="http://support.sas.com/xml/namespace/biwebservices/webservicemaker-9.2"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <complexType name="StringArrayType">
      <sequence>
         <element maxOccurs="unbounded" minOccurs="1" name="string" type="string"/>
      </sequence>
   </complexType>
   <element name="MakeWebService">
      <complexType>
         <sequence>
            <element name="storedProcessPaths" type="tns:StringArrayType"/>
            <element name="serviceName" type="string"/>
            <element minOccurs="0" name="serviceNamespace" type="string"/>
            <element minOccurs="0" name="keywords" type="tns:StringArrayType"/>
            <element default="true" minOccurs="0" name="publishMetadata"
             type="boolean"/>
         </sequence>
      </complexType>
   </element>
   <element name="MakeWebServiceResponse">
      <complexType>
         <sequence>
            <element name="MakeWebServiceResult" type="string"/>
         </sequence>
      </complexType>
   </element>
   <element name="ListWebServices">
      <complexType/>
   </element>
   <element name="ListWebServicesResponse">
      <complexType>
         <sequence>
            <element name="ListWebServicesResult" type="tns:StringArrayType"/>
         </sequence>
      </complexType>
   </element>
   <element name="RemoveWebService">
      <complexType>
         <sequence>
            <element name="serviceName" type="string"/>
         </sequence>
      </complexType>
   </element>
   <element name="RemoveWebServiceResponse">
      <complexType/>
   </element>
   <element name="Fault" type="tns:Fault"/>
   <complexType name="Fault">
      <sequence>
         <element name="Exception" type="tns:Exception"/>
      </sequence>
      <attribute name="code" type="string"/>
   </complexType>
   <complexType name="Exception">
      <sequence>
         <element minOccurs="0" name="Exception" type="tns:Exception"/>
      </sequence>
      <attribute name="message" type="string"/>
   </complexType>
</schema>
```

Next, associate the policy definition with the activity.

Policy Parameter	Value
Event	Process Started
Action (Policy)	Invoke Web Service

Policy Parameter	**Value**
Description	This policy is used to invoke a Web service function over SOAP/HTTP.
Endpoint URL	**`http://localhost:8080/SASBIWS/services/WebServiceMaker`**
Action (Web Service)	**`http://support.sas.com/xml/namespace/biwebservices/webservicemaker-9.2/ListWebServices`**
Input Data Object	WebServiceMaker/List BI Web Services/input
Output Data Object	WebServiceMaker/List BI Web Services/output
Username	sasadm@saspw
Password	*password*
Policy Label	Process SAS Code**->**Invoke SAS Code

The final response stored in the output data object is as follows:

```
<n:ListWebServicesResponse
 xmlns:n="http://support.sas.com/xml/namespace/biwebservices/webservicemaker-9.2"
 xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
   <n:ListWebServicesResult>
      <n:string>WebServiceMaker</n:string>
      <n:string>XMLA</n:string>
   </n:ListWebServicesResult>
</n:ListWebServicesResponse>
```

Refer to the examples provided with SAS Workflow Studio for details of this example.

Send Notification Using SAS Template Service Policy Example

The following policies support notifications and e-mail:

Notify Participant
: is used to send a notification via the SAS Alert Notification Service to a participant as defined by their preferences. The notification can be configured implicitly when an activity starts or finishes by selecting a check box for the activity. Alternatively, it can be defined explicitly as a policy for any other supported workflow events.

Note: The format for this notification type cannot be customized.

Send Email
: is used to send e-mail notifications via the SAS Mail Service using the information provided in the policy definition.

Note: The format for this notification type cannot be customized.

Note: This policy was formerly named Send Notification.

Send Notification By Data Object
is used to send notifications via the SAS Alert Notification Service based on the information stored in various workflow data objects.

Note: The format for this notification type cannot be customized.

Note: This notification policy is supported for legacy usage and does not leverage the SAS platform user preference settings regarding alert and e-mail support. Therefore, one of the other notification policies should be used.

Send Notification By Workflow Role
is used to notifications via the SAS Alert Notification Service based on the workflow role of a user.

Note: The format for this notification type cannot be customized.

Note: This notification policy is supported for legacy usage and does not leverage the SAS platform user preference settings regarding alert and e-mail support. Therefore, one of the other notification policies should be used.

Send Workflow Group Notification
is used to send a workgroup event notification, which triggers the SAS Alert Notification service to generate end-user notification messages. This policy is used to send notifications to a group or set of recipients and where all recipients are required to be addressed in the same message. This capability is useful in collaboration scenarios or where the event should be copied to all interested parties. E-mail is the only delivery channel supported using the group notification policy.

Note: There are five expiry options available: None, Remove, Reroute, Resend, and Start Workflow. For more information, see "Expiry Options for Notification Policies" on page 44.

Send Workflow Notification
is used to send a directed workgroup notification via the SAS Alert Notification service. This type of notification can be sent to users via e-mail, SMS message or displayed in a portlet and leverage the SAS Template Service to support custom formats.

Note: There are five expiry options available: None, Remove, Reroute, Resend, and Start Workflow. For more information, see "Expiry Options for Notification Policies" on page 44.

