



SAS® Workflow Studio 1.3: User's Guide, Second Edition

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2014. *SAS® Workflow Studio 1.3: User's Guide, Second Edition*. Cary, NC: SAS Institute Inc.

SAS® Workflow Studio 1.3: User's Guide, Second Edition

Copyright © 2014, SAS Institute Inc., Cary, NC, USA

All Rights Reserved. Produced in the United States of America.

For a hard copy 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 License Rights; Restricted Rights: The Software and its documentation is commercial computer software developed at private expense and is provided with RESTRICTED RIGHTS to the United States Government. Use, duplication, or disclosure of the Software by the United States Government is subject to the license terms of this Agreement pursuant to, as applicable, FAR 12.212, DFAR 227.7202-1(a), DFAR 227.7202-3(a), and DFAR 227.7202-4, and, to the extent required under U.S. federal law, the minimum restricted rights as set out in FAR 52.227-19 (DEC 2007). If FAR 52.227-19 is applicable, this provision serves as notice under clause (c) thereof and no other notice is required to be affixed to the Software or documentation. The Government's rights in Software and documentation shall be only those set forth in this Agreement.

SAS Institute Inc., SAS Campus Drive, Cary, NC 27513-2414

June 2025

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 trademarks of their respective companies.

version-P1:wfsug

Contents

<i>What's New in SAS Workflow Studio 1.3</i>	<i>v</i>
<i>Accessibility Features in SAS Workflow Studio 1.3</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
The SAS Workflow Architecture	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 Workflow Tree	19
Using the Clip Explorer	20
Chapter 4 • Defining Workflows with SAS Workflow Studio	21
Creating a Workflow	22
Working with Tasks	22
Working with Statuses	26
Working with Participants	29
Working with Policies	32
Working with Data Objects	54
Additional Workflow Features	56
Chapter 5 • Advanced Topics	59
Workflow Patterns	59
Using Timers	68
Deploying and Maintaining Workflows	72
SAS Alert Notification Templates	79
Appendix 1 • Workflow Loader Utility	81
About the Workflow Loader Utility	81
Dictionary	81
Appendix 2 • Decision Expression Examples	89
Operator Support in Decision Expressions	89
Function Support in Decision Expressions	91
Decision Expression Examples	91
Appendix 3 • Policy Usage Examples	95
Invoke REST Web Service Policy Example	95
Invoke SAS Code Policy Example	98
Invoke Web Service Policy Example	100
Send Notification Using SAS Template Service Policy Example	103

Submit a JES Job Policy Example	105
Appendix 4 • Timer Examples	109
Timer Expression Examples	109
Timer Expired and Tool Policy Example	111
Appendix 5 • Basic Workflow Examples	115
Basic Approval Workflow	115
Order Fulfillment Workflow	118
Glossary	123
Index	125

What's New in SAS Workflow Studio 1.3

Overview

The changes and enhancements for SAS Workflow Studio 1.3 include the following:

- additional authorization support for run-time access control and template management
- improved validation when creating and activating workflow templates
- a new template version comparison dialog box
- enhanced policy support for SOAP and REST web services
- new policy support for the SAS Job Execution Service
- Date data object and negative offset support for timer events
- support for a new TODAY function in decision gateways

Authorization Support

For managing access to content, SAS Workflow Studio now supports SAS Web Infrastructure Platform privileges and roles and web-layer permissions. Run-time access control is no longer dependent on SAS metadata and has been updated to support more granular privileges. In addition, SAS Workflow Studio now supports configurable authorization for workflow template management. See [“Workflow Roles and Privileges, and Template Permissions” on page 74](#) and [“Configuring Participants” on page 32](#) for more information.

Improved Workflow Template Validation

SAS Workflow Studio now supports the following template validation logic:

- verify mandatory policy properties
- prevent deletion of reference data objects
- filter data objects by type in selector and drop-down dialog boxes
- verify connections for start nodes and gateways

- verify participant existence on server

Workflow Template Version Comparison

A new template comparison dialog box that displays the workflow tree hierarchy for two versions of a template side-by-side has been added. This new dialog box enables you to easily evaluate the differences between the two versions of the template. See [“Comparing Workflow Template Versions” on page 77](#).

Web Service Policy Enhancements

The Invoke Web Service policy now has two new properties, Error Code and Error Message, to support business logic based on potential error conditions. Also, a new policy action, Invoke REST Web Service, has been added to support REST web services. See [“Invoke Web Service” on page 42](#) and for more information. [“Invoke REST Web Service” on page 39](#)

Job Execution Service Policy Support

A new policy action, Submit A JES Job, has been added. This action enables you to execute code that has been registered with the SAS Job Execution Service and resides on the server. See [“Submit a JES Job” on page 53](#).

Timer Enhancements

Timer events in SAS Workflow Studio now support Date data objects, which allows the dynamic use of datetime values at run time. Date data objects can be used with or without relative time offset expressions. Also, with Date data objects, you can now enter negative relative offsets, which triggers actions prior to the date specified by the data object. See [“Using Timers” on page 68](#).

New TODAY Function

Support for a new TODAY function has been added to decision gateway expressions. This function retrieves the current datetime system value, which enables you to specify a specific date offset from the current date. See [“Function Support in Decision Expressions” on page 91](#).

Accessibility Features in SAS Workflow Studio 1.3

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 pointer 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 website 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 start up and cleanup in each individual activity. For example, it can automate finding the right forms and locating relevant policies, or automate forwarding to the next person in the process.

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 the actual

performance of key processes in real time. These metrics give 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.

BPM can also be used to understand relationships between processes; relationships within the organization and across organizational boundaries. The analysis of those relationships, 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 the automation of business processes. It has evolved to integrate manual processes in which human interaction takes place in series or parallel with the use of technology. For example, when individual steps in a basic workflow require human intuition or judgment, these steps are assigned to 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 type of operating system for the enterprise. The function of this system 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?

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 like 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 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 workflow management into business operations and business offerings based on SAS solutions and products.

Chapter 2

SAS Workflow Architecture

The SAS Workflow Architecture	5
Overview of SAS Workflow Studio	6
Workflow Lifecycle	7

The SAS Workflow Architecture

The SAS workflow architecture provides a suite of applications and services in the SAS Web Infrastructure Platform that work together to model, automate, integrate, and streamline workflows. This architecture provides a platform for more efficient and productive business. The SAS workflow architecture consists of the following components:

SAS Workflow Services

consists of a set of standard interfaces that client applications can use to execute workflows. They provide integration between people, systems, and business logic according to the steps defined in each workflow template.

SAS Workflow Studio

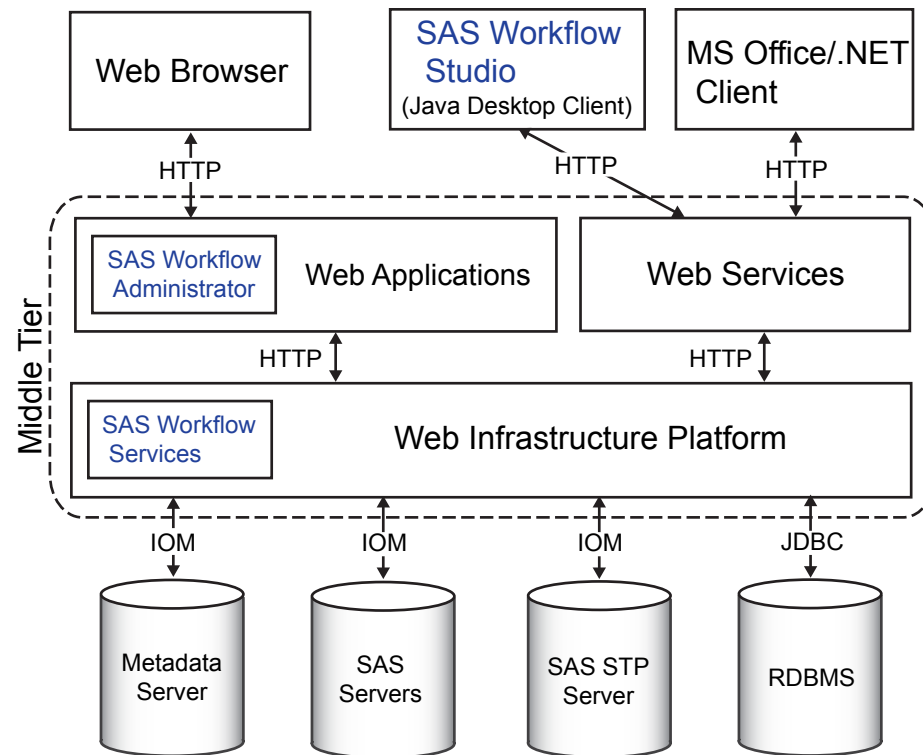
is a graphical workflow-modeling tool for the rapid development of workflow templates.

SAS Workflow Administrator

is a web client plug-in that enables you to monitor active workflow instances.

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

A workflow is a repeatable series of steps that, when accomplished, collectively realize a business objective or policy goal. These goals are typically accomplished within the context of an organizational structure that defines functional roles and relationships. With SAS Workflow Studio, workflow designers define the workflow and its data. Managers, administrators, and end users start individual instances of these workflows and interact with them using a SAS solution.

SAS Workflow Studio enables you to quickly build, organize, and reorganize the workflows and business logic at the heart of the workflow integration within a SAS solution. SAS Workflow Studio supports complex business or application workflow logic and many common workflow and process modeling patterns. Business process logic can be defined and implemented in SAS Workflow Studio, including data-driven flow control using logical decisions (alternate paths) and parallel paths. Workflow designers can use the point-and-click interface of SAS Workflow Studio to define workflows that simplify the automation of sophisticated business processes.

SAS Workflow Studio also provides predefined and customizable policies. Workflow designers can use policies to define actions (for example, send an e-mail) that are triggered by a particular workflow event (for example, task start). The policy represents an action at a specific step within the workflow. Because a task can potentially initiate multiple actions, multiple policies might be associated with a single task. Policies can perform arbitrarily complex programmed actions. For example, policies can send

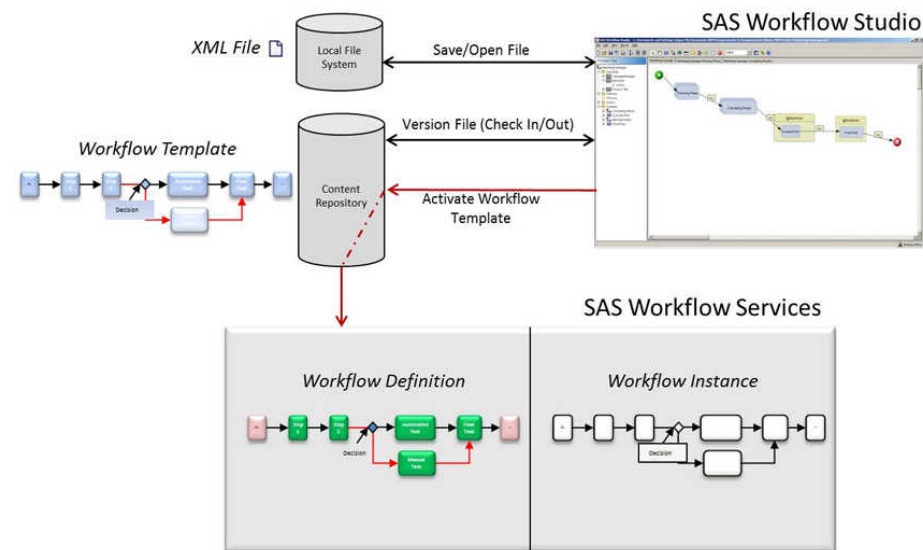
notifications as portal alerts or e-mails, alter process flow, or even provide integration points across applications by invoking SAS code or web services. Policies can also refer to data objects stored in a workflow to add, change, or update peer workflows during execution. For example, you can define a policy for updating a workflow with the completion date and time of a task or subflow.

