

SAS® Workflow Studio 1.2

User's Guide



The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2011. *SAS® Workflow Studio 1.2: User's Guide®*. Cary, NC: SAS Institute Inc.

SAS® Workflow Studio 1.2: User's Guide

Copyright © 2011, SAS Institute Inc., Cary, NC, USA

All rights reserved. Produced in the United States of America.

For a hardcopy book: No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the publisher, SAS Institute Inc.

For a Web download or e-book: Your use of this publication shall be governed by the terms established by the vendor at the time you acquire this publication.

The scanning, uploading, and distribution of this book via the Internet or any other means without the permission of the publisher is illegal and punishable by law. Please purchase only authorized electronic editions and do not participate in or encourage electronic piracy of copyrighted materials. Your support of others' rights is appreciated.

U.S. Government Restricted Rights Notice: Use, duplication, or disclosure of this software and related documentation by the U.S. government is subject to the Agreement with SAS Institute and the restrictions set forth in FAR 52.227–19, Commercial Computer Software-Restricted Rights (June 1987).

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

1st electronic book, December 2011

2nd electronic book, April 2012

SAS® Publishing provides a complete selection of books and electronic products to help customers use SAS software to its fullest potential. For more information about our e-books, e-learning products, CDs, and hard-copy books, visit the SAS Publishing Web site at

support.sas.com/publishing or call 1-800-727-3228.

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.

Contents

<i>What's New in SAS Workflow Studio 1.2</i>	<i>v</i>
<i>Accessibility Features in SAS Workflow Studio 1.2</i>	<i>vii</i>
Chapter 1 • Introduction to SAS Workflow Studio	1
What Is Business Process Management?	1
Why Use SAS Workflow Studio?	2
Chapter 2 • SAS Workflow Architecture	5
Overview	5
Overview of SAS Workflow Studio	6
Workflow Lifecycle	7
Chapter 3 • Navigating SAS Workflow Studio	9
Overview	9
Menus and Toolbars	10
Using the Navigator Pane	15
Using the Diagram Editor	16
Using the Process Tree	18
Using the Clip Explorer	19
Chapter 4 • Defining Processes with SAS Workflow Studio	21
Creating a Process	22
Working with Activities	22
Working with Statuses	26
Working with Participants	28
Working with Policies	31
Working with Data Objects	49
Additional Process Features	50
Chapter 5 • Advanced Topics	55
Workflow Patterns	55
Using Timers	64
Deploying and Maintaining Processes	66
SAS Alert Notification Templates	70
Appendix 1 • Workflow Loader Utility	73
Appendix 2 • Decision Expression Examples	75
Operator Support in Decision Expressions	75
Function Support in Decision Expressions	77
Decision Expression Examples	77
Appendix 3 • Policy Usage Examples	79
HTTP Request Policy Example	79
Invoke SAS Code Policy Example	80
Invoke Web Service Policy Example	82
Send Notification Using SAS Template Service Policy Example	85
Appendix 4 • Timer Examples	89

Timer on Border of Activity	89
Timer Inside Activity Border	89
Automatically Traversing a Transition	90
Timer Expression Examples	90
Timer Expired and Tool Policy Example	92
Appendix 5 • Basic Workflow Examples	95
Basic Approval Process	95
Order Fulfillment Process	98
Glossary	103
Index	105

What's New in SAS Workflow Studio 1.2

Overview

The changes and enhancements for SAS Workflow Studio 1.2 include the following:

- the ability to use tags to group or filter process templates
- a new dialog box that integrates SAS Workflow Studio with existing metadata
- usability enhancements such as the ability to visually map data objects to SAS variables when defining certain policies

Tagging Processes

SAS Workflow Studio now enables you to associate tags (labels) with a process. These tags enable you to group together related process templates and to use these tags to filter your searches for process templates. See [“Using Tags to Categorize and Filter Process Templates”](#) on page 51.

Integration with Metadata

SAS Workflow Studio is now more tightly integrated with existing metadata. When you are defining a process, a new dialog box enables you to retrieve and select participant names, groups, and roles directly from metadata that has already been defined on the server instead of re-entering the information.

Usability Enhancements

Version 1.2 introduces many usability enhancements. Highlights include the following:

- In the Invoke Web Service and Invoke SAS Code dialog boxes, you can now visually map data objects to SAS variables instead of manually entering all of this information. Also, if you are using WSDL 1.1, then SAS Workflow Studio can automatically read the schema and provide a selection list of Web service actions.

- Cut or copy and paste functionality has been enhanced. For example, your formatting is now automatically copied when you copy an element. Where possible, you can select multiple elements for a cut or copy and paste operation.
- There is now an option to automatically close a process template file when you check the file in. You can set this option by selecting **File** ⇒ **Options**.
- There are now default actions associated with elements in the process tree. For example, double clicking on an element opens that element's Edit dialog box.
- There are several enhancements related to 508 accessibility, such as shortcut keys and function keys for navigating the SAS Workflow Studio interface.
- You can now dock and undock the clip explorer. Also, the F11 key now toggles the clip explorer on and off.

Accessibility Features in SAS Workflow Studio 1.2

Accessibility Standards and SAS Workflow Studio

SAS Workflow Studio includes features that improve usability of the product for users with disabilities. These features are related to accessibility standards for electronic information technology that were adopted by the U.S. Government under Section 508 of the U.S. Rehabilitation Act of 1973, as amended. SAS Workflow Studio supports Section 508 standards except as noted in the following table.

If you have questions or concerns about the accessibility of SAS products, send e-mail to accessibility@sas.com.

Accessibility Exceptions

Accessibility Issue	Support Status	Explanation
Product functions shall be executable from the keyboard (keyboard equivalents).	Supported with exceptions	Users cannot control mouse cursor movement, resize objects, or connect diagram elements using keyboard equivalents.
Sufficient information about all user interface elements, including the identity, operation, and state of the element shall be available to Assistive Technology. When an image represents a program element, the information conveyed by the image must also be available as text.	SAS plans to address this issue in a future release.	Screen readers are unable to obtain information about elements in the application window. Labels are not provided for all icons.
Applications shall not override user-selected contrast and color selections and other individual display attributes.	Supported with exceptions	High contrast themes with larger font settings do not inherit the user setting for background color. Font settings are inherited for the menu bar but not for controls in the main application.
Applications shall not disrupt or disable activated features of other products that are identified as accessibility features.	Supported with exceptions	Not all user settings are inherited when applied from a theme.

Enabling Assistive Technologies

For assistive technologies based on Windows to interoperate with SAS desktop applications that are based on Java, you must download the Java Access Bridge (JAB) from [Oracle Corporation](#). See the SAS Web site at <http://www.sas.com/industry/government/java.html> for more information.

Chapter 1

Introduction to SAS Workflow Studio

What Is Business Process Management?	1
Why Use SAS Workflow Studio?	2

What Is Business Process Management?

Organizations of all sizes are constantly forced to deal with the changing business environment. The changing marketplace, technological advances, and shifting customer priorities create challenges that businesses must overcome every day. All successful organizations need efficient processes to convert their competencies and resources into 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 startup (for example, finding the right forms, locating relevant policies and procedures, and so on) and cleanup (for example, forwarding 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 emphasizes integrating technology into the business process such that the process itself drives the business goals, decoupled from the underlying systems and applications. Specifically, BPM emphasizes how the work is done within an organization, in contrast to what a product does.

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 Use SAS Workflow Studio?

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 Studio 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. SAS Workflow Studio 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	5
Overview of SAS Workflow Studio	6
Workflow Lifecycle	7

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. This architecture provides a platform for more efficient and productive business. The SAS workflow architecture consists of the following components:

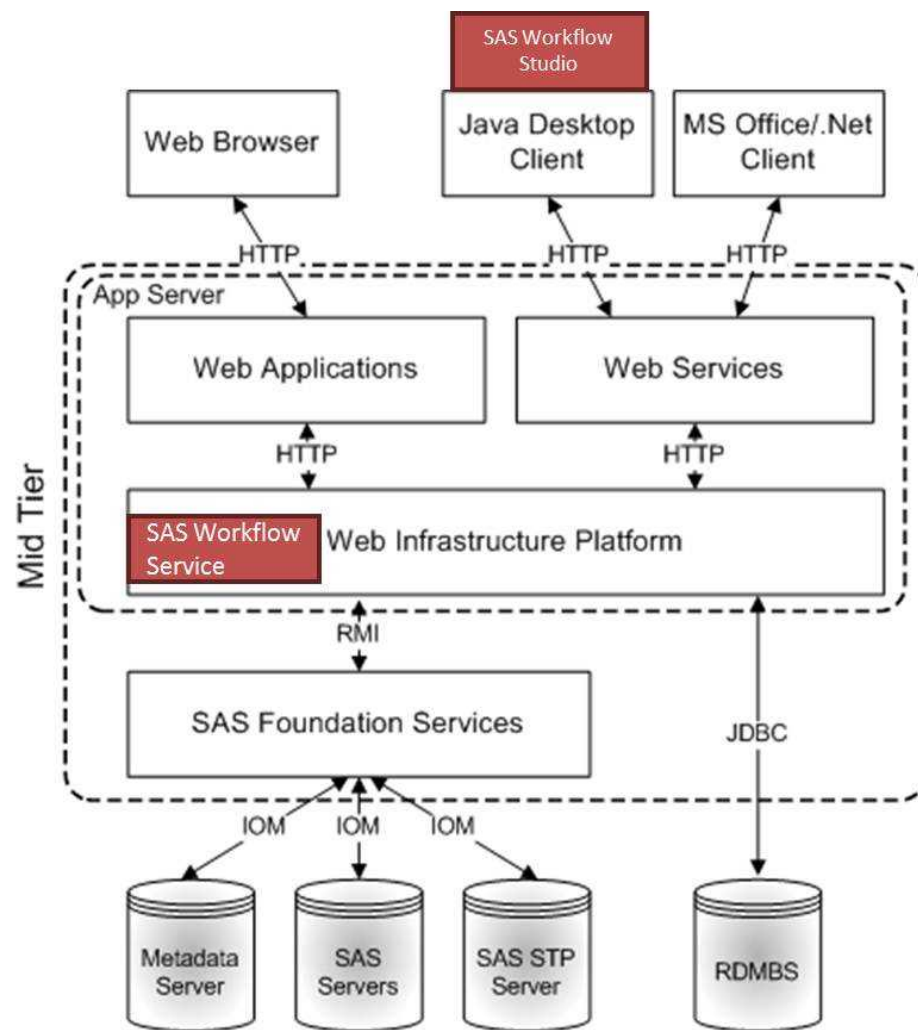
SAS Workflow Services

provides 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

Overview of SAS Workflow Studio

SAS Workflow Studio enables you 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. 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. Process designers can use the point and click interface of SAS Workflow Studio to define processes that simplify the automation of sophisticated business processes.