The Approval Escalation Process example demonstrates two types of timer policy usage as well as sending a notification using the SAS Template Service. The Timer Expired Event policy triggers a periodic notification to the group assigned to the Review Order task with the Send Workflow Notification action. The notification-related policy parameters are as follows:

Policy Parameter	**Value**
Recipient(s)	
Group Recipient(s)	${../Account Manager}
Description	
Template	SAS_Email_Message

Policy Parameter	Value
Directive	
Notification Variables	../MSG, ../DESC
HTTP Parameter	

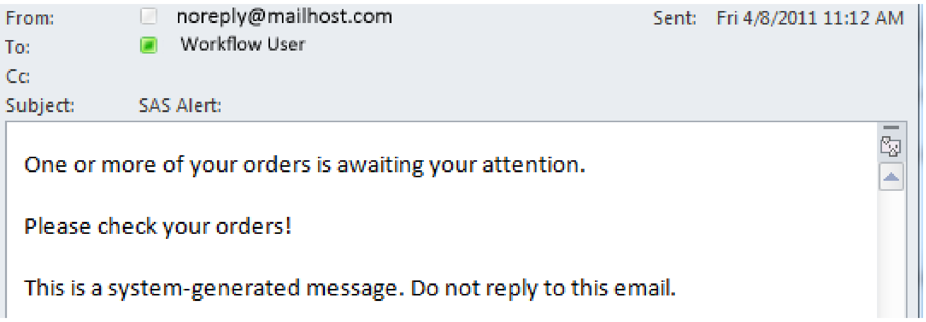
There are no individual recipients for this notification, but the Account Manager group is assigned indirectly via data object substitution. For more information, see “Data Object Substitution” on page 52.

The default value for this data object is US Region Accountants, so any users assigned to this group in SAS metadata receive an alert each time period as defined in the previous table. The SAS_Email_Message template is used, which is also the default if no template is specified.

File A2.1 *SAS_Email_Message Template*

```
$if(_SAS_TEMPLATE_TXT)$
The issue priority is: $ISSUEPRI$
$DESC$
--------------------------------------------
$TO$
$MSG$
This is a system-generated message. Do not reply to this email.
$endif$
```

This template contains three variables: DESC, TO and MSG. This policy example designates values for DESC and MSG by listing them as Notification Variables, which associates the values for the data objects within the example with the template. Here is the resulting e-mail message:



From: noreply@mailhost.com Sent: Fri 4/8/2011 11:12 AM
To: Workflow User
Cc:
Subject: SAS Alert:

One or more of your orders is awaiting your attention.

Please check your orders!

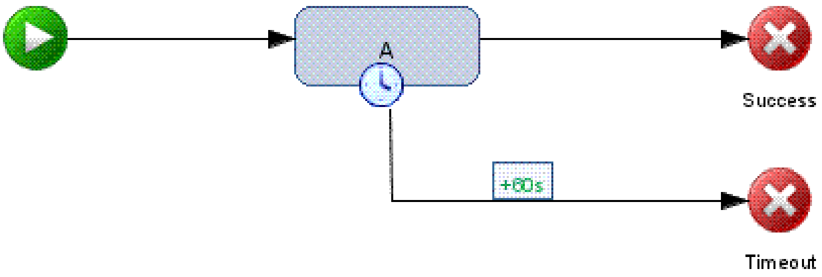
This is a system-generated message. Do not reply to this email.

Refer to the examples provided with SAS Workflow Studio for details of this example.

Appendix 3
Timer Examples

Timer on Border of Activity

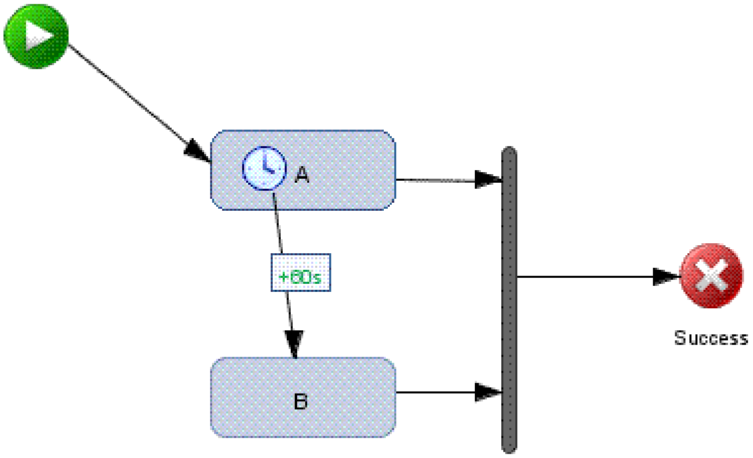
The timer is triggered 60 seconds after Activity A starts. When the timer fires, Activity A is stopped because it is located on its border. If the timer fires, then the transition to the Stop node is traversed. This causes this process instance to be terminated with a status of Timeout.



If Activity A is completed before the timer expires, then the timer is stopped, and the transition to the Stop node is traversed. This causes the instance to be stopped with a status of Success.