Workflow definitions 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 workflow definition is activated, it is available for instantiation by workflow-enabled applications. Workflows are versioned by 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: Template changes made in SAS Workflow Studio do not take effect until the new version is uploaded and activated. Once activated, any new instances started use the newly activated version while any currently running instances continue using their respective version of the workflow definition. For details, see [“Deploying and Maintaining Workflows”](#) on page 72.

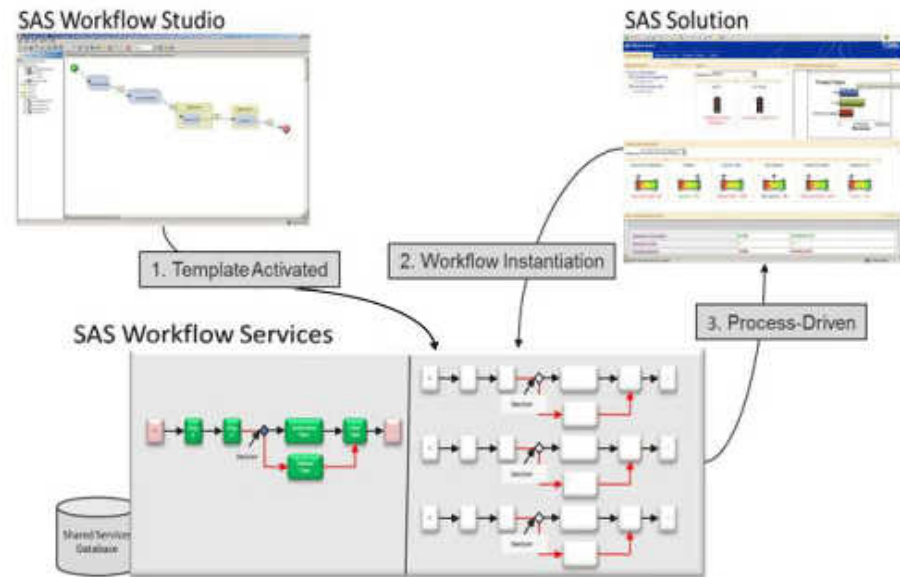
Note: Workflow templates from previous versions are migrated when they are opened in SAS Workflow Studio, and you are prompted to verify the migration.

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

using technologies like web services or published Java APIs to launch workflows, receive workflow status updates, generate notifications and alerts, and monitor workflow progress. Also, some solutions use the SAS Workflow Administrator web plug-in to enable system administrators to view active workflow instances.

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



CAUTION:

It is strongly recommended that administrators do not delete workflow templates from the repository. Deleting a workflow template can cause unpredictable results when instance information is accessed and presented using SAS Workflow Administrator.

Chapter 3

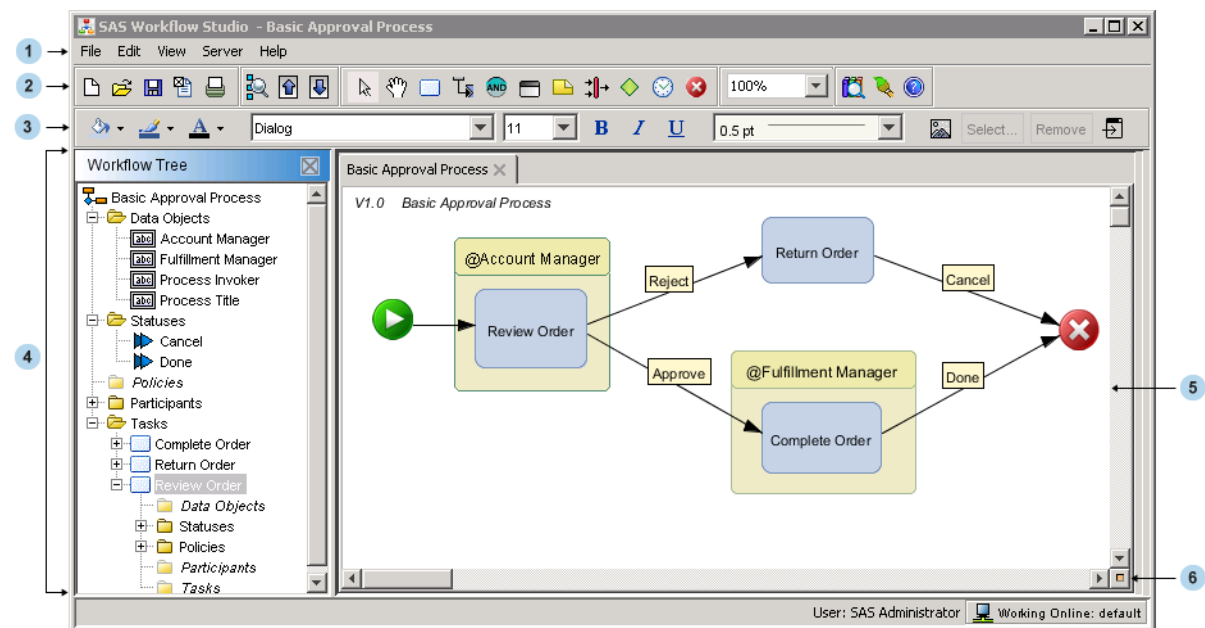
Navigating SAS Workflow Studio

Overview	9
Menus and Toolbars	10
Menus	10
Drawing Tools	13
Zoom Toolbar	14
Format Toolbar	14
Using the Navigator Pane	15
Using the Diagram Editor	16
Elements of a Workflow	16
Workflow Diagram Elements	17
Using the Workflow Tree	19
Using the Clip Explorer	20

Overview

The SAS Workflow Studio user interface consists of two main content panes: the workflow tree and the diagram editor. The workflow tree acts as a content hierarchy pane, organizing the workflow elements associated with the current template 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 [Workflow tree on page 19](#)
- 5 [Diagram editor on page 16](#)
- 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
enables you to modify properties of the active template, including the name, description, and associated attributes and tags.

Options
enables you to set session properties for SAS Workflow Studio such as time and date formats, history size, and other configuration options. Changes made to these settings are not saved after you click **Submit**. Instead, the changes to settings are saved after you close SAS Workflow Studio.

Note: If you are using Microsoft Windows, your user preference settings for SAS Workflow Studio are saved in the **C:\Users\username\AppData\Roaming\SAS** directory. Otherwise, they are saved in the installation directory.

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: workflow tree, format toolbar and Clip Explorer. Some of these commands are also included on the toolbar for quicker access.

Workflow Tree

displays the content hierarchy of the template that is open in the diagram editor. The workflow tree displays all of the template's Data Objects, Statuses, Policies, Participants, and Tasks organized by type and associated task.

**Format Toolbar (F10)**

is used to configure graphics properties of the diagram.

**Clip Explorer (F11)**

stores workflow elements as a library of symbols and formatted workflow 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 workflow template to the SAS Content Repository.

**Open from Repository (F5)**

retrieves or checks out a copy of a workflow template from the SAS Content Repository.

**Manage Templates (F3)**

displays a list of workflow templates currently available in the SAS Content Repository and facilitates management of active workflow definitions.

Compare Versions

compares the version of the template that is currently open in the editor pane with another version in the content repository.

Manage Access

manages user and group access control for the currently open template.

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 website, including a link to the *SAS Workflow Studio: User's Guide*.

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 (spacebar)**
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 Task (A)**
adds a new task to the diagram.
-  **Add Sequence flow Line (C)**
links together tasks and gateways.
-  **Add Logic Gateway (L)**
adds a logic gateway (AND or OR) that controls the execution flow behavior of the connected tasks.
-  **Add Swimlane (W)**
assigns the participants of the contained tasks.
-  **Add Annotations (N)**
adds annotations to the diagram.
-  **Add Merge/Fork Gateway (J)**
represents execution flow forking (splitting) 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, a Task 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 swimlane.

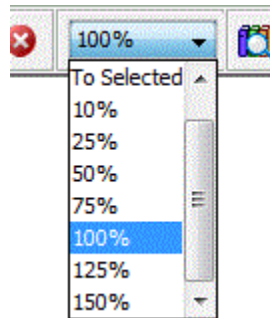
In addition, the following movement key bindings are available:

- 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 in or zoom out of the current diagram. Choosing a zoom factor of 100% yields a 1:1 magnification ratio. A higher zoom percentage zooms in to the diagram, and a lower zoom percentage zooms out from the drawing.



You can also use the navigator pane to view large diagrams. See [“Using the Navigator Pane” on page 15](#) for more information.

Format Toolbar

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

Color Property Controls



Text Property Controls



Outline Property Controls

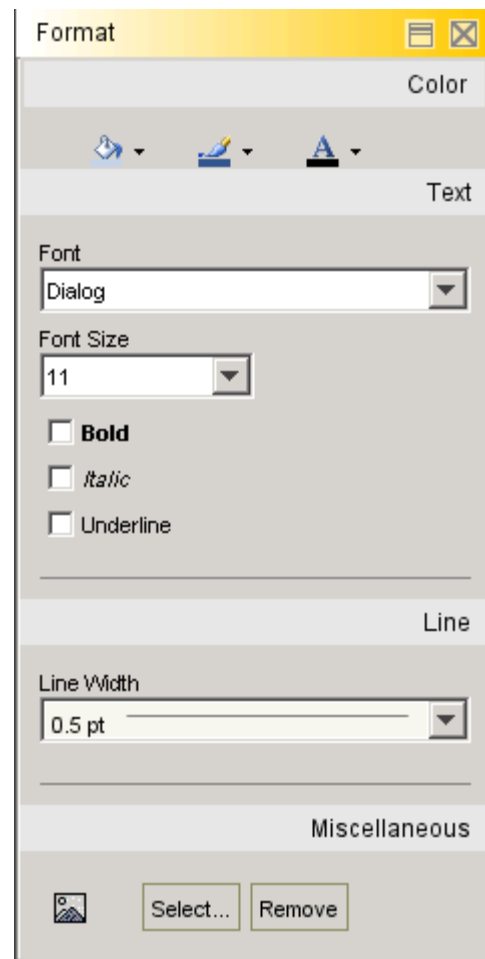


Miscellaneous Property Controls



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

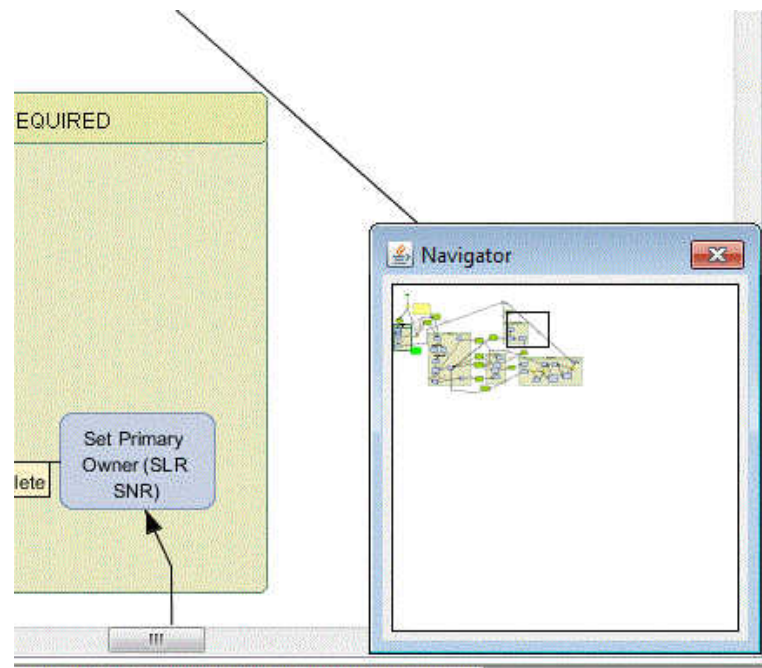
Note: Standard cut and paste operations do not maintain any special formatting. Formatted objects should be reproduced using the Clip Explorer.