SAS Workflow Studio also provides predefined and customizable policies. Process designers can use policies to define actions (for example, send an e-mail) that are 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

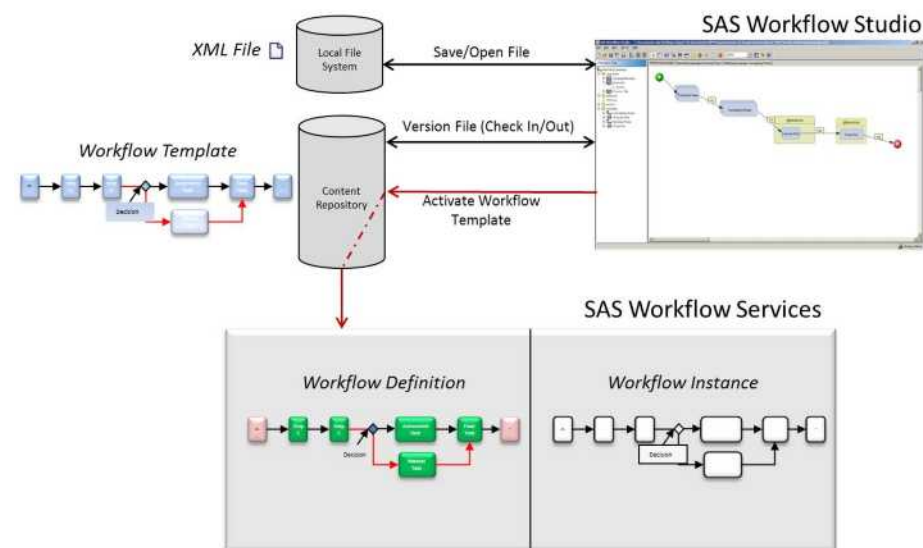
integration points across applications by invoking SAS code or Web services. Policies can also refer to data objects stored in a workflow to add, change, or update peer processes during process execution. For example, you can define a policy for updating a process with the completion date and time of a subprocess.

Process diagrams from SAS Workflow Studio are saved as XML template files. These template files can be uploaded to the server and activated using the SAS Workflow Services. After a process is activated, it is available for instantiation by workflow-enabled applications. Processes can also be versioned using SAS Content Services

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

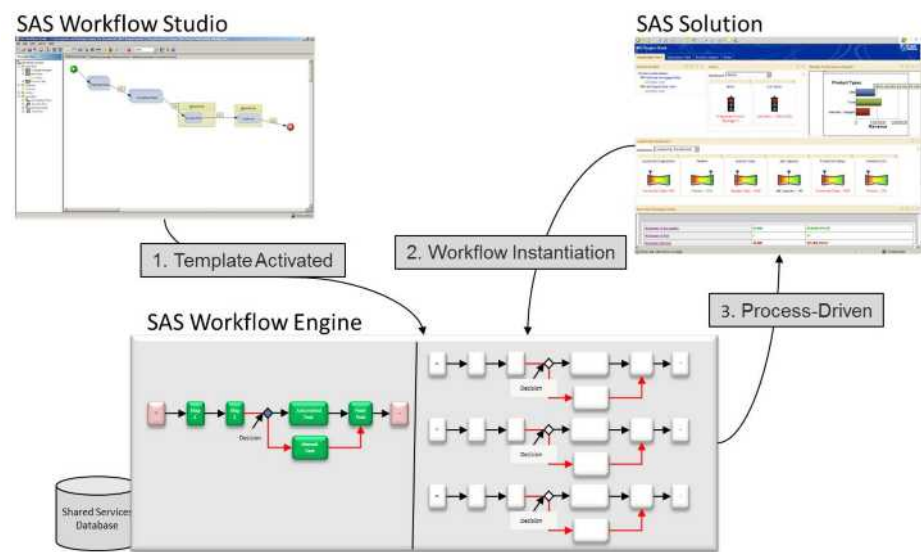


Note: Process changes made in SAS Workflow Studio do not take effect until the new version is uploaded and activated. Once activated, any new processes started use the newly activated version while any currently running process instances continue using their respective version of the workflow definition. For details, see [“Deploying and Maintaining Processes”](#) on page 66.

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 SAS Workflow Services using technologies such as Web Services or published Java APIs to launch workflows, receive workflow status updates, generate notifications, generate alerts, and monitor workflow progress.

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

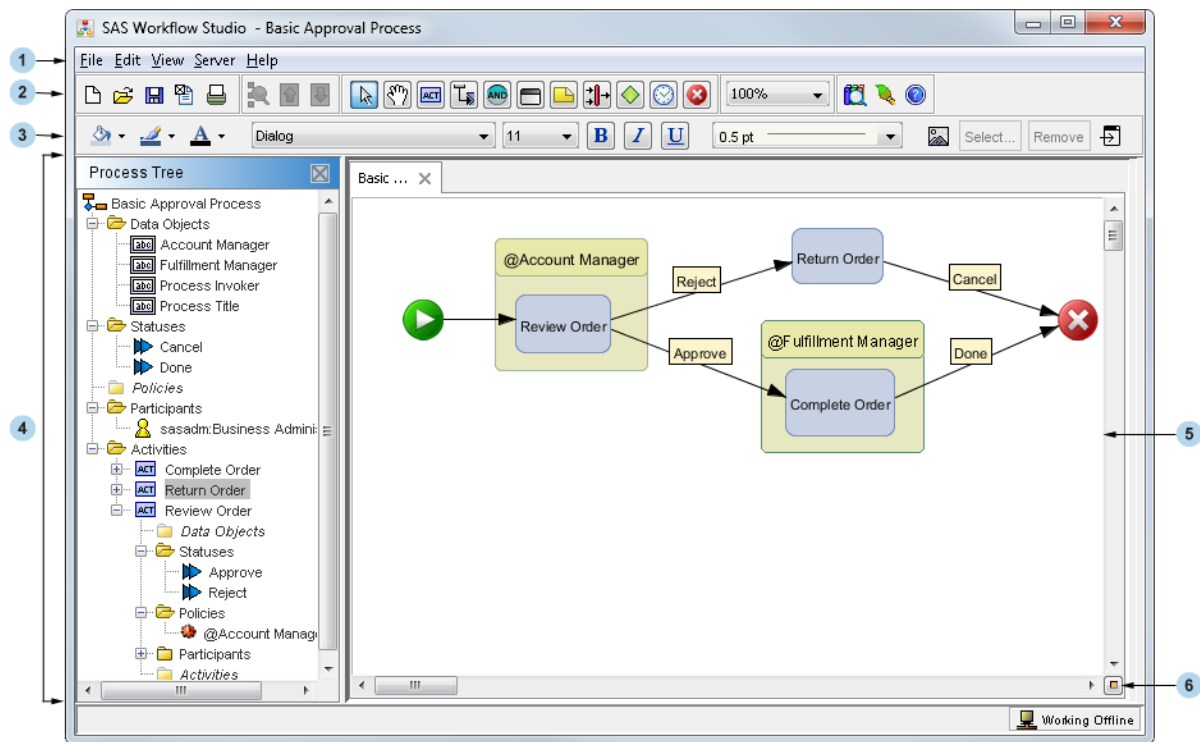
Navigating SAS Workflow Studio

Overview	9
Menus and Toolbars	10
Menus	10
Drawing Tools	12
Zoom Toolbar	13
Format Toolbar	14
Using the Navigator Pane	15
Using the Diagram Editor	16
Elements of a Process	16
Process Diagram Elements	17
Using the Process Tree	18
Using the Clip Explorer	19

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 that is open in the diagram editor in the right pane. Menus and toolbars enable you to build workflow diagrams quickly with minimal effort.

Figure 3.1 SAS Workflow Studio Layout






- 1 Menus on page 10
- 2 Toolbar on page 10
- 3 Format toolbar on page 14
- 4 Process tree on page 18
- 5 Drawing pane
- 6 Navigator pane button on page 15

Menus and Toolbars

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. The process tree is organized by element 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 on 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. 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.

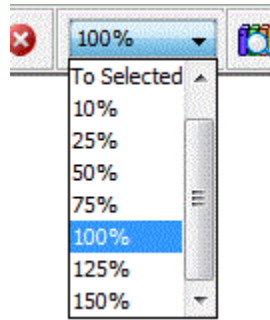
In addition, the following movement key bindings are available:

- 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

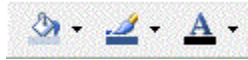
SAS Workflow Studio includes a zoom toolbar that allows users to zoom the current diagram in or out. It consists of a drop-down box with various zooming factors. Choosing a zoom factor of 100% yields a 1:1 magnification ratio. A higher zoom percentage zooms in to the diagram, while a lower zoom percentage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



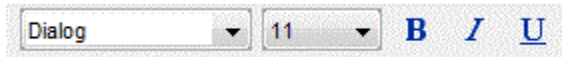
Format Toolbar

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




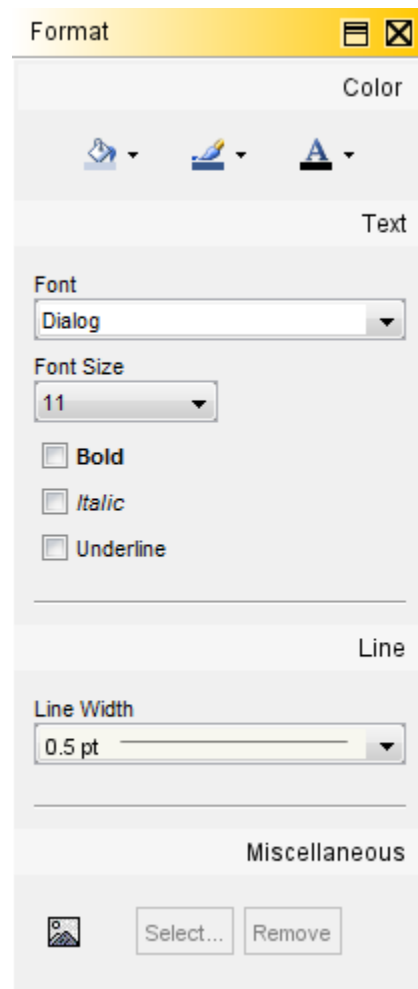
The format toolbar can be toggled on and off by pressing F10 or by selecting the **Format Toolbar** check box in the **View** menu. The format toolbar can also be displayed as a panel by selecting the Switch to panel mode icon () in the Miscellaneous Property Controls group.