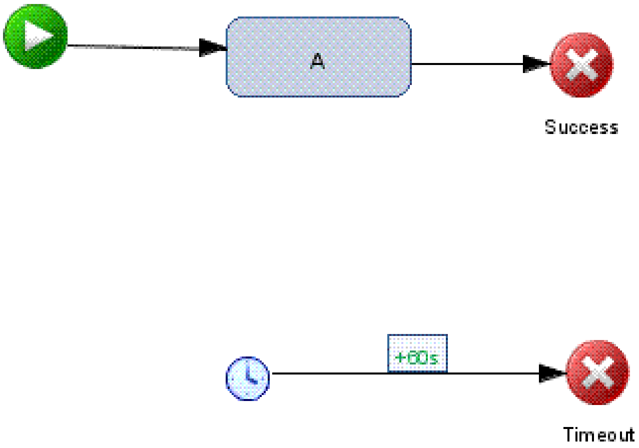
Timer Inside Activity Border

Activity A triggers Activity B 60 seconds after Activity A starts. Because the Timer is inside the border of Activity A, Activity A continues to be active after the timer is triggered. Therefore, after the timer has fired, both Activity A and Activity B are active (timer-controlled parallel processing).



Automatically Traversing a Transition

The timer is triggered 60 seconds after the process instance starts. If Activity A has not completed when the timer fires, then the transition to the Stop node is traversed. This causes the instance to be stopped with a status of **`Timeout`**. If Activity completes before 60 seconds, then the transition to the Stop node is traversed. This causes the instance to be stopped with a status of Success.



Timer Expression Examples

The following example specifies a schedule expression of **`+1d`**. This indicates that the timer is triggered in 1 day.

Timer Parameter	Value
Schedule Expression	+1d
Recurrence: Number of Repetitions[1]	0
Recurrence: Interval of Repetitions	None
End Timer	Disabled

Cron expressions are useful for defining the schedule time using a format similar to the UNIX **`cron`** command. These expressions are particularly convenient if you want the timer to expire based on calendar information, rather than exact dates or intervals. Refer to the org.quartz.CronExpression documentation for detailed syntax.

Note: The repeat count and repeat expression fields are ignored when a cron expression is specified.

The following example specifies a schedule expression of **`0 0 12 * * ?`**. This cron expression specifies a timer to fire every day at 12 p.m. (noon).

Timer Parameter	Value
Schedule Expression	**`0 0 12 * * ?`**
Recurrence: Number of Repetitions	0
Recurrence: Interval of Repetitions	None
End Timer	Disabled

The following example defines a timer that is scheduled to fire 1 day (**`+1d`**) after the process is started. No repeat count is specified, therefore the timer repeats indefinitely. The Interval of Repetitions is specified as **`+24h`**, which means that it repeats every 24 hours.

Timer Parameter	Value
Schedule Expression	+1d
Recurrence: Number of Repetitions	None
Recurrence: Interval of Repetitions	+24h
End Timer	Disabled

The following example is similar to the previous one, except that a Timer End value is specified. The Timer End value indicates when the timer should terminate. In this example, the timer will repeat every 24 hours and will cease after 1 week (as specified the value **`+1w`** for the Timer End).

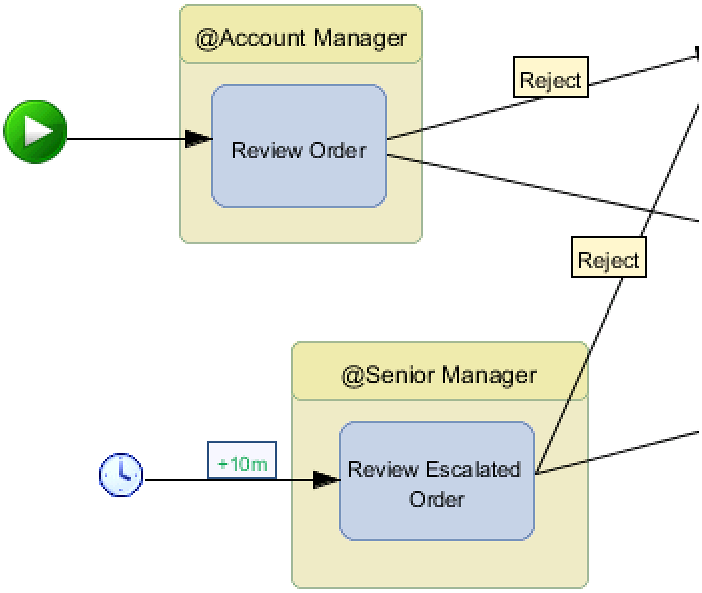
[1] If this value is left blank or equals -1, then there are unlimited repetitions until the containing activity or process ends.

Timer Parameter	Value
Schedule Expression	+1d
Recurrence: Number of Repetitions	Disabled
Recurrence: Interval of Repetitions	+24h
End Timer	+1w

Timer Expired and Tool Policy Example

The following example illustrates both the timer expired event and the timer tool policies. This workflow is initiated when the system receives an order. Then, an account manager must access the relevant ordering system, review the order details and either approve or reject the order. If the order is not reviewed within a certain time threshold, then the users within the Account Manager group are notified by e-mail up to three times before the order is escalated and reassigned to a Senior Manager. The remaining workflow activities are similar to those in “Basic Approval Process” on page 95.