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 Workflow

Overview

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

- tasks
- data objects
- policies
- statuses
- participants

Tasks

A task is a step or unit of work in the workflow. Tasks can be atomic nodes (tasks) or can contain collections of related nodes (local subflows). In the workflow tree, tasks are represented by a task icon (□). The root workflow and its subflows are represented by a process icon (🔗).

A task can be manual, or it can be automated by a script or a policy.

Data Objects

Data objects represent the business data values required to perform the workflow tasks. Data objects can mirror values retrieved from external data systems. In this way, SAS Workflow Studio is only loosely coupled with external data systems.

Data objects can be defined at the root workflow level or at the local task level. Data objects defined at the workflow level are global data objects. Global data objects can be accessed by all tasks and policies within the workflow.

Data objects defined at the task level are local data objects. They are accessible only for use by the parent task and its children.

Also, general workflow data and logic should be separated from actionable business data and logic. Workflows embody the abstract data and logic. Data objects embody the relevant portions of business data required to drive path execution and policies represent specific business actions.

Policies

A policy is a workflow element that associates event-driven logic with a task or subflow. Policies are usually triggered automatically by an event such as a status change or a timer event. A policy can reference workflow data to add, change, or update peer workflows at run time. These events occur when there is a change in the workflow. Events can be triggered in either of the following cases:

- a change is generated in the state or status of the task or subflow that was triggered by using a timer for single or repeated actions
- a signal that was received from an external system

Policies support workflow automation in an extremely flexible manner. They are customizable using properties defined for each action.

Statuses

A status is the outcome of a task. Status values are used to trigger state changes in tasks and subflows and to trigger the execution of policies. The status of a task is typically used to trigger the next task. Status values are part of the data that policies use to execute automated tasks. Status values have descriptive names such as Done, Cancel, and Accept to illustrate changes that occur in the workflow. Statuses link the workflow logic and solution logic because they represent transition states between tasks and other workflow elements.

Participants

A participant is a resource that performs the work represented by a workflow task instance. This work is typically manifested as one or more work items assigned to the workflow participant via the tasks in a workflow. (The tasks assigned to a participant are sometimes referred to as the participant's *worklist*.) The list of solution users, groups, roles, or privileges (that is, platform identities) are mapped to standard roles in workflows as a participant. The participant is the actual mapping between the platform or solution user, group, role, or privilege (identity) and the workflow role. Participants drive task access and authorization by linking the workflow authorization roles to the SAS platform users, groups, organizational roles, and privileges.

Workflow Diagram Elements

Global data objects, policies, statuses, and participants are associated with the top-level folders under the workflow root task. Child tasks can also contain locally defined data objects, policies, statuses, and participants. Local elements that exist in the context of a specific task are accessible only at the task level, not by the peer tasks or gateways. In addition, the global elements are not re-created as local values associated with the child elements. Therefore, refer to solution-specific best practices to specify data object scope at design time.

Each template begins with a Start node and contains one or more work items (tasks or subflows) 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 tasks. Likewise, the Start node can be used to initiate multiple tasks.

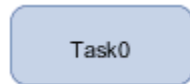
The following elements can be used in a workflow diagram:



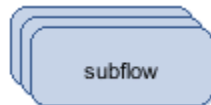
The Start node must precede the first task in the template unless a timer is used to offset the start of the instance.



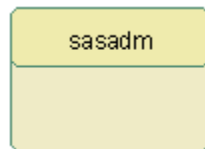
The Stop node must be connected to any task that leads to instance termination.



Tasks are individual work items in the workflow. They can represent automated or manual work items.



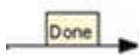
A task element with a stacked appearance represents a subflow. A subflow contains one or more tasks that might then represent subflows resulting in a workflow hierarchy. You can create subflows and edit the contained tasks from the drawing editor.



Swimlane elements are used in SAS Workflow Studio to group tasks assigned to the same participant. They can be explicitly assigned to a Participant object, or they can be implicitly assigned via a swimlane policy. The swimlane policy might be a dynamic reference to a data object. If so, then the user, group, role, or privilege value defined by the specified data object is derived at run time.



The **Sequence Flow tool** is used to connect workflow elements.



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

Called from "Manage External Process", demonstrating interaction with other processes.

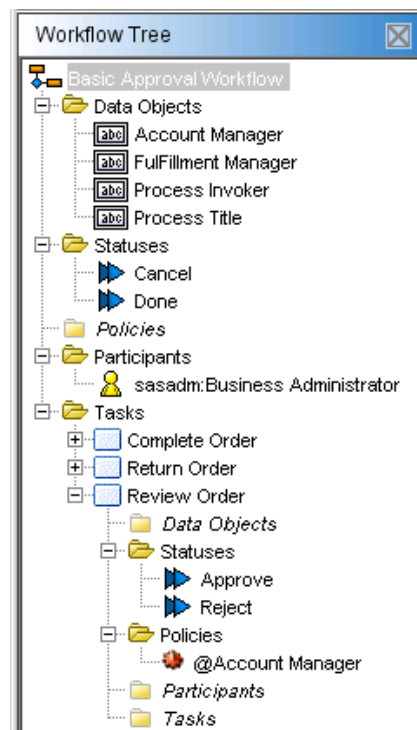
Annotations are used to hold additional information. These notes are for presentation only and are not associated with a running workflow instance.

You can use the drawing tools in the toolbar to place tasks on the diagram editor and connect them using the Sequence Flow element. You can also select and right-click any task or connection on the diagram editor to add objects. Alternatively, you might use the workflow tree context menus to add tasks and other workflow elements.

Using the Workflow Tree

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

Figure 3.4 Workflow Tree



The workflow root contains top-level folders for data objects, statuses, policies, participants, and all of the associated tasks. Each task folder can contain local definitions for each of these object types.

In the workflow 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.

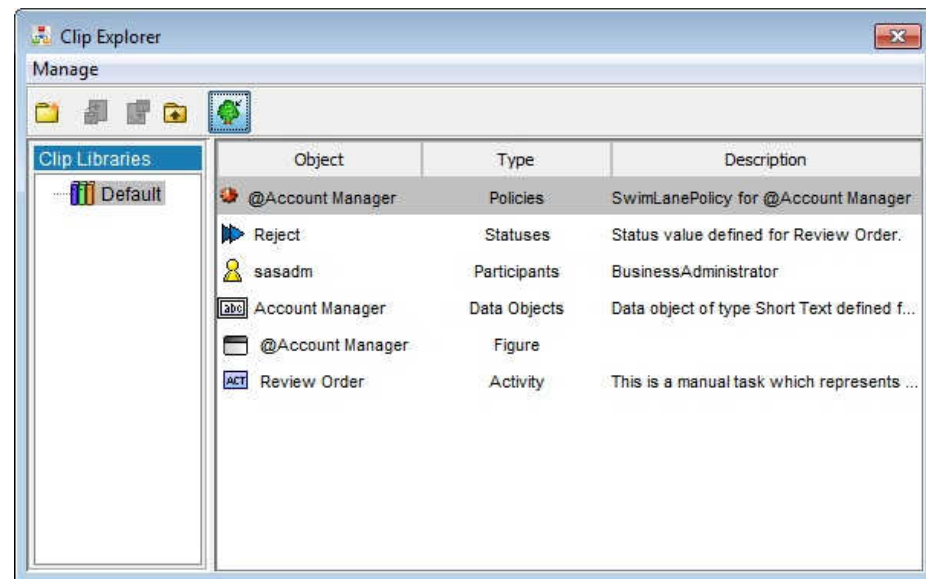
Using the Clip Explorer

The Clip Explorer utility can store a library of symbols and workflow objects that can be shared using the copy and paste commands. Copying figures and workflow tree objects to the clipboard creates a reusable copy of these formatted objects. If an object in the Clip Explorer is copied into a drawing or a workflow tree, the properties of the new instance can be edited independently. As a result, the properties of the original object are maintained. 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. You can also access another workflow 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 Workflows with SAS Workflow Studio

Creating a Workflow	22
Working with Tasks	22
Adding Tasks	22
Connecting Tasks in a Sequential Flow	23
Editing Tasks	23
Deleting Tasks	24
Adding a Subflow	24
Aligning Tasks	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	29
Workflow Roles	29
Adding Participants	30
Editing Participants	31
Deleting Participants	31
Assigning Participants	31
Configuring Participants	32
Working with Policies	32
Overview of Policies	32
Adding Policies to a Workflow	33
Editing Policies	33
Deleting Policies	33
Assigning Policies to a Task	34
Policy Events	34
Policy Events and Scope	34
Policy Actions	34
Working with Data Objects	54
Overview	54
Adding Data Objects	54
Editing Data Objects	55
Deleting Data Objects	55
Assigning Data Objects	56

Additional Workflow Features	56
Text Localization	56
Custom Attributes	56
Data Object Substitution	56
Using Tags to Categorize and Filter Workflow Templates	57

Creating a Workflow

A workflow is a series of tasks, together with the participants and the logic required to execute the tasks. A workflow can include policies, status values, and data objects. To create a workflow, follow these steps:

1. Start SAS Workflow Studio.

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

2. In the workflow tree, right-click on the root workflow node (Untitled1) and select **Edit**.

The Template Properties dialog box appears.

3. In the **Template Name** field, replace **Untitled1** with the name for the new workflow.
4. (Optional) Add a description for the workflow in the **Description** field.
5. Add and configure the required [tasks](#), [data objects](#), [statuses](#), [policies](#), and [participants](#).
6. Add [sequence flows](#) to connect the workflow elements.

Note: Changes to workflow templates do not take effect on the server until the revised template is uploaded and activated. Instances of a workflow that are already running continue to run with the version of the template definition with which they were started. When new workflow instances are started, they use the newly activated version of the template.

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

Working with Tasks

Adding Tasks

A task is a workflow element that represents a step or unit of work in the workflow. To define a new task, follow these steps:

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

2. In the toolbar, select the Add Task icon (.
3. Click anywhere on the diagram editor to create a task.
4. Move the task to the desired position on the diagram editor by clicking the selection tool () on the toolbar, selecting the desired task, and dragging it to the new location.

The default name of the first task is Task0. The second task is named Task1, and so on.

Connecting Tasks in a Sequential Flow

To define a logical sequence of tasks, 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 tasks 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 a Task

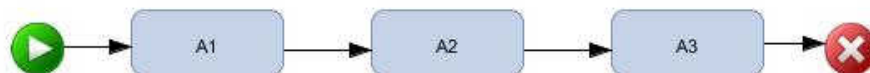


2. In the diagram editor, drag the mouse pointer from the first task figure toward the second task figure.

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

For example, a workflow might contain Tasks A1, A2, and A3. Task A1 executes first, followed by A2 upon completion of A1, and so on, until A3 completes to terminate the workflow. Such a workflow could be modeled as follows:

Figure 4.2 Sequential Workflow Flow



Editing Tasks

To edit a task, follow these steps:

1. Open the Edit Task dialog box by double-clicking the task.
Alternatively, right-click on the task in either the diagram or the workflow tree and select the **Edit** option.
2. Enter the desired name in the **Task Name** field.
3. (Optional) Add a description for the task 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 56](#).
6. (Optional) Add custom attributes. For more information, see [“Custom Attributes” on page 56](#).
7. Click **OK** to save the updated task definition.

Deleting Tasks

To delete a task, follow these steps:

1. Right-click on the task in either the diagram editor or workflow tree and select the **Delete** option.

Alternatively, select the task in the diagram editor or workflow tree, and press the Delete key or CTRL+X.