Figure 3.2 Format Toolbar in Panel Mode

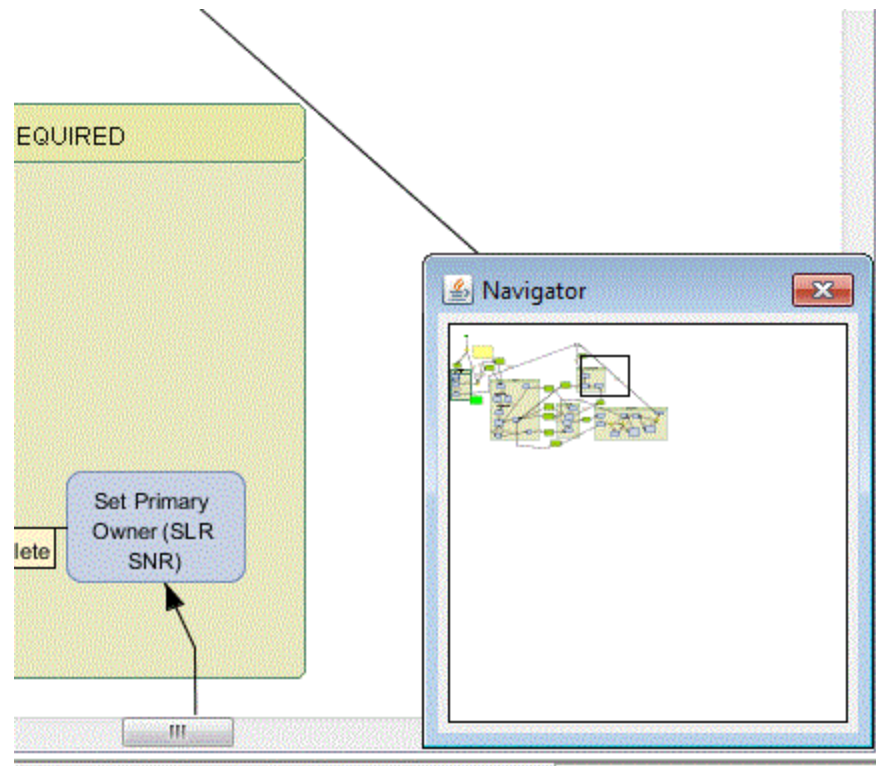
Note: Standard cut and paste operations do not maintain any special formatting. Formatted objects should be reproduced using the Clip Explorer.

Using the Navigator Pane

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

Figure 3.3 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 Diagram Editor



Elements of a Process

Overview

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

- activities
- data objects
- policies
- statuses
- participants

Activities

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

Data Objects

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

Policies

Policies encapsulate event-triggered, executable business logic that can reference process data to add, change, 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. Enabling this option 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:



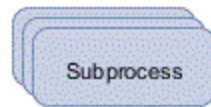
The Start node must precede the first activity in the process.



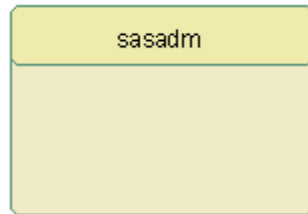
The Stop node must be connected to any activity that leads to process termination.



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



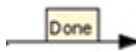
An activity element with a stacked appearance represents a subprocess. A subprocess contains one or more activities that might, in turn, represent subprocesses resulting in a process hierarchy. You can create subprocesses and edit the contained activities from the drawing editor.



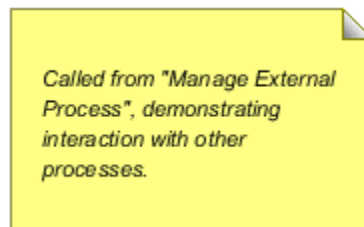
Swimlane elements are used in SAS Workflow Studio to group activities assigned to the same participant definition. They can be explicitly assigned to a Participant object, or they can be implicitly assigned via a swimlane policy. The swimlane policy derives the user, group, or role value defined by the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

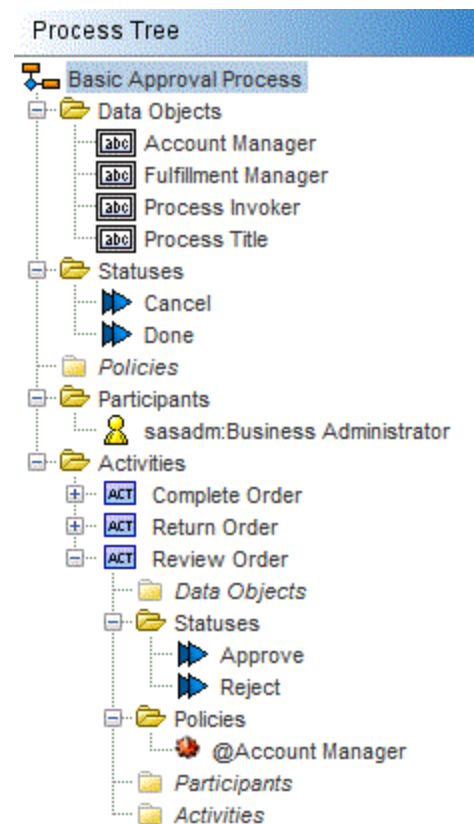


Annotations are used to hold additional information. These notes are for presentation only and are not associated with the run-time process definition.

You can use the drawing tools in the 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 3.4 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.

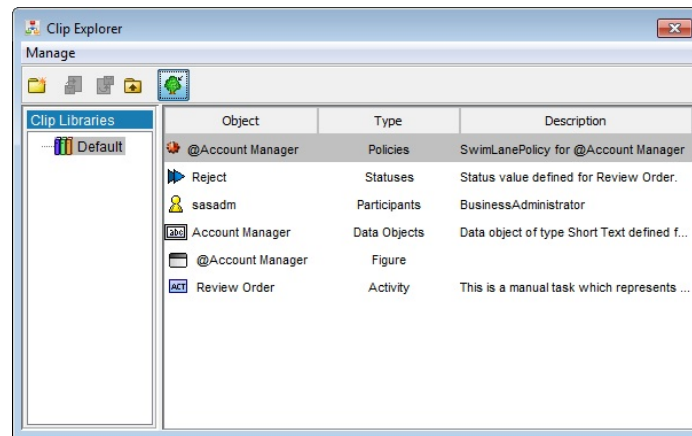
Using the Clip Explorer

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 in 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 you 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 3.5 Clip Explorer

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, select the relevant folder and press the Delete key.

To edit an object, right-click on the object and select **Edit** to open the corresponding property dialog box.

Chapter 4

Defining Processes with SAS Workflow Studio

Creating a Process	22
Working with Activities	22
Adding Activities	22
Connecting Activities in a Sequential Flow	23
Editing Activities	23
Deleting Activities	24
Adding a Subprocess	24
Aligning Activities	25
Working with Statuses	26
Overview	26
Adding Statuses	26
Editing Statuses	27
Deleting a Status	27
Assigning a Status	27
Local Statuses	28
Working with Participants	28
Overview	28
Adding Participants	29
Editing Participants	30
Deleting Participants	30
Assigning Participants	30
Configuring Participants	31
Working with Policies	31
Overview	31
Policy Actions	32
Adding Policies	47
Editing Policies	48
Deleting Policies	48
Assigning Policies	48
Working with Data Objects	49
Overview	49
Adding Data Objects	49
Editing Data Objects	50
Deleting Data Objects	50
Assigning Data Objects	50
Additional Process Features	50
Text Localization	50
Custom Attributes	51


Data Object Substitution	51
Using Tags to Categorize and Filter Process Templates	51

Creating a Process

To create a process, follow these steps:

1. Start SAS Workflow Studio.

By default, SAS Workflow Studio opens a new process template named **Untitled**. This new process template 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 template by selecting **File** ⇒ **New**, by clicking the New File icon () or by pressing CTRL+N.

2. In the process tree, right-click on the root process node (Untitled1) and select **Edit**. The Edit Process dialog box appears.
3. In the **Process Name** field, replace **Untitled1** with the name for the new process.
4. (Optional) Add a description for the process in the **Description** field.
5. Add and configure the required [activities](#), [data objects](#), [statuses](#), [policies](#), and [participants](#).
6. Add [sequence flows](#) to connect the process elements.



Note: Any workflow template changes take effect on the server after 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 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 toolbar, select the Add Activity icon ()
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. See [“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.


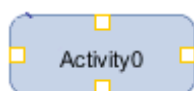
The cursor changes to a large plus sign () and a four-sequence flow anchor blinks when the cursor is in the proper position to make the connection.

Figure 4.1 Flow Anchors for an Activity

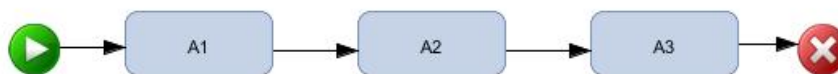


2. In the drawing panel, drag the mouse pointer from the first activity figure toward the second activity figure.

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

For example, a process might contain activities A1, A2, and A3. Activity A1 executes first, followed by A2 upon completion of A1, and so on, until A3 completes to terminate the process. Such a 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, right-click on the activity in either the diagram or the process tree and select the **Edit** option.
2. Enter the desired name in the **Activity Name** field.
3. (Optional) Add a description for the activity in the **Description** field.
4. (Optional) Enable the desired notifications, which are generated using the SAS Notification Service.
5. (Optional) Add localized versions of the name and description.
For more information, see [“Text Localization” on page 50](#).
6. (Optional) Add custom attributes.
For more information, see [“Custom Attributes” on page 51](#).

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

Deleting Activities

To delete an activity, follow these steps:

1. Right-click on the activity in either the drawing panel or process tree and select the **Delete** option.

Alternatively, select the activity 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 among 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 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**.
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.

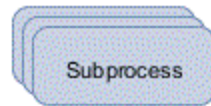
For each option, the relevant diagram opens. For the **Create New** selection, the diagram is blank and untitled. For the other options, the diagram is a copy of the designated existing template.

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

After 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 plus sign () appears on the drawing pane.
2. Drag the cursor (plus sign) vertically or horizontally to form an alignment bar.



You can toggle among the following alignment modes by pressing the space bar:

1. Space evenly
2. Pack tightly
3. Spread out
4. Original

Working with Statuses

Overview

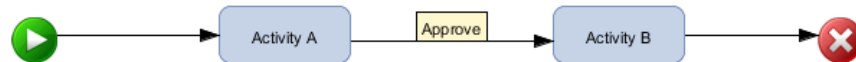
Statuses are used to denote the outcome of a process step as values associated with a logical transition from one activity to another. These values appear as labels on the flow connections and they represent the condition, which must be met in order to realize the transition.