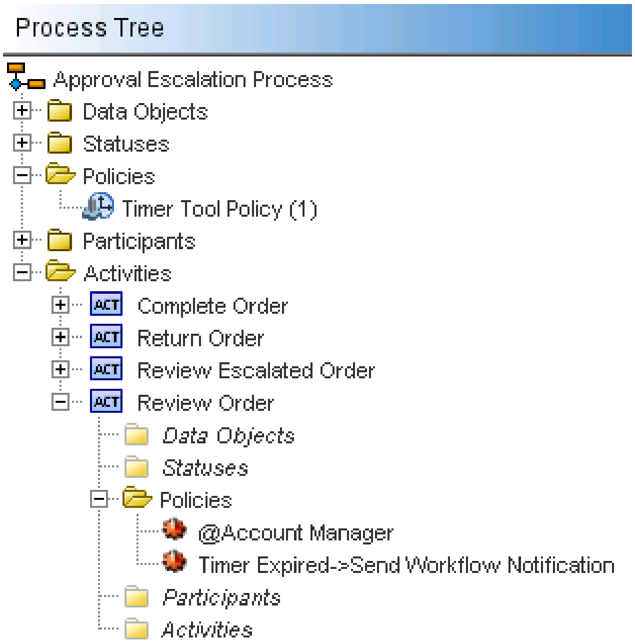
Figure A3.1 *Timer Expired Process Diagram*



Refer to the e-mail example for an example of SAS Templates usage.

The following figure shows the process tree for the Timer Expired process:

Figure A3.2 *Timer Expired Process Tree*



Process Tree

- Approval Escalation Process
  - Data Objects
  - Statuses
  - Policies
    - Timer Tool Policy (1)
  - Participants
  - Activities
    - Complete Order
    - Return Order
    - Review Escalated Order
    - Review Order
      - *Data Objects*
      - *Statuses*
      - Policies
        - @Account Manager
        - Timer Expired->Send Workflow Notification
      - *Participants*
      - *Activities*

The timer policy associated with the initial Review Order task (Timer Expired->Send Workflow Notification) has the following parameter values:

Policy Parameter	**Value**
Event	Timer Expired
Action	Send Workflow Notification
Description	This policy is used to send a notification. Each recipient is addressed in their own notification.
Recipient(s)	
Group Recipient(s)	${../Account Manager}
Description	
Template	SAS_Email_Message
Directive	
Notification Variables	../MSG, ../DESC
HTTP Parameters	
Action on Expiry	None

Policy Parameter	Value
Policy Label	Timer Expired->Send Workflow Notification

The schedule parameter values in the following table trigger an e-mail notification after three minutes (schedule expression) and twice (number of repetitions) more after two minutes (repetition interval).

Policy Parameter	Value
Schedule Expression	+3m
Recurrence: Number of Repetitions	2
Recurrence: Interval of Repetitions	+2m
End Timer	Disabled

The timer policy associated with the main workflow has the following parameters:

Policy Parameter	Value
Schedule Expression	+10m
Recurrence: Number of Repetitions	0
Recurrence: Interval of Repetitions	
End Timer	Disabled

The schedule parameter values in the table above will trigger a transition to the Review Escalated Order activity after 10 minutes (schedule expression) without repetition (0 or -1 for number of repetitions). This activity has another policy associated with it to stop the initial Review Order task because the order has been automatically reassigned.

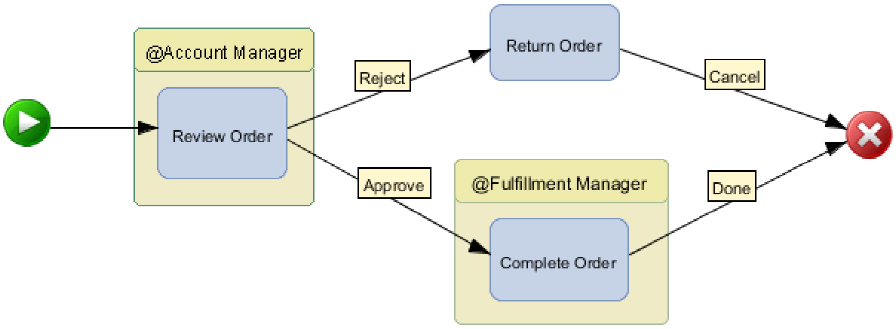
Refer to the examples provided with SAS Workflow Studio for details of this example.

Appendix 4
Basic Workflow Examples

Basic Approval Process

The Basic Approval process template illustrates one type of alternate flow control where only a single path is taken based on the completion status of the previous task. This workflow is initiated when the system receives an order. Then, an account manager must access the relevant ordering system, review the order details and either approve or reject the order. If the order is rejected, then it is returned by the system. If the order is approved, then a fulfillment manager must complete the order.