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

The deleted task is no longer visible in the diagram editor or the workflow tree.

TIP To delete multiple tasks 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 task is deleted. If you attempt to delete an element that is referenced elsewhere in the template, 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 Subflow

As described in [“Creating a Workflow”](#), a workflow is a series of tasks, together with the participants and the logic required to execute the tasks. A subflow is a workflow that is a child of a parent workflow. You can use subflows to refactor a larger workflow template into smaller components. Grouping elements together into a subflow helps organize and reuse business logic. Using subflows improves readability of the workflow template and promotes consistent reuse of business logic within the organization.

To create a subflow, follow these steps:

1. In the diagram editor, right-click on the task that you want to convert to a subflow and then select the **Create Subflow from** option.
2. When prompted, select one of the following options:
 - **Create New**
 - **Load from file**
 - **Open from repository**
3. Select **OK** to save the subflow definition.

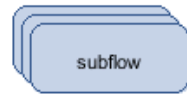
For each option, the relevant diagram opens. For the **Create New** selection, the diagram is blank and untitled. For the other options, the diagram is a copy of the existing designated template.

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

After an element is added to the new diagram, the transformed task is denoted with the process icon () in the workflow tree. The subflow data objects, statuses, policies, and participants are also visible in the workflow tree hierarchy.

After adding a subflow, the task symbol in the parent workflow becomes a stacked icon representing a set of tasks rather than a single step in the template. The following figure shows an example of subflow notation:

Figure 4.3 Subflow Symbol

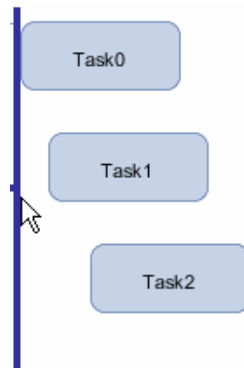


Aligning Tasks

Multiple tasks can be aligned using the alignment tool.

To activate the alignment tool, follow these steps:

1. In the diagram editor, hold down the Ctrl key, and click the left mouse button. A blue plus sign () appears on the diagram editor.
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 spacebar:

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

Working with Statuses

Overview

Statuses are used to denote the outcome of a workflow step as values associated with a logical transition from one task 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, when that status changes to the associated value, then the workflow advances to the next task. The status change causes the current task to complete.

For example, the workflow contains two tasks, A and B, with a connection from Task A to Task B. Task B is initiated after Task A has been approved. We can represent this task flow by assigning an Approve status to the connection between Task A and Task B. The following figure shows this flow.

Figure 4.4 Status Example



Adding Statuses

To add a new status, follow these steps:

1. In the workflow tree, right-click the **Statuses** folder and select **New Status**.
Alternatively, right-click on a task 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 56](#).
5. (Optional) Add custom attributes. For more information, see [“Custom Attributes” on page 56](#).
6. Click **OK** to save the new status definition.

Editing Statuses

To edit an existing status definition, follow these steps:

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

Deleting a Status

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

1. Right-click on the status in the workflow tree and then select **Delete**.
Alternatively, you can select the status directly in the workflow tree and press the Delete key or CTRL+X.
2. Select **Yes** in the confirmation dialog box to permanently remove the selected status.

Assigning a Status

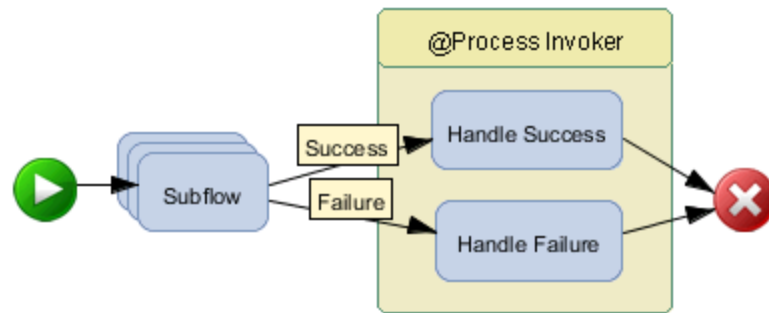
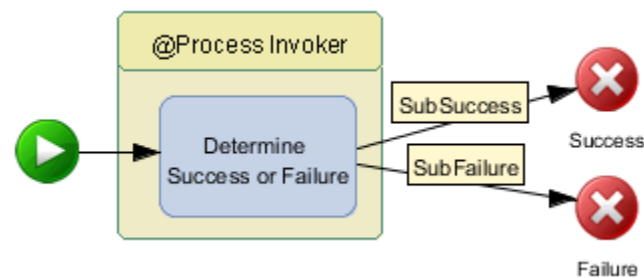
To assign a status, right-click on a connection between tasks and then select **Change status**. This opens a menu containing the status values defined for the workflow. If the desired status is not defined, then you can select the **New status** menu option to define the new status value.

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

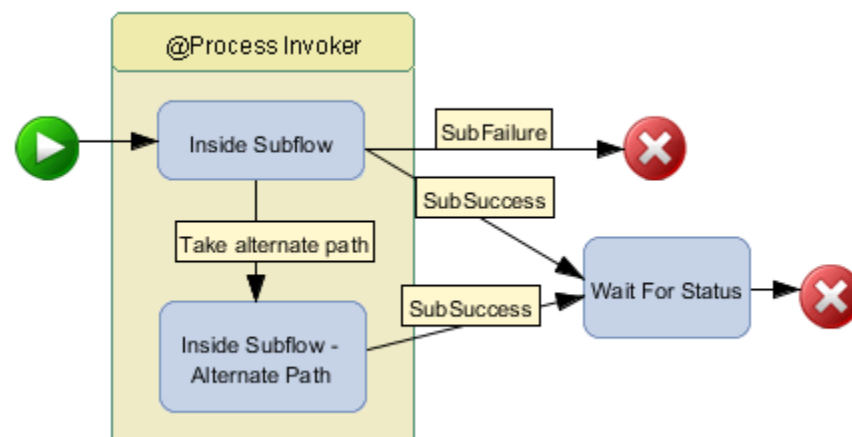


TIP You can use the **Selection** tool (🖱️) to reposition the task figures in the diagram editor to ensure that the statuses are visible. Also, you can reposition the connection endpoints by selecting the target connection. Then, select the endpoint and drag it to the new location in the figure by releasing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

If the termination node is within a local subflow, transitions with this status are executed in the parent workflow. This means status propagation is triggered in the parent workflow via the termination node with status.

Figure 4.5 Parent Workflow**Figure 4.6** Subflow Included in Parent Workflow

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

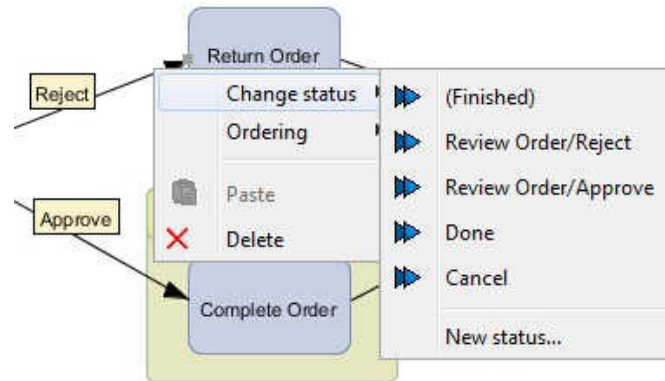
Figure 4.7 Subflow Joined into a Task

Local Statuses

When you define a status locally for a task, rather than globally for the entire template, SAS Workflow Studio displays the relative element path. An element path is an alternative way to refer to an object when using the object ID is not feasible. An element 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 workflow. The Review Order task includes the local statuses Approve and Reject, and their element paths include the object's ancestors.

Figure 4.8 Local Status Values



Working with Participants

As mentioned previously, a workflow often requires user interaction with the system to complete a task. In addition, automation of task assignment within the workflow to a particular set of users might be desired. These tasks are assigned to the appropriate members within the organization using specific workflow roles. SAS Workflow Studio uses participants to assign the individuals potentially involved with a workflow to the workflow or to specific tasks. The individuals involved must be associated with a platform identity (users, groups, organizations, and privileges). Participants map platform identities to workflow roles such as potential owner, actual owner, business administrator, and so on. Specific role-based permissions have been defined for each of these roles that control participant's access to workflow tasks.

Workflow Roles

Participants are the mapping between the authorized platform identities and workflow roles. 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.

Note: The Task Initiator role is set by SAS Workflow Services, but it is not explicitly used for authorization.

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 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. The Actual Owner role should be assigned to a single user because a task can have only one owner. A potential owner must claim a task in order to become an actual owner. When the task is claimed, it is locked and cannot be claimed by another potential owner without first being released by the actual owner.

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 Workflow Services.

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 Recipient

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.

Note: The Notification Recipient role is not currently used by SAS Workflow Services.

By default, specific abilities have been defined for these participant workflow roles, and the roles 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 workflow tree, right-click the top-level or local **Participants** folder and then select **Add Participant**.

Alternatively, right-click a task 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 Workflow Role**, choose the relevant role from the predefined participant workflow roles.
4. For **Identity**, select the **User Name**, **Group Name**, **Organizational Role Name**, or **Privilege Name** option and enter the desired value.

If you are logged on to the SAS platform, you can click  to search for a value for one of the identify fields. Enter a search string in the **Search** field, and click . SAS Workflow Studio searches not only the display labels used for the items that you are searching for but also the internal names associated with those items. For example, if you search for user names that contain **adm**, and the internal name of the user name **joe** contains the string **adm**, then the search returns the user name **joe** in addition to any user names that contain **adm**.

The **Organizational Role Name** field is case sensitive. All other fields are not case sensitive.

To display all of the possible values for a field, enter an asterisk (*) in the **Search** field.

See “Configuring Participants” on page 32 and “Workflow Roles and Privileges, and Template Permissions” on page 74 for more information.

5. (Optional) Add localized versions of the name.

For more information, see [“Text Localization” on page 56](#).

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

Participants can be also assigned using swimlanes. For details, see [“Assigning Participants” on page 31](#).

The workflow tree displays both the identity name and associated participant role. For example, if the user name is **admin** and the participant's role is Business Administrator, the tree shows **admin: Business Administrator**.

Editing Participants

To edit an existing participant definition, follow these steps:

1. In the 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 workflow tree and then select **Delete**.
Alternatively, select the participant in the workflow 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 tasks directly or indirectly via swimlane.

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

- Drag the participant from the workflow tree to the relevant task 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 task in the workflow tree.
- Define a swimlane and associate it with the relevant participant definition. Then assign the task 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, organizational role, or privilege identity.
2. Create a swimlane and assign the data object as the value.

The data object acts as a placeholder (designated by the @ preceding the data object name) for the value, which is derived and assigned at run time.

3. Drag the task onto the relevant swimlane in the diagram.

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

Configuring Participants

Run-time workflow participants are defined using the SAS Authorization Service. The following participant identity types are supported:

- 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.
- If a privilege type is specified, then the name must be a SAS platform privilege.

Note: You can use the SAS Web Administration Console to assign roles to users and groups and to edit the privileges that are associated with a role. See *SAS Intelligence Platform: Middle-Tier Administration Guide* for more information.

Working with Policies

Overview of Policies

Often, tasks require that actions be automatically triggered by specific workflow events. SAS Workflow Studio provides a way to define these actions through policies. Policies are triggered by events and execute actions. Actions can be configured to perform automated tasks such as the following:

- Indicate how and when notifications of deadlines should be sent.
- Notify the task owner of related task events such as task start or task stop.
- Notify users of arbitrary task events such as task start, task stop, or data object updates.
- Integrate the system with other back-end systems.

A task might initiate multiple actions simultaneously, so you can associate multiple policies with a single task.