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 after 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 select **New Status**.
Alternatively, right-click on an activity in the diagram editor and select the **New Status** option.
2. In the New Status dialog box, enter a name for the status in the **Status Name** field.
3. (Optional) Add a description for the status in the **Description** field.
4. (Optional) Add localized versions of the name and description.
For more information, see [“Text Localization” on page 50](#).
5. (Optional) Add custom attributes.
For more information, see [“Custom Attributes” on page 51](#).
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.

- Click **OK** to save the new status definition.

Editing Statuses

To edit an existing status definition, follow these steps:

- In the process tree, open the **Statuses** folder and then right-click on the desired status node.
- Select **Edit**.
- Change the desired values in the Edit Status dialog box.
- Select **OK** to save the updated status definition.

Deleting a Status

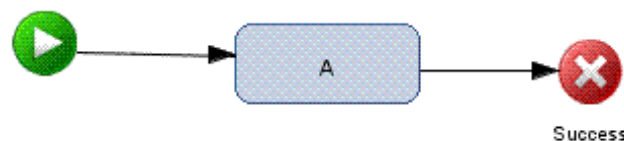
To completely remove a status value from the workflow, follow these steps:


- Right-click on the status in the process tree and then select **Delete**.
Alternatively, you can select the status directly in the process tree and press the Delete key or CTRL+X.
- 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**. This opens a menu 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.

If the termination node is within a local subprocess, transitions with this status are executed.

This means status propagation is triggered in the parent process via the termination node with status.

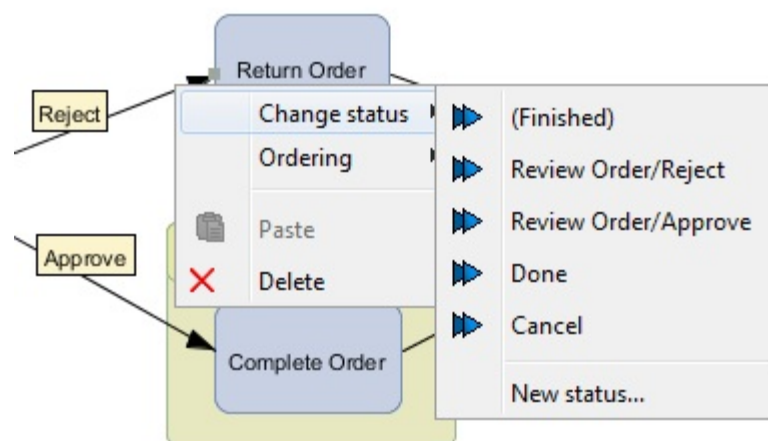
If the subprocess is executed more than once (within a loop, for example), it is recommended that the connection be joined into a single activity before terminating.

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 Studio 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**.

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 Name**, **Group Name**, or **Organizational Role Name** option and enter the desired value.
5. (Optional) Add localized versions of the name.

For more information, see [“Text Localization” on page 50](#).

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**.
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**.
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 Studio provides a way to define these actions through policies. A policy object encapsulates executable business logic. 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: It is strongly recommended that policies do not delete process templates from the repository. Deleting a process template can cause unpredictable results when process information is accessed from SAS Workflow Administrator.

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. However, 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 is 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 that 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. See [“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/webservicesmaker-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	<p>Specifies the name of the directive.</p> <p>The value can be specified using data object substitution. For more information, see “Data Object Substitution” on page 51.</p> <p><i>Note:</i> If you are logged in to the SAS platform, then the ellipsis button () launches a directive picker.</p>
URL	<p>Specifies the target URL to invoke, execute, or access.</p> <p>The value can be specified using data object substitution.</p>
Parameters	<p>Specifies additional HTTP parameters.</p> <p>The value can be specified using data object substitution.</p>
Method	<p>Specifies the HTTP method to execute.</p> <p><i>Note:</i> POST is the only method that is currently supported.</p>
Status Code Data Object	<p>Specifies an optional data object to bind to the return code of the HTTP request.</p>

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. See [Appendix 3, “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.

Parameter Name	Description
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 51 .
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 <i>n</i> Data Object Parameter <i>n</i> Macro Variable	specify up to five optional additional pairs of data object and macro variable values, where <i>n</i> 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. Click 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.

See [Appendix 3, “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. <i>Note:</i> 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.

Parameter Name	Description
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.

See [Appendix 3, “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 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 Studio 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 following form:

MM/dd/yyyy hh:mm:ss AM|PM

An example is: **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.

Note: 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 51 .
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 that is 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 51 .
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 51 .
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’s 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.
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.

Parameter Name	Description
HTTP Parameters	<p>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.</p> <p>Separate the label path values with commas.</p>
Action on Expiry	<p>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 43.</p>
Expiration Date Data Object	<p>Specifies the data object that defines the expiration date of the policy.</p>
Expiry Recipients(s)	<p>Specifies one or more users defined in SAS metadata to whom notifications are sent when the policy reaches its expiration date.</p> <p>Separate multiple user names with commas.</p> <p><i>Note:</i> This parameter is available only when the Action on Expiry parameter is set to Reroute.</p>
Expiry Group Recipients(s)	<p>Specifies one or more groups defined in SAS metadata to which notifications are sent when the policy reaches its expiration date.</p> <p>Separate multiple group names with commas.</p> <p><i>Note:</i> This parameter is available only when the Action on Expiry parameter is set to Reroute.</p>

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 51](#). 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 after 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.
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.

Parameter Name	Description
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 43 .
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. <i>Note:</i> 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. <i>Note:</i> 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 51](#). 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.

Parameter Name	Description
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 new, separate 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.

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 which 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. Click **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**.
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**.
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**.

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. (Optional) Add a description for the data object in the **Description** field.
6. (Optional) Add localized versions of the label and description.
For more information, see [“Text Localization” on page 50](#).
7. (Optional) Add custom attributes.

For more information, see [“Custom Attributes” on page 51](#).

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**.
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**.