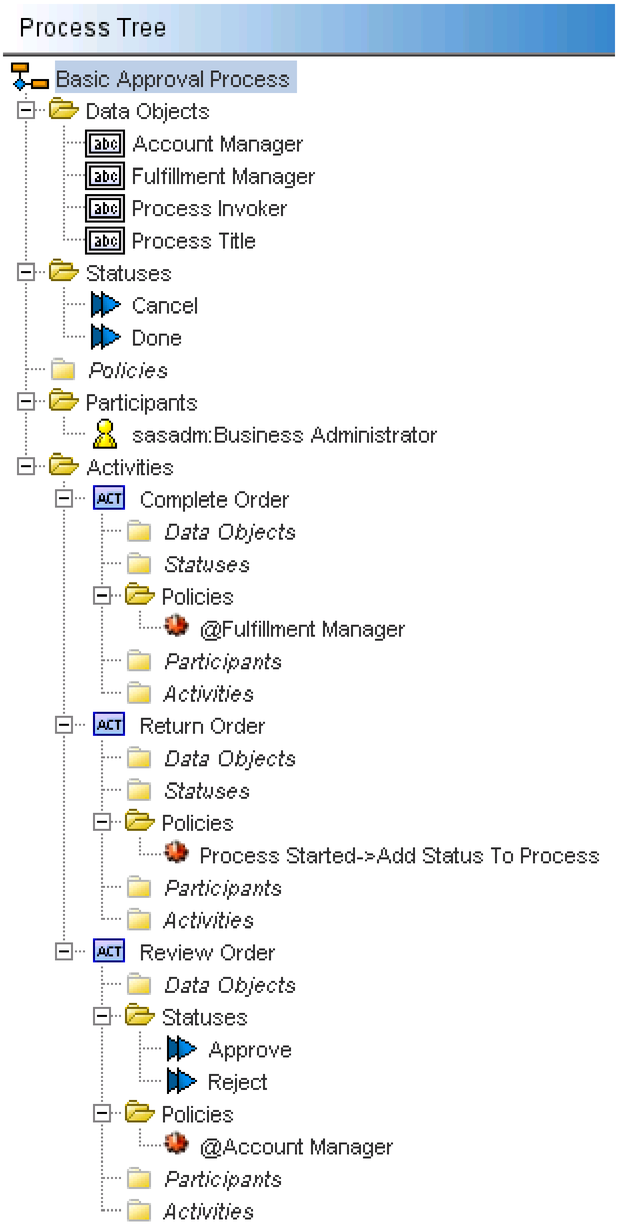
Figure A4.1 *Basic Approval Process Diagram*



This process uses the Exclusive Choice workflow pattern.

The following figure shows the process tree for the Basic Approval process:

Figure A4.2 *Basic Approval Process Tree*



Process Tree

- Basic Approval Process
  - Data Objects
    - Account Manager
    - Fulfillment Manager
    - Process Invoker
    - Process Title
  - Statuses
    - Cancel
    - Done
  - *Policies*
  - Participants
    - sasadm:Business Administrator
  - Activities
    - Complete Order
      - *Data Objects*
      - *Statuses*
      - Policies
        - @Fulfillment Manager
      - *Participants*
      - *Activities*
    - Return Order
      - *Data Objects*
      - *Statuses*
      - Policies
        - Process Started->Add Status To Process
      - *Participants*
      - *Activities*
    - Review Order
      - *Data Objects*
      - Statuses
        - Approve
        - Reject
      - Policies
        - @Account Manager
      - *Participants*
      - *Activities*

The following table provides details about the elements of the Basic Approval process:

Table A4.1 *Basic Approval Process Elements*

Element Name	**Element Type**	**Details**
Process Invoker	Data Object	Default data object of type Short Text defined for all processes. Holds the user name of the person who starts an instance of this process.*

Element Name	Element Type	Details
Process Title	Data Object	Default data object of type Short Text defined for all processes. Holds the title for a process instance.*
Account Manager	Data Object	Data object of type Short Text defined for all processes. Holds the group name used by the @Account Manager swimlane policy.* The default value is US Region Accountants.
Fulfillment Manager	Data Object	Data object of type Short Text defined for all processes. Holds the group name used by the @Fulfillment Manager swimlane policy.* The default value is US Region Supply Managers.
Cancel	Status	Status value defined for all processes.**
Done	Status	Status value defined for all processes.**
sasadm	Participant	This is a User type participant assigned to the root of the process with the workflow role of Business Administrator. This is a default SAS administrator user configured with the platform in SAS metadata. The Business Administrator role supports certain administrative tasks such as adding comments, delegating or transferring the task, or releasing the task locked by another user.
Review Order	Task (Atomic Activity)	This is a manual task that represents the user interacting with the system. Once the user completes the review and submits the relevant status, the relevant status is assigned when this task completes. The next activity is selected based on the status value as assigned to the transition.

Element Name	Element Type	Details
@Account Manager	Policy/Swimlane	The swimlane corresponds to a group participant defined in the process data object Account Manager. The value in the data object is assigned at run time using this policy. Only users who are members of this group can review orders. The group is an implicit value. Therefore, the value assigned to the data object must exist as a group in SAS metadata. For example, the default value is US Region Accountants, so if no initial value assignment is made for this data object when starting a workflow instance, there must be a group defined with that name in SAS metadata. Because the workflow role assigned is Potential Owner, any user from that group can claim the task. Once it is claimed, the user becomes the Actual Owner, the task is locked and no others users can perform the activity.[***]
Approve	Status	Status value defined for Review Order.[†]
Reject	Status	Status value defined for Review Order.[†]
Return Order	Task (Atomic Activity)	This is a system task (that is, automated activity) that initiates if the order is rejected. Once completed, the workflow instance ends with a status of Cancel because this activity transitions to the terminate node.
Process Started->Add Status to Process	Policy	This policy assigns the Cancel status because it is an automated activity (that is, is not explicitly performed like the previous manual task). This status value stops the Return Order activity and triggers the final transition into the terminate node.

Element Name	Element Type	Details
Complete Order	Task (Atomic Activity)	This is a manual task that represents the user interacting with the system. This activity starts if the order is approved. Then, the workflow instance ends with a status of Done because this activity transitions to the terminate node.