Note: A policy encapsulates executable business logic. Business data should not be defined within a policy. Data objects should be used to hold relevant business data in the workflow. Data objects can be updated within policies. However, in general, policies should be limited to actions and not used for data management.

Note: Policies are asynchronous and are subject to timing for execution order. For example, you might use a policy to copy the value from a root data object to a local task data object. In this case, the source object must be set and must not be subject to change.

Adding Policies to a Workflow

To define a new policy, follow these steps:

1. In the workflow tree, right-click the top-level or local **Policies** folder in the workflow tree and then select **New Policy**.

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

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

Name

specifies the name of the policy. The name is required.

Description

specifies the policy description. Specifying a description is optional.

Event

specifies the workflow event that triggers the policy. The event is required. See [“Policy Events” on page 34](#) for more information.

Action

specifies the policy type. Most policy types require that you specify additional values, and the Edit Policy dialog box displays the Properties section for the type that you selected. The action is required. See [“Policy Actions” on page 34](#) for more information.

3. Enter the required property values for the policy type that you selected.

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

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

Editing Policies

To edit an existing policy, follow these steps:

1. In the workflow 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 workflow tree and then select **Delete**.

Alternatively, select the policy directly in the workflow 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 a Task

To assign a policy to a task, drag the desired policy from the workflow tree to the relevant task in the diagram editor.

Alternatively, you can copy or drag the policy element into the **Policies** folder for the target task in the workflow tree. To reuse a global definition of a policy, copy the policy element. To demote a global definition to a local definition, drag the policy element.

Policy Events

The following events can be used to trigger policies:

Task Started

Generated when the task state changes to Started.

Task Finished

Generated when task state changes to Finished.

Status Addition

Generated when the specified status is added to task. When no status is specified (when the default — is selected), the associated policy is executed for any status addition.

Status Removal

Generated when the specified status is removed from a task.

Timer Expired

Generated when a timer associated with a task fires.

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. See “[Policy Events and Scope](#)” on page 34 for additional information.

Participants Updated

Generated when the participants list of the task is updated.

Policy Events and Scope

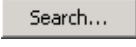
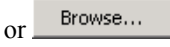
Except for Data Object Updated events, the scope of all events is local to the task that contains the policy.

The scope of a Data Object Updated event starts at the point where the data object that triggers the event is defined and propagates up the workflow tree. Therefore, a policy triggered by a Data Object Updated event must be defined at the same level or above the affected data object.

Policy Actions

Entering Data for Policy Actions

In the dialog boxes for defining policies, if a field is required, the name of the field is in bold type.

Many fields in the policy action dialog boxes have buttons such as , , or  that can display a list of valid values for that field. However, in order for these buttons to be able to display the information, you must be logged on to the SAS Server.

Add Status to External Workflow

The Add Status to External Workflow policy is used to propagate the selected status between workflow instances. The referenced data object can be updated while the parent workflow is active.

The following properties can be defined for this policy:

Property Name	Description
Workflow ID Data Object	Specifies the element path to a data object that contains the ID value of the target workflow.
Status Label	Specifies the element path to the status that is added to the target task.

Add Status to Task

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

The following properties can be defined for this policy:

Property Name	Description
Task	Specifies the element path of the task or subflow to which the status is added.
Status	Specifies the element path of the status to be added.

Copy Data Object

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

The following properties can be defined for this policy:

Property Name	Description
Source Data Object	Specifies the element path of the data object whose value is to be copied.
Target Data Object	Specifies the element path of the target data object where the value is to be copied.

Note: This policy action triggers a Data Object Updated event. See [“Policy Events and Scope” on page 34](#) for additional information.

Copy Data Object from External Workflow

The Copy Data Object from External Workflow policy is used to copy a data object value from another instance (remote or external) into the current workflow.

The following properties can be defined for this policy:

Property Name	Description
Source Workflow ID Data Object	Specifies the element path to a data object that contains the workflow ID value of the source workflow.
Source Data Object Label	Specifies the element path of the external data object whose value is to be copied.
Target Data Object	Specifies the element path of the target data object in the current workflow where the value is to be copied.

Note: This policy action triggers a Data Object Updated event. See [“Policy Events and Scope” on page 34](#) for additional information.

Copy Data Object to External Workflow

The Copy Data Object to External Workflow policy is used to copy data object values from the current instance to another workflow instance (remote or external).

The following properties can be defined for this policy:

Property Name	Description
Source Data Object	Specifies the element path of the local data object whose value is to be copied.
Target Workflow ID Data Object	Specifies the element path to a data object that contains the workflow ID value of the target workflow.
Target Data Object Label	Specifies the element path of the target data object where the value is to be copied.

Note: This policy action triggers a Data Object Updated event. See [“Policy Events and Scope” on page 34](#) for additional information.

Copy Participants to Task

The Copy Participants to Task policy is used to copy participant values between tasks.

The following properties can be defined for this policy:

Property Name	Description
Source Task	Specifies the element path of the task from which the participant is copied.
Source Role	Specifies the workflow role that is copied from the source task.

Property Name	Description
Target Task	Specifies the element path of the task to which the participant is copied.
Target Role	Specifies the workflow role for the target task.

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

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 42](#) 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 properties can be defined for this policy:

Property Name	Description
XML Data Object	Specifies the element path to the data object that contains the XML value from which the value is extracted.
Output Data Object	Specifies the element path to the data object to which the extracted text 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 property.

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** property, then the policy stores the value **Harry Potter** in the target data object.

If the XML schema includes namespace prefix notations, then you should declare and use the namespace in the XPath expression in a manner similar to the following:

```
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 properties can be defined for this policy:

Property Name	Description
Directive Name	Specifies the name of the directive. The value can be specified using data object substitution. (See “ Data Object Substitution ” on page 56.) If you are logged on to the SAS platform, you can click  to display the list of registered directives.
URL	Specifies the target URL to invoke, execute, or access. The value can be specified using data object substitution. If the value contains non-ASCII characters, you should select Encode the data objects in the URL when the policy executes . See the standard for Uniform Resource Identifier (URI): Generic Syntax at http://tools.ietf.org/html/rfc3986 for information about character encoding.
Parameters	Specifies any required additional HTTP parameter values. The value can be specified using data object substitution.
Status Code Data Object	Specifies an optional data object to hold the value of the return code of the HTTP request.
Disable Authentication	Specifies whether the proxy ticket for the HTTP request URL should be generated. Selecting Disable Authentication prevents the HTTP request policy from generating a proxy ticket to add to the URL.

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.

If the URL is not specified or if it references a data object that contains an empty value, then the **Directive Name** property 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 a URL is specified, it is combined with the directive. 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 95 for a detailed example.

Note: Only the POST method is supported.

Increment Data Object

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

The following properties can be defined for this policy:

Property Name	Description
Data Object	Specifies the element path to a data object to be incremented.
Increment Value	Specifies the numeric value by which the data object should be incremented. The default value is 1. Using operand substitution in place of a value is also supported.

Invoke REST Web Service

The Invoke REST Web Service policy is used to invoke a REST web service over HTTP.

The following properties can be defined for this policy:

Property Name	Description
Web Service Location	Specifies the URL of the web service endpoint. The value can be specified using data object substitution. (See “Data Object Substitution” on page 56.) If you are logged on to the SAS platform, you can click  to display the list of all of the registered REST web services.
HTTP Method	Specifies the HTTP method for the web service. Specify GET, PUT, POST, or DELETE.
Content Type	Specifies the type of the body of the request. The content type is used for POST and PUT requests only. The value can be specified using data object substitution. If no content type is specified, the default value is “text/plain”.
Web Service Input	Specifies the Text data object used for input. The content of the data object can use data object substitution. The value of the Text data object should contain the body of the REST request message.
Web Service Output	Specifies the Text data object used for output. The value of the Text data object contains the body of the REST response message.
Error Values	
Error Code	Specifies the data object (number type) for storage for the code value in the case of an error. You can click  to display a list of the existing Number data objects.
Error Message	Specifies the Text data object for storage for the message value in the case of an error. You can click  to display a list of the existing Text data objects.

Property Name	Description
Authentication	
User Name	Specifies the user name. The value can be specified using data object substitution. If you are logged on to the SAS platform, you can click  to search for registered users. If a value is specified, then security-level headers are added to the request at the transport level. <i>Note:</i> The user name and password properties are required only if the target service mandates access control.
Password	Specifies the password. The value can be specified using data object substitution. If this property is specified, then it is used in the security-level headers on the web service request. <i>Note:</i> The user name and password properties are required only if the target service mandates access control.

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. Also, if a fault occurs, then the response object is not set, and the error code and message values are stored in the specified error data objects.

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 workflow.

The following properties can be defined for this policy:

Property Name	Description
SAS Source Code	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. (See “Data Object Substitution” on page 56.)
Server	Specifies the name of the Workspace Server to use to execute the SAS code. The value can be specified using data object substitution.
Repository	Specifies the name of the metadata repository in which the Workspace Server is defined. The value can be specified using data object substitution.

Property Name	Description
Pass all root data objects?	Controls the scope of root data objects when executing the SAS code. If this property is checked, then each root-level data object is converted to a macro variable. The macro variables are available for use within the SAS code. Each macro variable is prepended to the SAS code before the code is submitted to the server for processing. See “Macro Name Generation Rules” for more information.
Mappings	
Macro Variable Name	Enables you to add a macro variable to the list of SAS Macro Variables. Enter the name of the macro variable in the field, and click Add . Macro variable names are limited to 32 characters. Values can be specified using data object substitution.
Data Object and Task	Lists the data objects, with their associated tasks, that have been defined for the workflow. You can use the arrows ( and ) to reorder the data objects in this list. See “Mapping Data Objects to SAS Macro Variables” for more information.
SAS Macro Variable	Lists the SAS macro variables that have been defined for this workflow. Click  to delete a macro variable from the list. See “Mapping Data Objects to SAS Macro Variables” for more information.

Macro Name Generation Rules

When root data objects are passed in, 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;
```

To customize the default prefix, follow these steps:

1. Open SAS Management Console and navigate to **Configuration Manager** ⇒ **SAS Application Infrastructure** ⇒ **Workflow Services 9.4** ⇒ **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 95](#) for a detailed example.

Mapping Data Objects to SAS Macro Variables

To map a data object to a macro variable, follow these steps:

1. Select a data object.
2. Select the macro variable that you want to map the data object to.
3. Click  to send the value of the data object to the macro variable, or click  to assign the value of the macro variable to the data object.

SAS Workflow Studio adds a connecting arrow between the data object and the macro variable that it is mapped to.

For example, the mapping shown in the following figure sends the value of the Total Category data object to the VAR macro variable. The mapping also assigns the value of the RESULT macro variable to the Total data object.

Data Object	Task		SAS Macro Variable	Remove
Total	InvokeSASCode ...		var	
Total Category	InvokeSASCode ...		result	

To delete a mapping, select the arrow connecting the data object and the macro variable, and click .

Invoke Web Service

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

The following properties can be defined for this policy:

Property Name	Description
Web Service Location	Specifies the URL of the web service endpoint. The value can be specified using data object substitution. (See “Data Object Substitution” on page 56 .) If you are logged on to the SAS platform, you can click  to display the list of all of the registered SOAP web services.
Web Service Action	Specifies the SOAP action header value for the web service. The value can be specified using data object substitution.
Web Service Input	Specifies the XML data object used for input. The value can be specified using data object substitution. The associated schema property 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.