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 Studio 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
```

Using Tags to Categorize and Filter Process Templates

What Is a Tag?

Tags are alphanumeric labels that are associated with template files. Tags enable you to categorize and group related templates on a user-defined basis. For example, you can define a tag for a process template based on the application or organization that use that process. You can use these tags to filter the list of templates in the Manage Processes dialog box. (See [“Process Management” on page 69](#).)

- Tags are static labels. Existing tags can be modified using SAS Workflow Studio only. Tags have no effect on active process instances and cannot be modified at run time.
- Tags are not used for access control. They do not prevent the sharing of process templates among different applications. They do not affect whether or how a process is activated.
- Tags are defined at the server level and can be assigned to any template for that system. If the template is moved to another system, the user is prompted to create the relevant tags on the new system before uploading and activating the template.
- Tags cannot be renamed or deleted, but they can be disassociated from a template.

You must be logged in to the SAS server before you can work with tags.

Managing Tags through the Process Properties Dialog Box

Through the Process Properties dialog box, you can manage tags both for templates that have been saved locally and for templates that have been saved in the content repository. However, you can manage tags for only one process at a time.

To manage tags for a specific process through the Process Properties dialog box, follow these steps:

1. Log on to the server if you have not already done so. See [“Logging On to the Server” on page 67](#).
2. Open the template to which you want to add a tag. If the template has been saved (uploaded) to the repository, then you must check out the template first. See [“Process Management” on page 69](#).
3. Open the Process Properties dialog box in one of the following ways:
 - Select **File** ⇒ **Properties**.
 - Right-click on the root node of a process in the process tree and select **Edit**.
 - Select the root node in the process tree, and select **Edit** ⇒ **Edit properties**.
 - Double click on the root node in the process tree.

You can add new tags and manage existing tags associated with the template in the **Tags** section of the dialog box.

- To add a new tag, click **Add**, enter the new tag, and click **OK**. SAS Workflow Studio adds the tag to the list of available tags.
 - To associate a tag with the current template, select the check box next to that tag. You can associate several tags with one template.
 - To disassociate a tag from the template, clear the check box next to that tag.
4. Click **OK** to save your changes and to close the Process Properties dialog box.

Managing Tags through the Manage Tags Dialog Box

The Manage Tags dialog box enables you to manage tags for several templates at once. However, you can manage tags only for templates that have been saved to the content repository.

To manage tags through the Manage Tags dialog box, follow these steps:

1. Log on to the SAS server if you have not already done so. See [“Logging On to the Server” on page 67](#).
2. Select **Server** ⇒ **Manage Processes**.

3. Select the check box next to the templates that you want to modify.
4. Select **Manage Tags**.
 - To add a new tag, enter the tag name in the **New Tags** field, and click **Add**. SAS Workflow Studio adds the tag to the list of available tags.
 - To associate a tag with the selected template, select the check box next to that tag.
 - To disassociate a tag from the template, clear the check box next to that tag.
 - If there is a small blue square in the check box for a tag, it means that the tag is associated with some of the selected templates but not others. If you select the check box, then the tag will be associated with all of the selected templates. If you clear the check box, then the tag will be disassociated from all of the selected templates.
5. Click **OK** to save your changes to the repository and to close the Manage Tags dialog box.
6. Click **Close** to close the Manage Processes dialog box.

Chapter 5

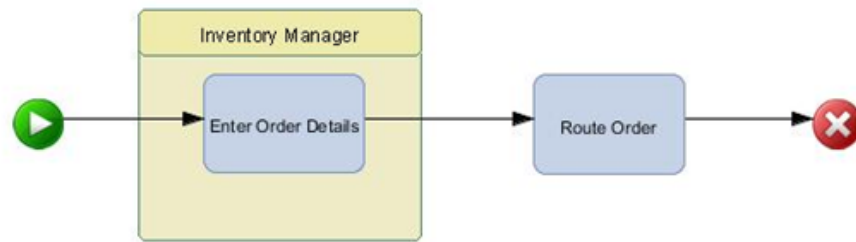
Advanced Topics

Workflow Patterns	55
Overview	55
Defining Alternate Paths	56
Defining Parallel Paths	60
Defining Convergent Paths	61
Additional Information on Workflow Patterns	63
Using Timers	64
Overview	64
Adding Timers	64
Timer Considerations	64
Advanced Timer Parameters	65
Deploying and Maintaining Processes	66
Overview	66
Saving a Workflow Template	66
Restoring a Workflow Template	67
Logging On to the Server	67
Versioning Process Definitions	68
Process Management	69
SAS Alert Notification Templates	70
Editing Alert Notification Templates	70
Adding Alert Notification Templates to the WebDAV Repository	71
Adding Fields to Alert Notification Templates	71
Configuring the Subject for an Alert Notification	72

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

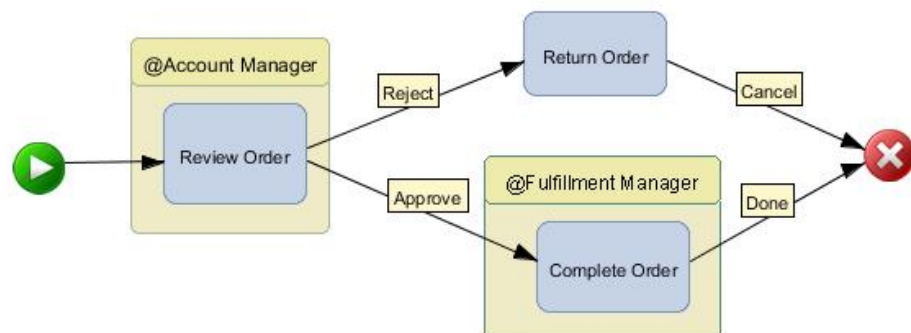
Overview

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

You can define a status for each path 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 Studio also supports multiple paths for the same status value (Workflow Pattern: Parallel). Using multiple paths is similar to using a merge/fork gateway, which might not have status assignments.


See [Appendix 5, “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 Studio 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**.
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. (Optional) 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 comply with the following rules:

- The expression must evaluate to true or false.
- The expression can use comparison operators (<, =, ==, or >).
Note: The = and == operators are evaluated in an identical manner.
- The expression can contain logical combinations (&&, |, !) of some set of comparisons.
- The expression 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.

See [Appendix 2, “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 that 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 on the connection, select **Change status**, and select the appropriate expression.

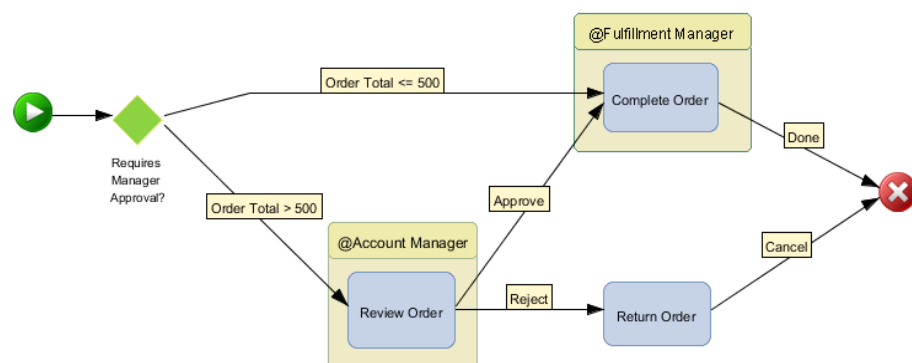
Each expression corresponds to a calculated value or outcome and is represented as a label in a similar manner to status values. In addition to these user-defined calculations, the special status of **Otherwise** can be assigned. Use this value to designate the logical path that should be traversed when 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

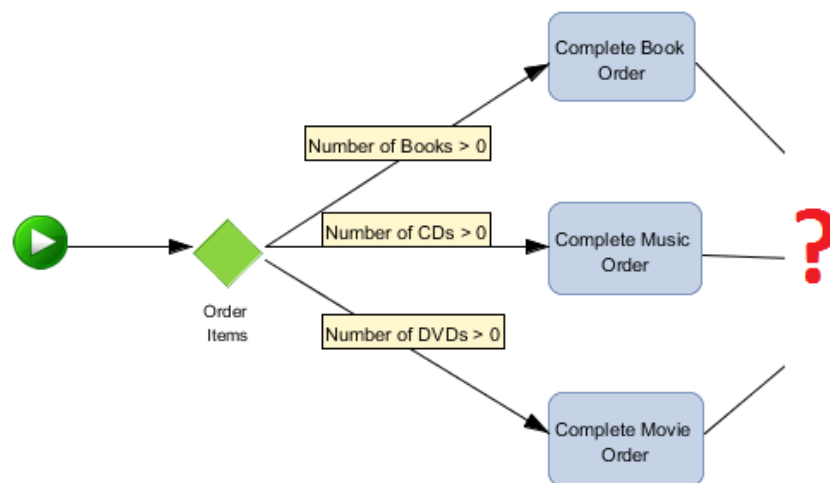
Figure 5.3 Example of Path Selection Based on Expression (Exclusive Choice)



See 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:

- There is a 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 Studio 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 preceding 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

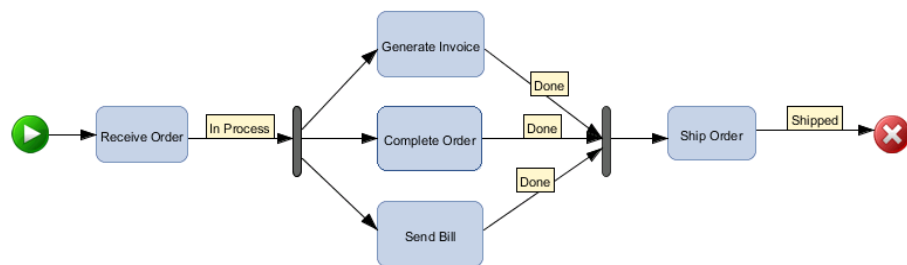
To add a merge/fork gateway, follow these steps:

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 before 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). The activities start only after the Receive Order activity has completed.

Figure 5.6 Example of Parallel Processes



See [Appendix 5, “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**.

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 after the current activity preceding it is completed.
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 (.
- Configure the inputs to flow into a single task directly (no gateway).

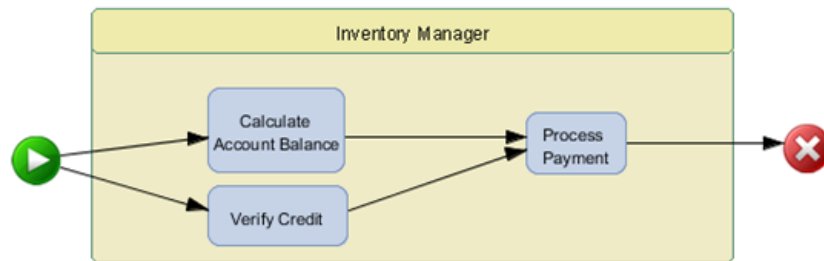
SAS Workflow Studio currently supports logical XOR (exclusive) for the first option and logical OR (inclusive) for the second option. So, for a single execution of the convergent path, the logical OR gateway should be used. If multiple executions are desired (one for each path), then the paths should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

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

1. Select the Logic Gateway icon (.
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** 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.

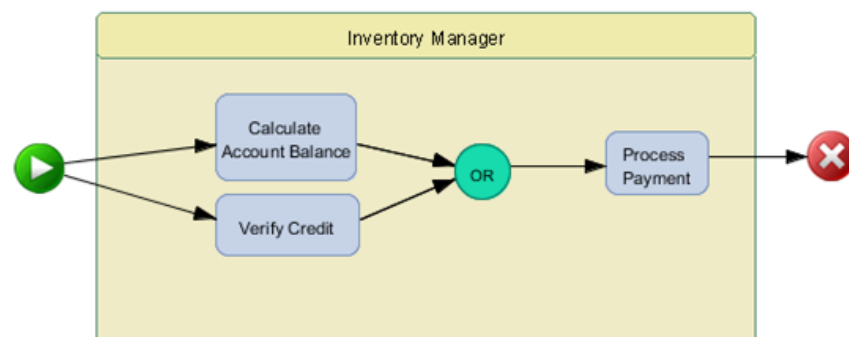
The following example results in only a single work item (task instance) for Process Payment after both Calculate Account Balance and Verify Credit have been completed. However, any actions associated with the Process Payment activity are executed twice.

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.

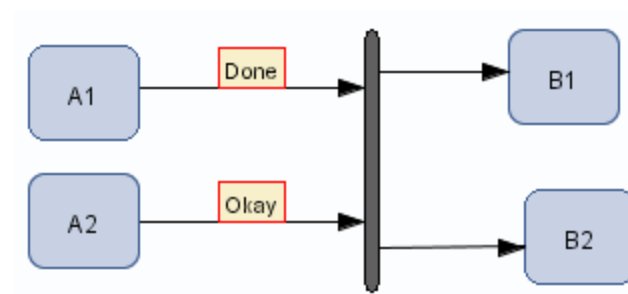
Figure 5.8 Example of Exclusive Merge (XOR)



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 the use of a merge/fork gateway where the Ship Order task is not executed until all three of the preceding tasks (Generate Invoice, Complete Order, and Send Bill) complete. If more than one activity exists after the gateway, then the activities are started together and run independently. The following example shows 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, three inputs might lead to the converged path, but there might be only two trigger actions. 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)

Additional Information on Workflow Patterns

For more information on workflow patterns, see the [Workflow Patterns Initiative](http://www.workflowpatterns.com) Web site. The following documents are of particular interest:

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>.

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 () , 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

Accessing Advanced Timer Properties

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-type 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.
+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.

See [Appendix 4, “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. See [“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. After a new process is uploaded and 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 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 ().

2. Navigate to the desired directory in a dialog box.
3. Enter the 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. After you have logged on to the server, 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, see *SAS Intelligence Platform: Web Application Administration Guide*.

Versioning Process Definitions

Who Can Activate 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** 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 66](#) for details.

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**.

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.

To access these functions, follow these steps:

1. Select **Server** ⇒ **Manage Processes**.

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 is 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 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.

Note: It is strongly recommended that process templates are not deleted from the repository. Deleting a process template can cause unpredictable results when process information is accessed from SAS Workflow Administrator.

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**.
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**.

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 add the \$ character as a prefix or suffix to these fields 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**.

The SAS Application Infrastructure Properties dialog box appears.

3. Click 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.

See [Appendix 3, “Policy Usage Examples,” on page 79](#) for a detailed example that leverages the SAS Template Service.

Appendix 1

Workflow Loader Utility

You can 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 Windows 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>
```

Parameters:

nnn.nnn.nnn.nnn

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

nnnn

is the multicast port number of the SAS middle tier.

user-ID

is the ID of a user who is authorized to load content. In SAS 9.3, any user that is 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 the modified XML is written if any data types are fixed. The utility automatically fixes any invalid data types by mapping them to appropriate supported types for SAS 9.3 (for example, Short Text). A new file with the corrected data type is created in the directory specified, and the new file is automatically loaded into the content repository.

-replace

replaces the process definitions if they already exist in the engine. In the SAS Content Server, this option updates the version.

-activate

activates or loads the template into the engine after the template is loaded into the repository.