@Fulfillment Manager	Policy/Swimlane	The swimlane corresponds to a group participant defined in the process data object Fulfillment Manager. The value in the data object is assigned at run time using this policy. Only users who are members of this group can complete orders. The group is an implicit value. Therefore, the value assigned to the data object must exist as a group in SAS metadata. For example, the default value is US Region Supply Managers, so if no initial value assignment is made for this data object when starting a workflow instance, there must be a group defined with that name in SAS metadata. Because the workflow role assigned is Potential Owner, any user from that group can claim the task. Once it is claimed, the user becomes the Actual Owner, the task is locked and no others users can perform the activity.***

* This data object value is defined at the root process level and is set as visible within the entire subtree by default. This setting allows access to the value by decisions and policies at run time by all contained activities and subprocesses, but is not accessible as a local copy of the data object.

** This status value is defined at the root process level and is set as visible within the entire subtree by default. This setting allows access to the value by decisions and policies at run time by all contained activities and subprocesses, but is not accessible as a local copy of the status.

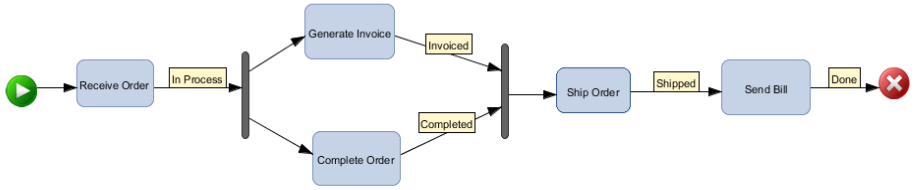
*** Only users can be assigned to the Actual Owner workflow role. Participants of SAS Metadata type group or role type must be assigned as Potential Owner and their user members must claim the task to become Actual Owner.

† This status value is defined at the activity level and is set as visible within the entire subtree by default. This setting allows access to the value by decisions and policies at run time by all contained activities and subprocesses, but is not accessible as a local copy of the status.

Order Fulfillment Process

The Order Fulfillment process template illustrates parallel flow control where two paths are processed concurrently and the converged path does not start until both of the parallel activities finish. This workflow is initiated when the system receives an order. Then, a billing associate generates an invoice while a fulfillment manager completes the order. Once both of these tasks are completed, the fulfillment manager ships the order. Finally, once the order has shipped, the billing associate sends the bill.

Figure A4.3 *Order Fulfillment Process Diagram*

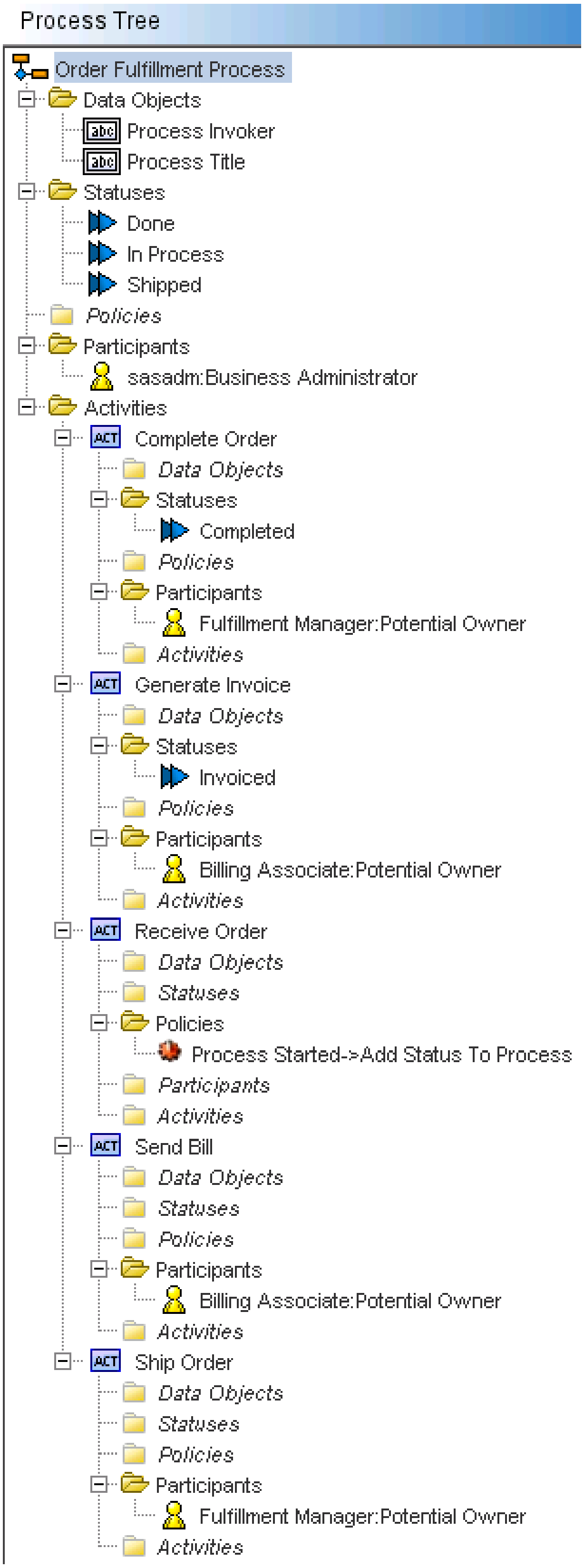


This process uses the following workflow patterns:

- Parallel
- Join (Merge with Synchronization)

The following figure shows the process tree for the Order Fulfillment process:

Figure A4.4 *Order Fulfillment Process Tree*



```
Process Tree
Order Fulfillment Process
  Data Objects
    Process Invoker
    Process Title
  Statuses
    Done
    In Process
    Shipped
  Policies
  Participants
    sasadm:Business Administrator
  Activities
    Complete Order
      Data Objects
      Statuses
        Completed
      Policies
      Participants
        Fulfillment Manager:Potential Owner
      Activities
    Generate Invoice
      Data Objects
      Statuses
        Invoiced
      Policies
      Participants
        Billing Associate:Potential Owner
      Activities
    Receive Order
      Data Objects
      Statuses
      Policies
        Process Started->Add Status To Process
      Participants
      Activities
    Send Bill
      Data Objects
      Statuses
      Policies
      Participants
        Billing Associate:Potential Owner
      Activities
    Ship Order
      Data Objects
      Statuses
      Policies
      Participants
        Fulfillment Manager:Potential Owner
      Activities
```

The following table provides details of the elements in the Order Fulfillment process:

Table A4.2 *Order Fulfillment Process Elements*

Element Name	Element Type	Details
Process Invoker	Data Object	Default data object of type Short Text defined for all processes. Holds the user name of the person who starts an instance of this process.*
Process Title	Data Object	Default data object of type Short Text defined for all processes. Holds the title for a process instance.*
Done	Status	Default status value defined for all processes.**
In Process	Status	Status value defined for entire process.**
Shipped	Status	Status value defined for entire process.**
sasadm	Participant	This is a User type participant assigned to the root of the process with the workflow role of Business Administrator. This is a default SAS administrator user configured with the platform in SAS metadata. The Business Administrator role supports certain administrative tasks such as adding comments, delegating or transferring the task, or releasing the task locked by another user.