Property Name	Description
Web Service Output	Specifies the XML data object used for output. The associated schema property 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.
Error Values	
Error Code	Specifies the data object (number type) for storage for the code value in the case of an error. You can click  to display a list of the existing Number data objects.
Error Message	Specifies the data object (text type) for storage for the message value in the case of an error. You can click  to display a list of the existing Text data objects.
Authentication	
User Name	Specifies the user name. The value can be specified using data object substitution. If you are logged on to the SAS platform, you can click  to display the list of registered users. If a value is specified, then security-level headers are added to the request at both the transport level and the message level. <i>Note:</i> The user name and password properties are required only if the target service mandates access control.
Password	Specifies the password. The value can be specified using data object substitution. If this property is specified, then it is used in the security-level headers on the web service request. <i>Note:</i> The user name and password properties are required only if the target service mandates access control.

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 error code and message values are stored in the specified error data objects.

See [Appendix 3, “Policy Usage Examples,”](#) on page 95 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 a task starts or finishes.

The following properties can be defined for this policy:

Property 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 a task has started, follow these steps:

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

If you expand the task's **Policies** folder in the workflow tree, then you should see the Workflow:Notify Participant policy definition. The notification is triggered by the Task Started event that is generated when the associated task starts. When the policy executes, it evaluates the task to see whether the Actual Owner workflow role exists on the current task. 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, policy execution obtains the person or group defined for the workflow role. The policy execution then looks up that person or group in the SAS Authorization Service and obtains the relevant notification settings.

To create a policy that notifies a participant that a task has finished, follow these steps:

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

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

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

Note: The Send E-mail policy is best suited for testing templates. The Send Workflow Notification and Send Workflow Group Notification policies are recommended for use in production applications because they support flexible configuration options via platform services.

Remove Status from Task

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

The following properties can be defined for this policy:

Property Name	Description
Task	Specifies the element path for the target workflow or task.
Status	Specifies the element path for the status that is removed from the target task.

Schedule Task

This policy is used to schedule the initiation of a task or subflow.

The following properties can be defined for this policy:

Property Name	Description
Task to Schedule	Specifies the element path for the scheduled task or subflow.
Schedule Timer Data Object	Specifies the data object containing the expression that defines when the scheduled task or subflow 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: <i>MM/dd/yyyy hh:mm:ss AM PM</i> For example, 12/31/2010 11:59:00 PM. You can use a relative expression to configure the task to start at a moment in time relative to when the policy is triggered. Relative expressions are based on when the parent element starts. You can use the reserved keyword Now, or a plus sign, followed by one or more integers, followed by a unit of measure, as shown in Table 4.1. You can also combine sequences of plus signs, integers, and units of measure, or you can specify a cron expression, as shown in Table 4.2.
End Timer Data Object	Specifies the data object containing the expression that defines when the schedule timer should be stopped. This property is optional.

Table 4.1 Timer Expressions in the Schedule Task Policy

Unit or Keyword	Description	Example	Explanation
Now		Now	The timer fires immediately.
s	Seconds	+45s	The timer fires in 45 seconds.
m	Minutes	+30m	The timer fires in 30 minutes.
h or H	Hours	+6h	The timer fires in 6 hours.

Unit or Keyword	Description	Example	Explanation
d	Days	+2d	The timer fires in 2 days.
w	Weeks	+1w	The timer fires in 1 week.

Table 4.2 Timer Expression Examples in the Schedule Task Policy

Example	Explanation
+2d+12h+30m	The timer fires in 2 days, 12 hours, and 30 minutes.
0 0 12 * * ?	The timer fires every day at 12 p.m. (noon).

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 properties can be defined for this policy:

Property Name	Description
To	Specifies a recipient's e-mail address.
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 56 .
Message	Specifies the body text of the e-mail message. The value can be specified using data object substitution.

Note: The Send E-mail policy is best suited for testing templates. The Send Workflow Notification and Send Workflow Group Notification policies are recommended for use in production applications because they support flexible configuration options via platform services.

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 properties can be defined for this policy:

Property Name	Description
Recipients(s)	Specifies the primary recipients. Separate multiple recipients with commas.
CC Recipient(s)	Specifies the copied recipients. Separate multiple recipients with commas.
BCC Recipient(s)	Specifies the blind-copied recipients. Separate multiple recipients with commas.
Group Recipient(s)	Specifies the primary group recipients. Separate multiple group recipients with commas.
Group CC Recipient(s)	Specifies the copied group recipients. Separate multiple group recipients with commas.
Group BCC Recipient(s)	Specifies the blind-copied group recipients. Separate multiple group recipients with commas.
Description	Specifies the description or e-mail subject for the notification.
Template	Specifies the SAS notification template to use. The default template is 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, you can click  to display the list of all of the registered directives.
Notification Variables	Specifies one or more element 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 element path values with commas.
HTTP Parameters	Specifies one or more element paths for data object that represent name-value pairs that are used as HTTP parameters. The data object name corresponds to the HTTP parameter name while the data object value is used as the parameter value. Separate the element path values with commas.
Expiry	

Property Name	Description
Action on Expiry	Specifies the action to perform if the workflow is not completed by the date defined by the Expiration Date Data Object property. The available expiry options are as follows: None no expiration action for this notification. <i>Note:</i> The Expiration Date Data Object property 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 workflow instance using the specified template.
Expiration Date Data Object	Specifies the data object that defines the expiration date of the policy. <i>Note:</i> This property is available only when the Action on Expiry property is set to a value other than None.
Expiry Recipients(s)	Specifies one or more SAS Platform users to whom notifications are sent when the policy reaches its expiration date. Separate multiple recipients with commas. <i>Note:</i> This property is available only when the Action on Expiry property is set to Reroute.
Expiry Group Recipients(s)	Specifies one or more SAS Platform groups to which notifications are sent when the policy reaches its expiration date. Separate multiple group recipients with commas. <i>Note:</i> This property is available only when the Action on Expiry property is set to Reroute.
Workflow	Specifies the workflow template to start when the policy reaches its expiration date. If you are logged on to the SAS platform, you can click  to display the list of activated templates. <i>Note:</i> This property is available only when the Action on Expiry property is set to Start Workflow.

All policy properties 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 56. The **Expiration Date Data Object** property is an element path to a data object and does not support data 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** property 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.

Note: This notification is directed, and SAS platform users cannot opt out based on preferences.

Send Workflow Notification

This policy is used to send a directed 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 properties can be defined for this policy:

Property Name	Description
Recipients(s)	Specifies the primary recipients. Separate multiple recipients with commas.
Group Recipient(s)	Specifies the primary group recipients. Separate multiple group recipients with commas.
Description	Specifies the description or e-mail subject for the notification.
Template	Specifies the target notification template to use. The default template is 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, you can click  to display the list of all of the registered directives.
Notification Variables	Specifies one or more element 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 element path values with commas.

Property Name	Description
HTTP Parameters	Specifies one or more element 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 element path values with commas.
Expiry	
Action on Expiry	Specifies the option to perform if the workflow is not completed by the date specified by the Expiration Date Data Object property. The available expiry options are as follows: None no expiration action for this notification. <i>Note:</i> The Expiration Date Data Object property 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 workflow instance using the specified template.
Expiration Date Data Object	Specifies the data object that defines the expiration date of the policy. <i>Note:</i> This property is available only when the Action on Expiry property is set to a value other than None.
Expiry Recipients(s)	Specifies the users to whom notifications are sent when the policy reaches its expiration date. Separate multiple recipients with commas. <i>Note:</i> This property is available only when the Action on Expiry property is set to Reroute.
Expiry Group Recipients(s)	Specifies the groups to which notifications are sent when the policy reaches its expiration date. Separate multiple group recipients with commas. <i>Note:</i> This property is available only when the Action on Expiry property is set to Reroute.

Property Name	Description
Workflow	Specifies the workflow template to start when the policy reaches its expiration date. If you are logged on to the SAS platform, you can click  to display the list of activated templates. <i>Note:</i> This property is available only when the Action on Expiry property is set to Start Workflow.

All policy properties 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 56. The **Expiration Date Data Object** property is an element 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** property 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** property. 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 Multiple Participants

The Set Multiple Participants policy is used to add multiple participants, which specify access control information for the specified (target) task or subflow.

The following properties can be defined for this policy:

Property Name	Description
Workflow	Specifies the element path for the target task or subflow.
Role	Specifies the workflow role to be copied from the source task.
Names Data Object	Specifies the data object that defines the platform identity names of the participants.
Delimiter	Specifies the delimiter to be used if multiple values are specified for the Names Data Object property.

Note: In order for the policy to execute, it must be defined in the same template where the data object is defined.

Set Overdue Status

The Set Overdue Status policy is used to add the Overdue status to a task or subflow.

The following properties can be defined for this policy:

Property Name	Description
Task	Specifies the element path for the target task or subflow.

Set Participant

This policy is used to set a participant, which specifies access control information for the specified (target) task or subflow.

The following properties can be defined for this policy:

Property Name	Description
Task	Specifies the element path for the target task or subflow.
Role	Specifies the workflow role for the participant.
Type	Specifies the platform identity type: User, Group, Role, or Privilege.
Name Data Object	Specifies the data object that defines the platform identity name of the participant.

Note: In order for the policy to execute, it must be defined in the same template where the data object is defined.

Start Workflow

The Start Workflow policy is used to start a task or subflow.

The following properties can be defined for this policy:

Property Name	Description
Task	Specifies the element path for the target task or subflow.

Start Workflow with Label

The Start Workflow with Label policy is used to start a separate (external) workflow instance.

The following properties can be defined for this policy:

Property Name	Description
New Workflow Label Data Object	Specifies the element path to a data object that contains the name of the workflow template to start.
New Workflow ID Data Object	Specifies the element path to a data object that holds the instance ID of the created workflow instance. This data object must exist in the current (parent) instance.

Property Name	Description
Current Workflow ID Data Object	Specifies the element path to a data object that contains the instance ID of the parent task executing this policy. This data object must exist in the newly created child instance.

When the policy executes, the workflow instance is started. The instance ID of the newly started (child) instance is stored in the New Workflow ID data object of the existing instance (parent). In addition, the policy execution looks for the Current Workflow ID data object in the child instance and stores the instance ID of the parent task associated with the policy.

Stop Workflow

The Stop Workflow policy is used to stop a task or subflow.

The following properties can be defined for this policy:

Property Name	Description
Task	Specifies the element path for the target task or subflow.

Stop Workflow with ID

The Stop Workflow with ID policy is used to stop an external workflow instance as defined by the identifier that is stored in a data object.

The following properties can be defined for this policy:

Property Name	Description
Workflow ID Data Object	Specifies the element path to a data object with the ID value of the target instance.

Submit a JES Job

The Submit a JES Job policy enables you to execute code that has been registered with the SAS Job Execution Service and resides on the server. A single job can execute a single task or multiple tasks bundled together. The job is executed via SAS Job Execution Service. For more information, see *SAS Intelligence Platform: Middle-Tier Administration Guide*.

The following properties can be defined for this policy:

Property Name	Description
Job Name	Specifies the name of the JES job. You must be logged on to the SAS platform to select a Job Name. Click  to display the list of all of the registered JES jobs.

Property Name	Description
Error Code	Specifies the data object (Short text type) for storage for the code value in the case of an error. If you are logged on to the SAS platform, you can click  to display a list of the existing Short text data objects.
Error Message	Specifies the data object (text type) for storage for the message value in the case of an error. If you are logged on to the SAS platform, you can click  to display a list of the existing Text data objects.

After you have selected the Job Name, SAS Workflow Studio displays the list of tasks with Input and Output fields for each task. You can then map data objects to parameters and results for each of these tasks.

Working with Data Objects

Overview

Data objects are similar to variables in a computer program. They hold pieces of business data required by the workflow. You can add global data objects that are available for use by the entire workflow or local data objects at the task level. Local data objects are accessible only for use by the parent task and its children.

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 workflow templates with less coupling, which minimizes impact from potential application data model changes.