Appendix 2

Decision Expression Examples

Operator Support in Decision Expressions	75
Function Support in Decision Expressions	77
Decision Expression Examples	77
Example Data Objects	77
Example 1: Comparing the Value of a Data Object to a String Constant	78
Example 2: Comparing the Values of Multiple Data Objects to String Constants .	78
Example 3: Comparing the Value of a Data Object to a Numeric Constant	78
Example 4: Comparing the Values of Two Different Data Objects	78
Example 5: Expression Using the Uppcase Function	78
Example 6: Expression Using Data Object Substitution	78

Operator Support in Decision Expressions

The following tables describe the operators that you can specify in expressions in decision gateways.

Table A2.1 Comparison Operators

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

Table A2.2 Arithmetic Operators

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

Table A2.3 Conditional Operators

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

Table A2.4 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 A2.5 Unary Operators

Operator	Operation
+	unary plus
-	unary minus
++	increment value by 1

Operator	Operation
--	decrement value by 1
!, NOT, not	logical NOT

Function Support in Decision Expressions

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

Table A2.6 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. <i>Note:</i> This function is supported only by Date data objects.
datepart	Uses only the date part of the data object value when evaluating. <i>Note:</i> This function is supported only by Date data objects.

Decision Expression Examples

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 Uppcase Function

```
uppercase(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. (See [“Data Object Substitution” on page 51](#).) This is useful if a local data object has the same name as a global data object that has been 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 (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 3

Policy Usage Examples

HTTP Request Policy Example	79
Invoke SAS Code Policy Example	80
Invoke Web Service Policy Example	82
Send Notification Using SAS Template Service Policy Example	85

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 A3.1 HTTP Request Policy Parameters

Policy Parameter	Value
Event	Process Started
Action	HTTP Request
Description	Execute an HTTP Request
Directive	GRC_Workflow
URL	
Parameters	<code>Command=setFields&tpHandle=\${../../GRC_HANDLE}</code>
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.

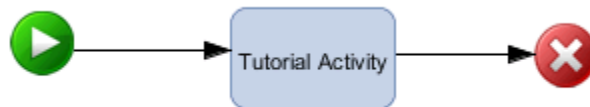
result

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.

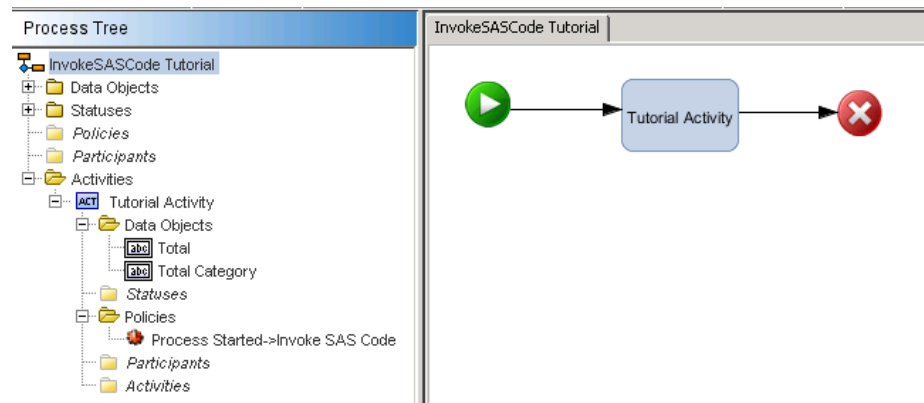
Return Data Object

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. Because this example only requires one, the following line of code is prepended to the SAS code prior to execution:

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

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

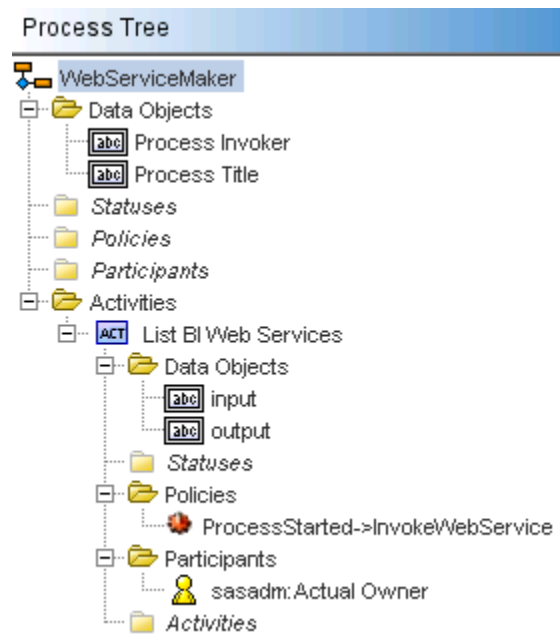
Invoke Web Service Policy Example

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:



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

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

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"/>
      </sequence>
    </complexType>
  </element>
</schema>

```

```

        <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	<i>password</i>
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>
```

See 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 notifications via the SAS Alert Notification Service to participants 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: Five expiry options are available: None, Remove, Reroute, Resend, and Start Workflow. For more information, see [“Expiry Options for Notification Policies” on page 43](#).

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: Five expiry options are available: None, Remove, Reroute, Resend, and Start Workflow. For more information, see [“Expiry Options for Notification Policies” on page 43](#).

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)	<code>\${../Account Manager}</code>
Description	
Template	<code>SAS_Email_Message</code>
Directive	

Policy Parameter	Value
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 51](#).

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.