Receive Order	Task (Atomic Activity)	This is an automated system task.
Process Started->Add Status to Process	Policy	This policy assigns the In Process status because it is an automated activity (that is, is not explicitly performed like the other manual tasks). This status value stops the Receive Order activity and triggers the transition into the merge/fork gateway.

Element Name	Element Type	Details
Generate Invoice	Task (Atomic Activity)	This is a manual task that represents the user interacting with the system. This activity starts after the order is received. This task represents invoice generation by the billing associate, which is completed by setting the status to Invoiced.
Billing Associate	Participant	This is a group type participant assigned to the activity level of the process with the workflow role of Potential Owner. Only users who are members of this group can generate invoices and send bills. The group is an explicit value, so it must exist in SAS metadata. Because the workflow role assigned is Potential Owner, any user from that group can claim the task. Once it is claimed, the user becomes the Actual Owner, the task is locked and no others users can perform the activity.[***]
Invoiced	Status	Status value defined for Generate Invoice.[†]
Complete Order	Task (Atomic Activity)	This is a manual task that represents the user interacting with the system. This activity starts after the order is received. This task represents order completion by the fulfillment manager, which is completed by setting the status to Completed.

Element Name	Element Type	Details
Fulfillment Manager	Participant	This is a group type participant assigned to the activity level of the process with the workflow role of Potential Owner. Only users who are members of this group can complete and ship orders. The group is an explicit value, so it must exist in SAS metadata. Because the workflow role assigned is Potential Owner, any user from that group can claim the task. Once it is claimed, the user becomes the Actual Owner, the task is locked and no others users can perform the activity.***
Completed	Status	Status value defined for Complete Order.†
Ship Order	Task (Atomic Activity)	This is a manual task that represents the user interacting with the system. This activity starts after both the invoice has been generated and the order has been completed. This task represents order shipment by the fulfillment manager, which is completed by setting the status to Shipped.
Send Bill	Task (Atomic Activity)	This is a manual task that represents the user interacting with the system. This activity starts after the order has been shipped. This task represents bill delivery by the billing associate, which is completed by setting the status to Done.

* This data object value is defined at the root process level and is set as visible within the entire subtree by default. This setting allows access to the value by decisions and policies at run time by all contained activities and subprocesses, but is not accessible as a local copy of the data object.

** This status value is defined at the root process level and is set as visible within the entire subtree by default. This setting allows access to the value by decisions and policies at run time by all contained activities and subprocesses, but is not accessible as a local copy of the status.

*** Only users can be assigned to the Actual Owner workflow role. Participants of SAS Metadata type group or role type must be assigned as Potential Owner and their user members must claim the task to become Actual Owner.

† This status value is defined at the activity level and is set as visible within the entire subtree by default. This setting allows access to the value by decisions and policies at run time by all contained activities and subprocesses, but is not accessible as a local copy of the status.

Glossary

access control list (ACL)

a list of participants

access control element (ACE)

See participant.

activity

a step or unit of work in a business process or workflow. In SAS Workflow Studio, the representation of a workflow task and its relationship to tasks preceding and following it, including the data, policies, and participants required to fulfill the task (as a node in the process tree). An activity can be manual or automated (via script or program). SAS Workflow Studio also uses activities to represent subprocesses as a stacked set of activities.

actor

See participant.

annotation

a documentation artifact for a process that provides supporting information about the process or elements within the process. These notes are for presentation only and are not associated with the run-time process definition.

business activity monitoring (BAM)

an assembly of functionality that delivers alerts to end users so that they can proactively work to resolve issues. BAM solutions are typically built on top of a service-oriented architecture, integrated into business applications, or run as a stand-alone product. BAM can be assembled from individual components including workflow engines, rules engines, notification services, and other BI tools.