Note: Data Object Updated events are generated when the value of the data object changes at run time. Policies triggered by a data object update must be defined at the same level or above the affected data object. See [“Policy Events and Scope” on page 34](#) for more information.

Adding Data Objects

To define a new data object, follow these steps:

1. In the workflow tree, right-click the top-level or local data object folder in the workflow tree and then select **Add Data Object**.

Alternatively, right-click a task 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. It is recommended that all of the data objects in your workflow have unique labels to help avoid confusion.
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

Note: The following data types are no longer supported:

- Check Box
 - Database Object
 - File
 - Organizational Role
 - Picklist
 - User
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 56](#).
 7. (Optional) Add custom attributes.
 For more information, see [“Custom Attributes” on page 56](#).
 8. Select **OK** to save the data object definition.

Editing Data Objects

To edit an existing data object, follow these steps:

1. In the workflow 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 workflow tree and then select **Delete**.
 Alternatively, select the data object directly in the workflow 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 a task, drag the data object from the workflow tree to the relevant task 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 task in the workflow tree.

Additional Workflow Features

Text Localization

SAS Workflow Studio supports localization of name and description text fields for workflows, data objects, statuses, and tasks. 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 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 template.

Custom Attributes

Custom attributes are application-specific values that are not related to the workflow business data. You can define custom attributes for workflows, data objects, statuses, and tasks. 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: Extensive use of application-specific attributes is not recommended to minimize coupling thereby maximizing workflow agility.

Data Object Substitution

As noted, some policy properties 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 property. The variable syntax used to indicate this value substitution is `${relative-element-path}`. SAS Workflow Studio replaces all occurrences with the value contained in the data object specified by relative element 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 Workflow 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 workflow template based on the application or organization that uses that workflow. You can use these tags to filter the list of templates in the Manage Workflows dialog box. (See [“Template Management” on page 78](#).)

- Tags are static labels. Existing tags can be modified using SAS Workflow Studio only. Tags have no effect on active workflow instances and cannot be modified at run time.
- Tags are not used for access control. They do not prevent the sharing of workflow templates among different applications. They do not affect whether or how a workflow 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.

Note: You must be logged on to the SAS server before you can work with tags.

Managing Tags through the Template Properties Dialog Box

Through the Template 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 template at a time.

To manage tags for a specific template through the Template 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 73](#).
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 [“Template Management” on page 78](#).
3. Open the Template Properties dialog box in one of the following ways:
 - Select **File** ⇒ **Properties**.

- Right-click on the root node of a template in the workflow tree and select **Edit**.
 - Select the root node in the workflow tree, and select **Edit** ⇒ **Edit properties**.
 - Double click on the root node in the workflow tree.
4. 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 name to the list of available tags, click **Add**, enter the new tag, and click **OK**.
 - 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.
 5. Click **OK** to save your changes and to close the Template 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 add new tags to the list of available tags using 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 73](#).
2. Select **Server** ⇒ **Manage Templates**.
3. Select the templates that you want to modify.
4. To add new tags to the list of available tags:
 - a. Click **New Tags**.
 - b. For each new tag, enter the new tag name in the **New Tags** field, and click **Add**.
 - c. Click **Save** to save your changes.
5. To change the tags that are associated with the selected templates:
 - a. Click **Select Tags**.
 - b. As needed, modify the tags associated with the selected templates.
 - To associate a tag with the selected templates, select the check box next to that tag.
 - To disassociate a tag from the templates, clear the check box next to that tag.
 - c. Click **OK** to save your changes.
6. Click **Close** to close the Manage Templates dialog box.

Chapter 5

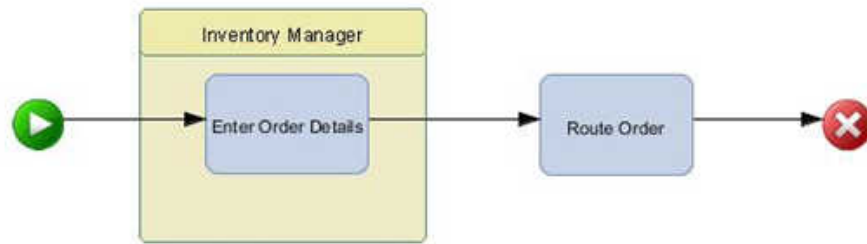
Advanced Topics

Workflow Patterns	59
Overview	59
Defining Alternate Paths	60
Defining Parallel Paths	64
Defining Convergent Paths	65
Additional Information about Workflow Patterns	67
Using Timers	68
Overview of Timers	68
Defining Timer-Triggered Actions	68
Adding Timer Tools to a Workflow	68
Controlling When Timers Start and Stop	68
Timer Tool Placement and Execution	69
Specifying Timer Settings (Schedule Expressions)	70
Deploying and Maintaining Workflows	72
Overview	72
Saving a Workflow Template	73
Opening a Workflow Template	73
Logging On to the Server	73
Managing Workflow Template Authorization	74
Versioning Workflow Templates	76
Template Management	78
SAS Alert Notification Templates	79
Editing Alert Notification Templates	79
Adding Fields to Alert Notification Templates	79
Configuring the Subject for an Alert Notification	79

Workflow Patterns

Overview

Only the most basic workflows can be represented as a single, sequential flow (Workflow Pattern: Sequence). More realistic workflows generally contain varying combinations of tasks, and a path through the tasks is based on business decisions derived from the business data. The path is determined at run time by specific outcomes from each task or by expression evaluation at decision points.

Figure 5.1 Sequential Workflow Example

Another common flow pattern is parallel processing, when two or more workflow 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

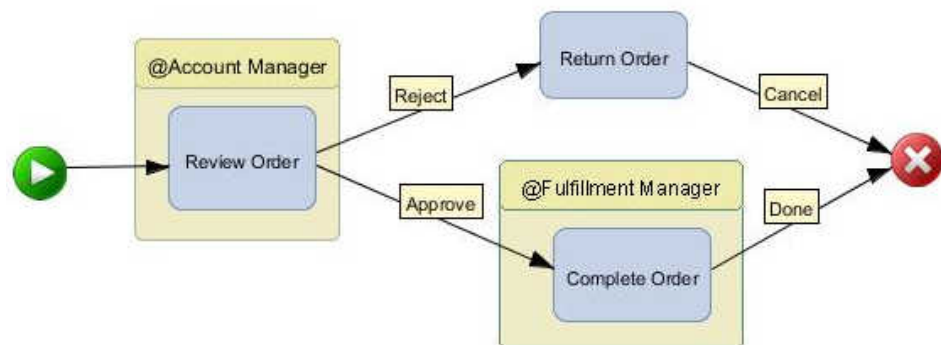
Overview

Many workflows contain multiple paths where each path represents one potential case or workflow 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 workflow with two possible outcomes: approved or denied. You can define two status values: Approve, and Reject. 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 workflow continues to the next task 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 115](#) for a detailed example.

Note: Most SAS products that leverage workflow use status values for every transition. Without status assignments, the task execution is controlled purely by sequence flow—the order in which elements are connected—and gateways. SAS Workflow Studio allows significant flexibility for defining workflow flow logic, so workflow 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 workflow execution 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 workflow 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 89](#) 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 workflow.

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

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

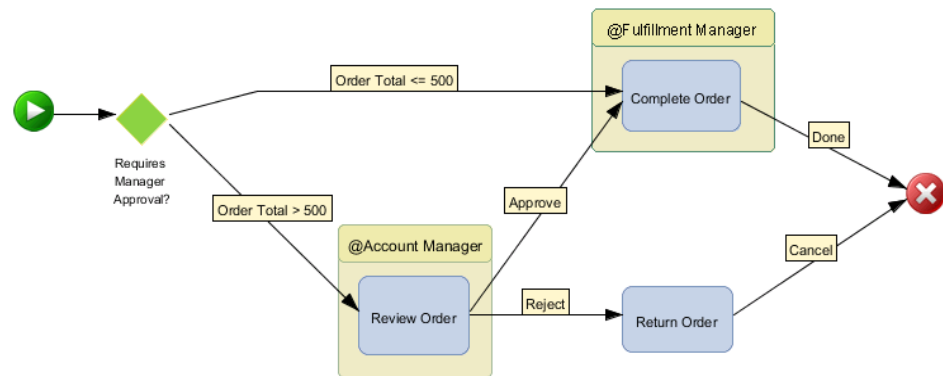
1. Ensure that you have defined all the required Boolean expressions for the decision gateway.
2. Define the necessary workflow execution paths.
3. Connect the decision to the relevant tasks for each path.
4. For each connection, right-click on the connection, select **Change status**, and select the 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.

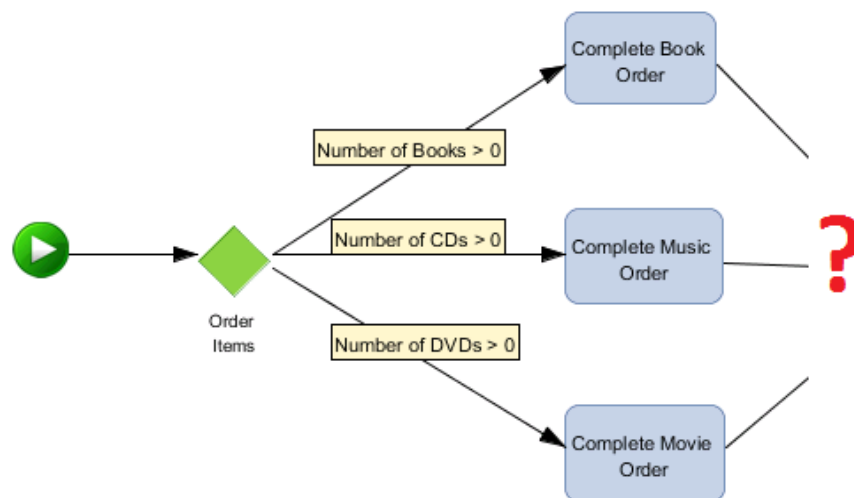
For the following example, any order total that exceeds \$500 requires manager approval before it is fulfilled. The decision gateway is based on an Order Total data object of type Number where the **Yes** path corresponds to a value exceeding the threshold and **No**, where the order total is below the 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 workflow 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 tasks 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, a workflow consists of a sequence of tasks where each task is activated upon completion of the preceding task. At other times, a workflow might require multiple tasks or paths to be initiated in parallel after a specific task has completed. SAS Workflow Studio supports parallel paths via the merge/fork gateway element.

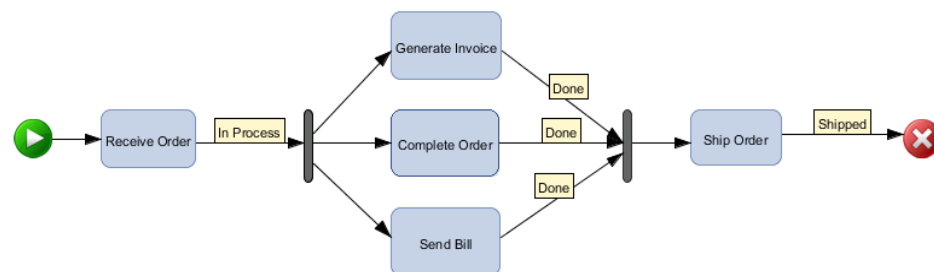
Adding Merge/Fork Gateways

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 task structure and connect to the merge/fork gateway.
4. On the right side, create the desired task set and create a sequence flow line from the merge/fork gateway figure to each task.

If more than one task exists before the bar, then all tasks must complete before any tasks on the other side are initiated.

Here is an example of a workflow that splits into three parallel tasks (Generate Invoice, Complete Order, Send Bill). The tasks start only after the Receive Order task has completed.

Figure 5.6 Example of Parallel Paths

See [Appendix 5, “Basic Workflow Examples,”](#) on page 115 for a more detailed example using parallel paths.