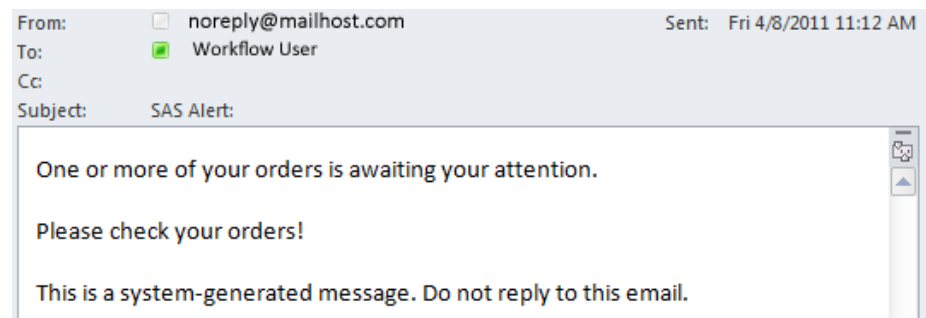
File A3.1 SAS_Email_Message Template

```

$if(_SAS_TEMPLATE_TXT)$
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:



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

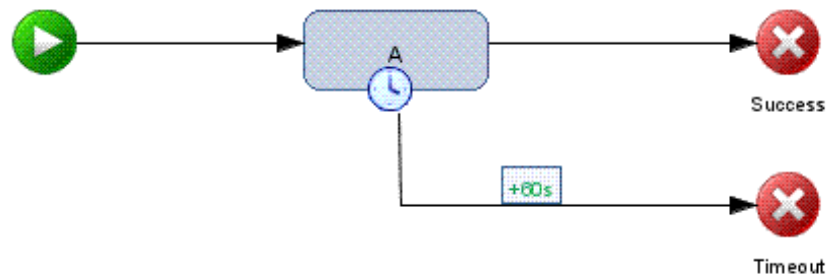
Appendix 4

Timer Examples

Timer on Border of Activity	89
Timer Inside Activity Border	89
Automatically Traversing a Transition	90
Timer Expression Examples	90
Timer Expired and Tool Policy Example	92

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**.

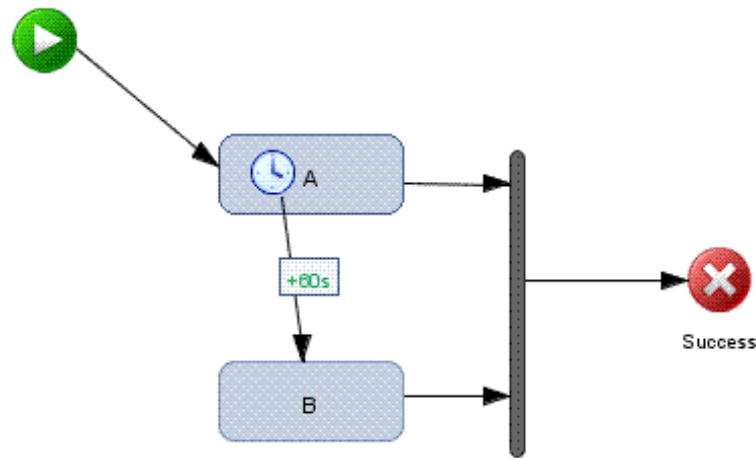


If Activity A is completed before the timer expires, then the timer is stopped, and the transition to the Stop node is traversed. This causes the instance to be stopped with a status of **Success**.

Timer Inside Activity Border

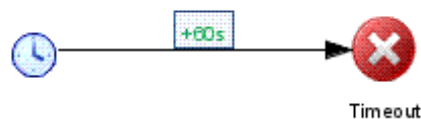
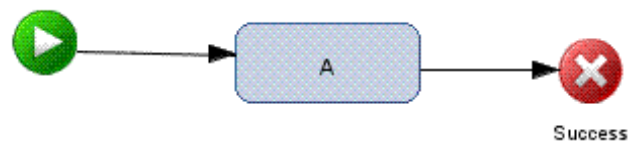
Activity A triggers Activity B 60 seconds after Activity A starts. Because the Timer is inside the border of Activity A, Activity A continues to be active after the timer is

triggered. Therefore, both Activity A and Activity B are active after the timer has fired (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 A 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 one day.

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

Note: If the Recurrence: Number of Repetitions value is left blank or equals -1, then there are unlimited repetitions until the containing activity or process ends.

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. See the [org.quartz.CronExpression](#) class 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 one 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

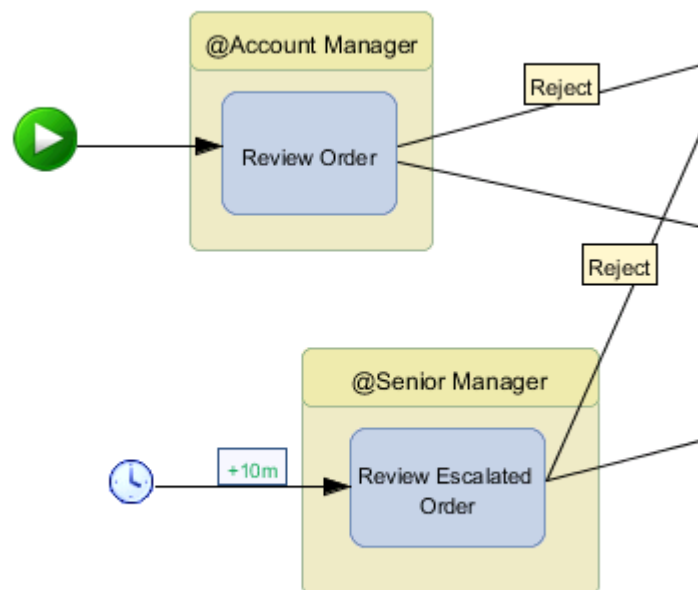
The following example is similar to the previous one, except that a Timer End value is specified. The Timer End value indicates when the timer should terminate. In this example, the timer will repeat every 24 hours and will cease after 1 week (as specified the value **+1w** for the Timer End).

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.

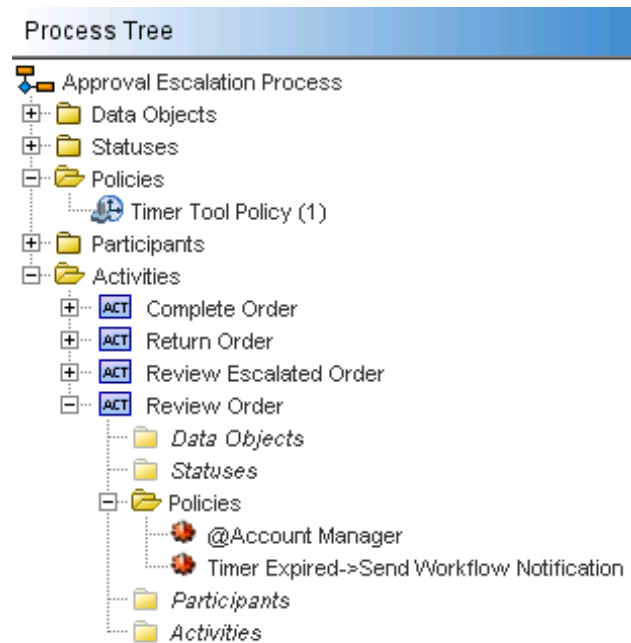
Figure A4.1 Timer Expired Process Diagram



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

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

Figure A4.2 Timer Expired Process Tree



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 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.

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

Appendix 5

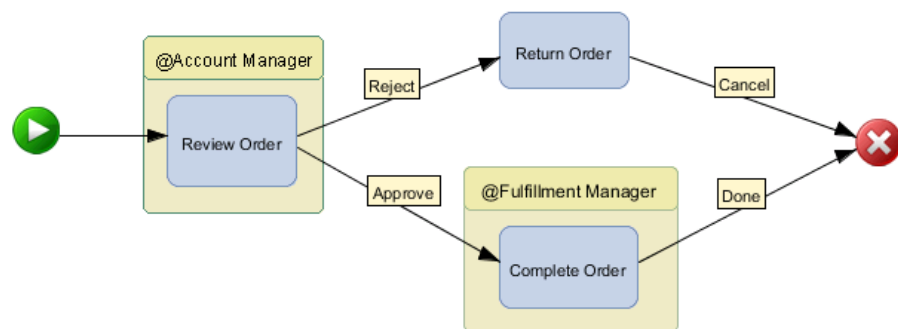
Basic Workflow Examples

Basic Approval Process	95
Order Fulfillment Process	98

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.

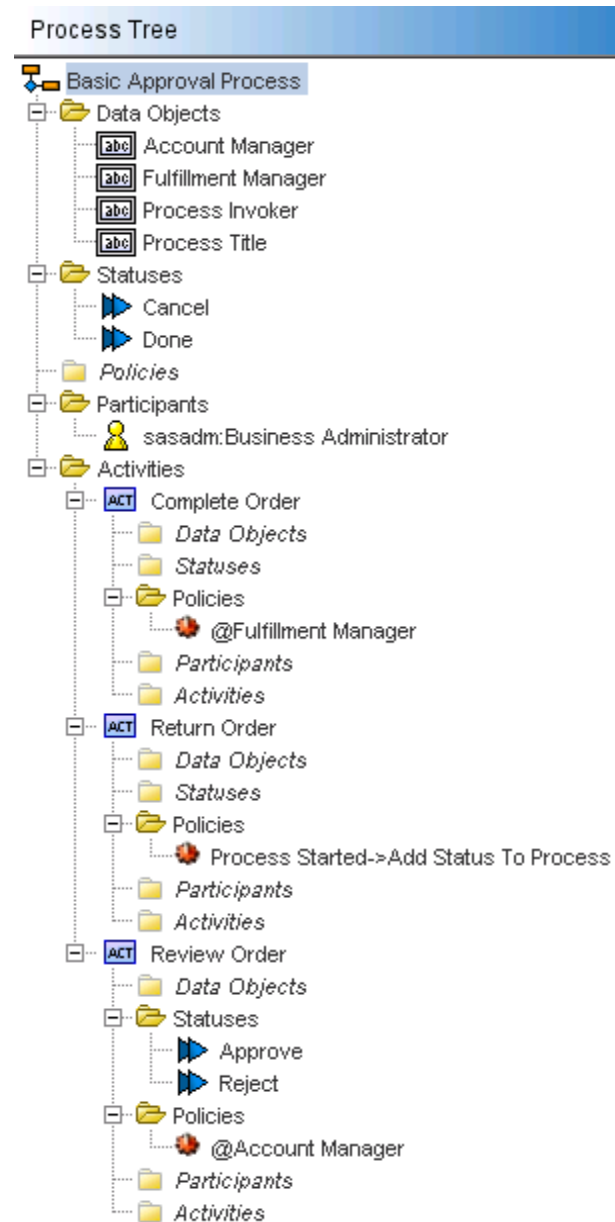
Figure A5.1 Basic Approval Process Diagram



This process uses the Exclusive Choice workflow pattern.

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

Figure A5.2 Basic Approval Process Tree



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

Table A5.1 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.*
Process Title	Data Object	Default data object of type Short Text defined for all processes. Holds the title for a process instance.*

Element Name	Element Type	Details
Account Manager	Data Object	<p>Data object of type Short Text defined for all processes. Holds the group name used by the @Account Manager swimlane policy.*</p> <p>The default value is US Region Accountants.</p>
Fulfillment Manager	Data Object	<p>Data object of type Short Text defined for all processes. Holds the group name used by the @Fulfillment Manager swimlane policy.*</p> <p>The default value is US Region Supply Managers.</p>
Cancel	Status	Status value defined for all processes.**
Done	Status	Status value defined for all processes.**
sasadm	Participant	<p>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.</p> <p>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.</p>
Review Order	Task (Atomic Activity)	<p>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.</p>
@Account Manager	Policy/Swimlane	<p>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.</p> <p>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.</p> <p>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.***</p>
Approve	Status	Status value defined for Review Order.†
Reject	Status	Status value defined for Review Order.†

Element Name	Element Type	Details
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.
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	<p>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.</p> <p>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.</p> <p>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.***</p>

* 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. However, the value 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. However, the value 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.

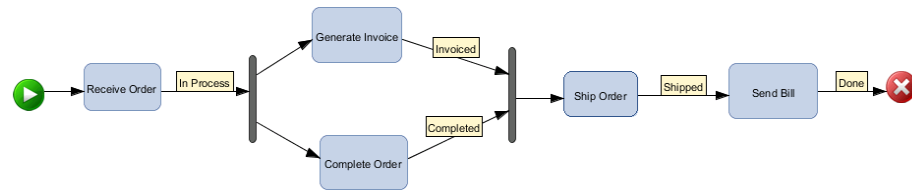
† This status value is defined at the activity level and is set as visible within the entire subtree by default. This setting allows access to the value by decisions and policies at run time by all contained activities and subprocesses. However, the value is not accessible as a local copy of the status.

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. When both of these tasks are completed, the fulfillment manager ships the order. Finally, after the order has shipped, the billing associate sends the bill.

Figure A5.3 Order Fulfillment Process Diagram

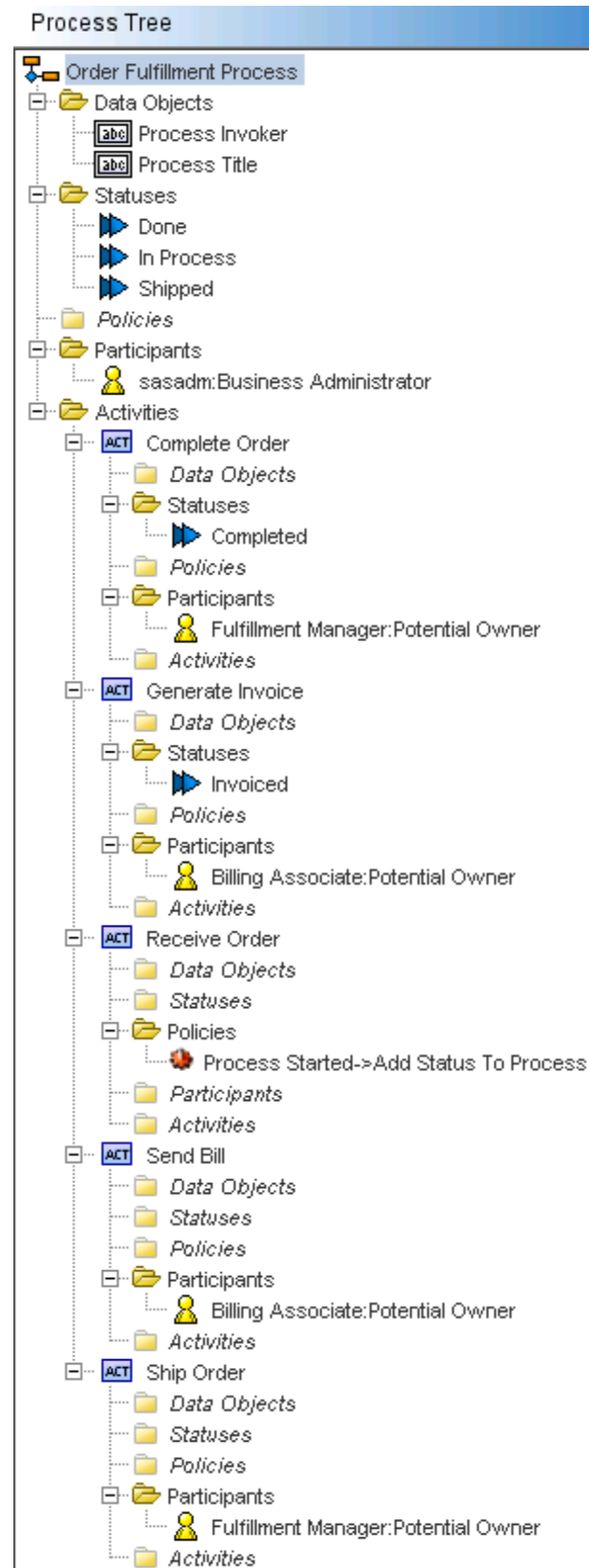


This process uses the following workflow patterns:

- Parallel
- Join (Merge with Synchronization)

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

Figure A5.4 Order Fulfillment Process Tree



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

Table A5.2 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	<p>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.</p> <p>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.</p>
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.
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 and is completed by setting the status to Invoiced.
Billing Associate	Participant	<p>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.</p> <p>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.***</p>
Invoiced	Status	Status value defined for Generate Invoice.†

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 after the order is received. This task represents order completion by the fulfillment manager and is completed by setting the status to Completed.
Fulfillment Manager	Participant	<p>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.</p> <p>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.***</p>
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 and 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 and 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. However, the value 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. However, the value 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. However, the value is not accessible as a local copy of the status.

Glossary

activity

a process element that associates executable logic with an event such as a status change or timer event.

activity status

the outcome of an activity in a process, for example, Started, Canceled, or Accepted. The status of an activity is typically used to trigger the next activity.

annotation

a note that is added to a process to provide supporting information about a process or element within the process. These notes are for presentation only and are not associated with the run-time process instance.

data object

an object that holds the business data required to execute workflow process activities.

decision gateway

a type of gateway that controls which sequence flow is executed by evaluating the expression associated with the gateway. Expressions are evaluated at run time. See also gateway.

gateway

a process diagram element that controls the execution of sequence flows through a process. Types of gateways include logic gateways, merge/fork gateways, and decision gateways.

instance

See process instance

logic gateway

a type of gateway that controls which sequence flow is executed by using AND or OR logic. See also gateway.

merge/fork gateway

a type of gateway that enables multiple tasks or sequence flows to execute in parallel. A merge/fork gateway allows multiple sequence flows to converge and individual sequence flows to diverge. All paths from a merge/fork gateway are executed. See also gateway.

participant

a user, group, or role that is assigned to an activity. These users, groups, and roles are defined in SAS metadata and are mapped to standard roles for the process.

policy

a process element that associates executable logic with an event such as a status change or timer event.

process