business process

a set of coordinated tasks and activities, conducted by both people and systems that lead to accomplishing a specific organizational goal. ITIL definition: A process that is owned and carried out by the business. A business process contributes to the delivery of a product or service to a business customer. For example, a retailer can have a purchasing process that helps to deliver services to their business customers. Many business processes rely on IT services. *See also* process.

business process management (BPM)

a systematic approach to improving a company's business processes. BPM activities seek to make business processes more effective, more efficient, more pervasive, and more agile so that they can adapt to changing business environments.

Business Process Modeling Notation (BPMN)

a standard modeling notation for process modelers, users, analysts, and so on, developed by the Business Process Management Institute.

collaboration

an approach in which people jointly contribute to the development of an idea or work product using technologies to share documents and other idea-generating artifacts. Collaboration is typically needed in applications that require simultaneous contributions on a work product that is managed from a central location, for example in a meeting space or document repository.

data object

represents the business data values required by SAS Workflow to execute the workflow activities. Data objects can mirror values retrieved from external data systems. In this way, SAS Workflow is only loosely coupled with external data systems. Just as global variables provide shared values to code, data objects on the process can provide values to all activities and policies within a process. Data objects can also be defined on an activity as local objects.

decision gateway

control the flow of a process. Decision gateways handle the management of alternate paths within a process. Formerly known as decision node.

definition

See process definition.

event

a message that captures information associated with a specific occurrence. Events are used to trigger policies or process actions.

gateway

a process diagram element that controls the flow of a process. Types of gateways include logic gateway, merge/fork gateway, and decision gateway.

logic gateway

a process diagram element that controls the flow of a process through use of AND or OR logic to control the merging and forking of sequence flows in a process.

merge/fork gateway

a process diagram element that controls the flow within the business process through either the convergence of two or more paths to one or more paths or the divergence one or more paths to two or more paths.

note

See annotation.

operand

the former name for data object

participant

a resource that performs the work represented by a workflow activity instance. This work is normally manifested as one or more work items assigned to the workflow participant via the worklist. The list of users, groups, or roles defined in SAS metadata are mapped to standard roles in workflow.

policy

represents executable logic and instructions that are relevant to the process. They allow processes to be automated in an extremely flexible manner. These actions are customizable using parameters defined for each action. Policies associate the executable logic with status change and timer events in the process.

process

the repeatable series of steps that must be accomplished in order to realize a business objective or policy goal, normally within the context of an organizational structure defining functional roles and relationships. With SAS Workflow, process authors define the process and its data in SAS Workflow Studio, and managers, administrators, and end-users start processes and interact with them using corresponding API calls from the service interface. In SAS Workflow Studio, denotes the highest level process as a root node in the process tree. Subprocesses are represented as a stacked set of activities in the diagram, but are also denoted as processes ( ) in the process tree.

process definition

an activated process template that is available in the SAS platform for use. In a SAS workflow, process definitions are populated at run time as process instances.

process template

a specification of business activities with associated actions and business logic that comprise a process. In SAS Workflow, a template can be stored as an XML file or stored in the SAS Content Server.

process tree

part of the SAS Workflow Studio user interface for displaying and accessing all of the activities, data objects, statuses, policies, and participants for the process open in the diagram editor.

process instance

a running process artifact in a SAS workflow.

sequence flow

the connection between process elements used to designate the order in which activities are performed in a process. Each sequence flow has only one source and only one target.

state

the condition of a process instance or activity at any point in time. An instance can exhibit only a single state at any point in time, and valid states for an instance include Started, Finished, Aborted, and Unstarted.

state change

a transition from one state to another. For example, a process state changes from Unstarted to Started. Note that only a process or activity has state.

status

an outcome of an activity. In SAS Workflow, status objects are used to trigger preconditions, state changes, and policies. Policy objects use status information as part of the data required to execute an automated procedure. In SAS Workflow Studio, statuses have descriptive names such as Done, Cancel, or Accept to illustrate what is occurring in a process.

stop node
a termination node for a process. There can be more than one stop node in a process definition.

subprocess
a process contained by another process and is a type of activity.

swimlane
a graphical designation for assignment of activities to roles, groups, or users. Swimlanes are used to group activities with the same role. They can be used to explicitly assign activities to a participant object or they can be used to implicitly assign via a swimlane policy. The swimlane policy assigns the user, group, or role value to a process variable, represented by a data object, at run time.

task
a type of activity performed by a user or application (for example, by a Web service), which cannot be broken down to finer level of granularity. *See also* activity.

task list
a list of process instance activities. *See also* worklist.

template
an XML file stored on disk or in SAS Content Server containing a representation of the process definition core elements. Also see Process Template.

transition
represents a possible change in process status. Transitions can also represent either the change in an activity's state or status.

workflow
the tasks, procedural steps, people or organizations involved, required inputs and outputs, and tools needed for each step in a business process. *See also* process.

workflow engine
the component in a workflow automation system that understands and directs all of the procedures, steps in a procedure, and rules for handling each step. The workflow engine determines whether a process is ready to move to its next step (for example, a change in state). Some vendors sell workflow automation products for particular industries such as insurance or banking, or for commonly used processes such as handling product service calls. SAS workflow is a set of services within the SAS Web Infrastructure Platform and is not a separate server.

worklist
a list of process instance activities. *See also* task list.