a series of activities, together with the participants and the logic required to execute the activities. A workflow process includes policies, status values, and data objects.

process definition

a process template that has been uploaded to the server and activated. Process definitions are used by the SAS Workflow Engine to create new workflow process instances.

process instance

a process that is running in the SAS Workflow Engine. After a process template is uploaded to the server and activated, client applications can use the template to create and run a new copy of the process definition. Each new copy is a process instance.

process template

a model of a process that has been saved to an XML file.

sequence flow

a connection, typically represented by an arrow, between process diagram elements. Sequence flows indicate the order in which activities are executed.

swimlane

a process diagram element that enables you to group activities that are assigned to the same participant.

workflow

a sequence of processes, together with the people, the tools, and the information necessary to accomplish a specific business objective or goal.

Index

A

- activities
 - adding 22
 - aligning vertically or horizontally 25
 - alignment modes 25
 - connecting 23
 - deleting 24
 - editing 23
 - in the diagram editor 16
 - localizing text for 50
- actual owner participant 29
- Add Status to Process policy
 - parameters 32
- Add Status to Process with ID policy
 - parameters 32
- alignment modes 25
- arithmetic operators 76

B

- bitwise and bit shift operators 76
- business administrator participant 29
- business process management, description 1

C

- clip explorer 19
- comparison operators 75
- conditional operators 76
- Copy Data Object from External Process
 - policy
 - parameters 34
- Copy Data Object policy
 - parameters 33
- Copy Data Object to External Process
 - policy
 - parameters 33
- Copy Participants to Process policy
 - parameters 32
- copying elements

- maintaining object properties 19
- using the clip explorer 19

D

- data object
 - adding 49
 - assigning to an activity 50
 - copy from external process, policy for 34
 - copying between data objects, policy for 33
 - copying to external process, policy for 33
 - deleting 50
 - description of 49
 - editing 50
 - Extract from XML Data Object policy 34
 - functions for use with 77
 - in expressions, examples 77
 - Increment Data Object policy 35
 - localizing text for 50
 - Send Notification by Data Object policy 40
 - substitution, example 78
 - types of 49
- Data Object Updated event 31
- decision expression, examples 77
- decision gateway, adding 57
- decision operators
 - See operators
- diagram editor 16
- diagram elements, description 17

E

- e-mail
 - Send E-mail policy 40
 - Send E-mail policy, example 85
- events

- Data Object Updated 31
- initiating automatically 31
- Participants Updated 32
- Process Finished 31
- Process Started 31
- Status Addition 31
- Status Removal 31
- Timer Expired 31
- excluded owner participant 29
- exclusive choice
 - example 95
 - workflow pattern 56
- expiry options for notification policies 43
- expression examples, decision 77
- Extract from XML Data Object policy
 - parameters 34

F

- functions, in decision expressions 77

H

- HTTP Request policy
 - example 79
 - parameters 35

I

- Increment Data Object policy
 - parameters 35
- Invoke SAS Code policy
 - example 80
 - parameters 36
- Invoke Web Service policy
 - example 82
 - parameters 37

L

- localization 50

N

- notification participant 29
- notifications
 - expiry options for policies 43
 - Send Notification by Data Object policy 40
 - Send Notification by Workflow Role policy 41
 - Send Workflow Group Notification policy 41
 - Send Workflow Notification policy 44
 - sending, example 85
- Notify Participant policy

- parameters 38

O

- operators
 - arithmetic 76
 - bitwise and bit shift 76
 - comparison 75
 - conditional 76
 - unary 76

P

- parallel, workflow pattern 56
- parameters, policy 32
- participant 28
 - adding 29
 - assigning to activities 30
 - configuring 31
 - copying between processes, policy for 32
 - deleting 30
 - editing 30
 - Notify Participant policy 38
 - roles 28
 - Set Multiple Process Participants policy 46
 - Set Process Participant policy 45
- Participants Updated event 32
- policy
 - actions 32
 - Add Status to Process 32
 - Add Status to Process with ID 32
 - adding 47
 - assigning to an activity 48
 - Copy Data Object 33
 - Copy Data Object from External Process 34
 - Copy Data Object to External Process 33
 - Copy Participants to Process 32
 - deleting 48
 - description of 31
 - editing 48
 - events that initiate automatically 31
 - expiry options for notification policies 43
 - Extract from XML Data Object 34
 - HTTP Request 35
 - HTTP Request, example 79
 - Increment Data Object 35
 - Invoke SAS Code 36
 - Invoke SAS Code, example 80
 - Invoke Web Service 37
 - Invoke Web Service, example 82
 - Notify Participant 38

- parameters 32
- parameters, application-specific values 51
- parameters, data object substitution 51
- Remove Status from Process 39
- Schedule Process 39
- Send Workflow Group Notification 41
- Send E-mail 40
- Send E-mail, example 85
- Send Notification by Data Object 40
- Send Notification by Workflow Role 41
- Send Workflow Notification 44
- Set Multiple Process Participants 46
- Set Overdue Status 46
- Set Process Participant 45
- Start Process 46
- Start Process with Label 47
- Stop Process 47
- Stop Process with IDI 47
- timer tool, example 92
- potential owner participant 28
- process
 - adding a subprocess 24
 - adding decision gateways 57
 - creating 22
 - elements of 16
 - example, basic approval process 95
 - example, order fulfillment 98
 - loading templates 73
 - localizing text for 50
 - Schedule Process policy 39
 - Start Process policy for subprocesses or activities 46
 - Start Process with Label policy for processes 47
 - Stop Process policy for subprocesses or activities 47
 - Stop Process with ID policy for processes 47
 - versioning 7
- Process Finished event 31
- process management
 - definition 1
- Process Started event 31
- process tree 18

R

- Remove Status from Process policy
 - parameters 39
- role
 - Send Notification by Workflow Role policy 41

S

- SAS code
 - Invoke SAS Code policy 36
- SAS workflow architecture
 - description 5
 - diagram 5
- SAS Workflow Services 5
- SAS Workflow Studio
 - benefits of 2
 - clip explorer 19
 - diagram editor 16
 - drawing tools 12
 - features 6
 - format toolbar 14
 - interface 9
 - menus 10
 - navigation pane 15
 - process tree 18
 - SAS workflow architecture 5
 - zoom toolbar 13
- Schedule Process policy
 - parameters 39
- Send E-mail policy
 - example 85
 - parameters 40
- Send Notification by Data Object policy
 - parameters 40
- Send Notification by Workflow Role policy
 - parameters 41
- Send Workflow Group Notification policy
 - parameters 41
- Send Workflow Notification policy
 - parameters 44
- sequence, workflow pattern 55
- sequence flow, connecting activities 23
- Set Multiple Process Participants policy
 - parameters 46
- Set Overdue Status policy
 - parameters 46
- Set Process Participant policy
 - parameters 45
- Start Process policy
 - parameters 46
- Start Process with Label policy
 - parameters 47
- status
 - adding 26
 - adding to any process, policy for 32
 - adding to current process, policy for 32
 - assigning to a connection 27
 - default status 26
 - deleting 27
 - editing 27
 - local to an activity 28
 - localizing text for 50

- purpose 26
- Remove Status from Process policy 39
- Set Overdue Status policy 46
- workflow path based on status 56
- Status Addition event 31
- Status Removal event 31
- Stop Process policy
 - parameters 47
- Stop Process with ID policy
 - parameters 47
- subprocess, adding 24

T

- tags
 - adding 52
 - description of 51
 - removing 52
- task initiator participant 28
- task stakeholder participant 29
- templates
 - See also* process
 - loading with the workflow loader utility 73
 - versioning 7
- timer
 - automatically traversing transition, example 90
 - expired, example 92
 - expression examples 90
 - inside activity border, example 89
 - on activity border, example 89

- tool policy, example 92
- Timer Expired event 31

U

- unary operators, decision expressions 76

W

- web service
 - Invoke Web Service policy 37
- workflow group
 - Send Workflow Group Notification policy 41
- workflow instance management 7
- workflow lifecycle management 7
- Workflow Loader utility 73
- workflow path
 - based on expression evaluation 57
 - based on status 56
- workflow pattern
 - alternate paths 56
 - exclusive choice 56, 57
 - exclusive choice, example 95
 - inclusive choice 57
 - join, example 98
 - parallel 56
 - parallel, example 98
 - sequence 55
- workflow role
 - Send Notification by Workflow Role policy 41