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 task is triggered after the current task preceding it is completed.
Exclusive Choice	Decision Gateway or unique Status values	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 paths eventually converge 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 a task. In other words, any policies associated with the convergent task are executed multiple times if the logical gateway is not used.

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.

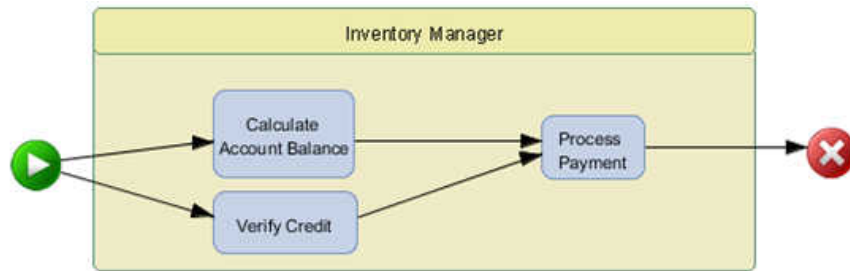
Note: The gateway is AND by default.

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 task set and connect each path to the logic gateway.
5. On the right side, draw the desired task structure and create a sequence flow line from the logic gateway figure to the first task in the converged path.

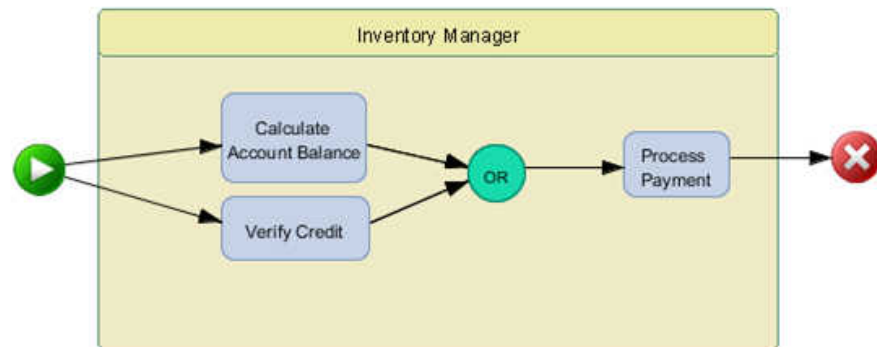
The following example results in only a single work item (task instance) for Process Payment after both Calculate Account Balance and Verify Credit have been completed. However, any actions associated with the Process Payment task are executed twice.

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.

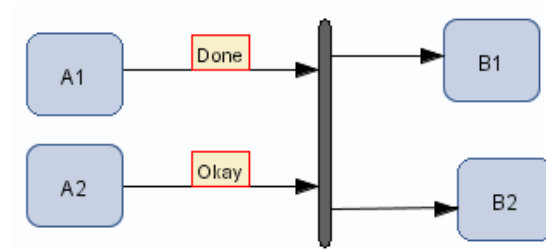
Figure 5.8 Example of Exclusive Merge (XOR)



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 workflow in [Figure 5.6 on page 64](#) illustrates the use of a merge/fork gateway. In that example, 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 task exists after the gateway, then the tasks are started together and run independently. The following example shows a workflow where a set of tasks (B1, B2) starts only after another set of tasks (A1, A2) finishes:

Figure 5.9 Example of a Complex Parallel Workflow

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.

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 each input is 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 about Workflow Patterns

For more information about workflow patterns, see the [Workflow Patterns Initiative](http://www.workflowpatterns.com) website. 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 of Timers

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

- start a task or workflow
- stop a task or workflow
- automatically traverse a transition
- deliver a notification

Defining Timer-Triggered Actions

There are two ways to define timer-triggered actions:

- by defining a timer tool. A timer tool enables you to start or stop a task or workflow based on its position in the template. See [“Adding Timer Tools to a Workflow” on page 68](#) and [“Timer Tool Placement and Execution” on page 69](#) for more information.
- by defining a policy that is triggered by a timer. A policy can execute any of the actions described in [“Policy Actions” on page 34](#). To define a policy that starts when a timer fires, define a policy that is triggered by a Timer Expired event. See [“Policy Events” on page 34](#) and [“Timer Expired and Tool Policy Example” on page 111](#) for more information. Policy-based timers start when the task that contains the timer starts.

Adding Timer Tools to a Workflow

To add a timer to a workflow, click  in the toolbar, and then click in the workflow diagram. SAS Workflow Studio automatically adds a Timer Tool policy to the associated task in the workflow tree.

Note: To define a policy that starts when a timer fires, define a policy that is triggered by a Timer Expired event.

Controlling When Timers Start and Stop

Timers start when the task that contains the timer is started or, in the case of stand-alone timers ([“Outside a Task \(Stand-alone Timer Tools\)” on page 70](#)), when the workflow containing the timer is started.

The timer expression that you enter in the Timer Settings dialog box controls when the timer fires (stops). You can enter fire times using Date data objects, relative offsets, or cron expressions. With Date data objects, the fire times can be defined at run time. You can specify both a Date data object and a relative offset. For the **Start date** firing time, you can specify either a positive or negative offset. For the **End date** firing time, you

can specify positive offsets. See “[Specifying Timer Settings \(Schedule Expressions\)](#)” on [page 70](#) for more information.

All outgoing connections from a timer are traversed each time the timer fires.

Timer execution varies depending on its placement in the workflow diagram. See “[Timer Tool Placement and Execution](#)” on [page 69](#) for more information.

Note: Policy-based timers start when the task that contains the timer starts.

Timer Tool Placement and Execution

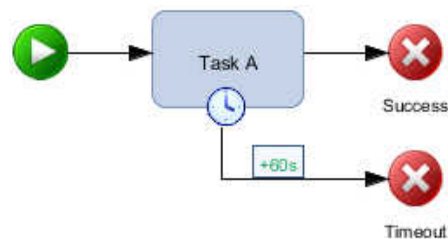
Timer Tool on the Border of a Task

A timer tool that is placed on the border of a task stops the task when the timer fires.

In the following example, the timer starts when Task A starts. The timer expression is +60s, which means that the timer is set to fire 60 seconds after it starts.

If Task A completes before the timer fires, then the timer is stopped, and the transition to the **Success** Stop node is traversed. The workflow instance is stopped with a status of **Success**.

If Task A has not completed before the timer fires, then Task A is stopped, and the transition to the **Timeout** Stop node is traversed. The workflow instance is terminated with a status of **Timeout**.

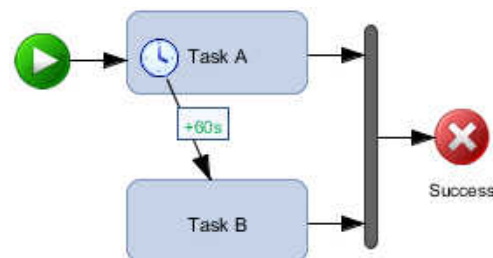


Timer Tool inside the Border of a Task

A timer tool that is placed inside the border of a task executes an associated action while the task remains active.

In the following example, the timer starts when Task A starts. The timer expression is +60s, which means that the timer is set to fire 60 seconds after it starts.

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



Outside a Task (Stand-alone Timer Tools)

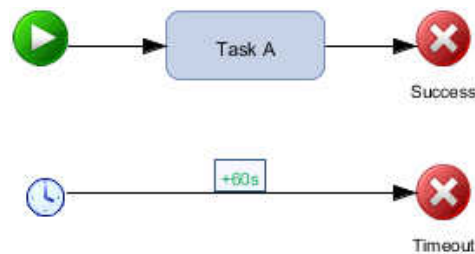
A timer tool that is placed outside a task executes an associated action.

All transitions leading from a stand-alone timer share the same timer interval. You cannot specify different timer intervals for different transitions from the timer. If this behavior is desired, then a separate timer is required for each interval.

In the following example, the timer is placed in the main workflow, so it starts when the workflow instance starts. The timer expression is +60s, which means that the timer is set to fire 60 seconds after it starts.

The timer fires 60 seconds after the workflow instance starts. If Task A has not completed when the timer fires, then the transition to the **Timeout** Stop node is traversed. The workflow instance is stopped with a status of **Timeout**.

If Task A completes before the timer fires, then the timer is stopped, and the transition to the **Success** Stop node is traversed. The workflow instance is stopped with a status of **Success**.



Specifying Timer Settings (Schedule Expressions)

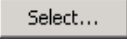
The settings in the Start date section of the Timer Settings dialog box are used to specify the initial time at which the timer fires. The time value can be a Date data object, relative expression, or a cron expression. The End date settings specify when a recurring timer stops firing.

To edit a timer's settings, double-click on the timer element in the workflow diagram. Alternatively, right-click the Timer Tool policy in the workflow diagram, and select **Edit**. For policy-based timers, right-click on the policy in the workflow tree, and select **Edit**.

Note: If the schedule expression has already passed, the timer fires immediately when the associated task or workflow starts.

Timer Setting	Description
Start Date Settings	
Date	Specifies a Date data object for initial date and time at which the timer fires. Click Select... to select the data object or to define a new one. You can also specify a relative offset in addition to Date data object. With Date data objects, after the timer has started, changes to the data object do not have any effect. Date data objects are not evaluated until the task or workflow containing the timer has started.

Timer Setting	Description
Time value	Specifies a relative time offset for initial time at which the timer fires. The offset is based on when the task that contains the timer starts. Specify the offset with either a plus or minus sign followed by an integer (for example +30 or -5). A negative offset (minus expression) means that the timer fires earlier than the value specified in the Date data object in the Date field. <i>Note:</i> You can enter a negative offset only if you have specified a Date data object.
Time unit	Specifies a time unit for the offset specified in the Time value field. Select Seconds, Minutes, Hours, Days, or Weeks.
Specify cron expression	Specifies a cron expression for initial time at which the timer fires. Relative expressions are based on when the task that contains the timer starts. Cron expressions are especially convenient if you want the timer to expire based on calendar information instead of exact dates or intervals. See the <code>org.quartz.CronExpression</code> documentation at http://quartz-scheduler.org/documentation for information about specifying cron expressions. <i>Note:</i> If you specify a cron expression, all other fields are ignored except for the end date in the Specify date and/or offset field. The syntax for cron expressions implicitly supports repeated firings (recurrence) of the timer object.
This event occurs only one time	Specifies that the timer fires only once.
Recurrence Settings	
Recurrence interval	Is a relative expression for the time interval between repeated executions. By default, timers fire only once. Specify an integer (for example 30).
Time unit	Specifies a time unit for the Recurrence interval. Select Seconds, Minutes, Hours, Days, or Weeks.
End date	Specifies when a recurring timer stops firing.
No end date	Repeats until the associated task or subflow stops.
End after	Repeats until the specified the number of occurrences has elapsed.

Timer Setting	Description
Specify date and/or offset	Repeats until the associated date or offset occurs (or both). You can specify a Date data object, a time offset, or both. - To specify a date, specify a Date data object in the Date field. Click  to select the data object or to define a new one. - To specify an offset, specify the offset in the Offset time value field and the time unit in the Time unit field. The offset is based on when the parent element starts. Specify the offset with a plus sign followed by an integer (for example +30). <i>Note:</i> You cannot specify negative offsets (minus expressions) for the Offset time value. <i>Note:</i> If you specify a cron expression in the Specify cron expression field, the Offset time value field is ignored.

See [Appendix 4, “Timer Examples,” on page 109](#) for examples of defining timer expressions, including cron syntax.

Deploying and Maintaining Workflows

Overview

Workflow 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 a specific template version stored in the workflow repository. Activation results in uploading the specified version as a formal workflow definition into the SAS Workflow Services for instantiation by end-user client applications. See [“Workflow Lifecycle” on page 7](#) for more details.

Note: Only one version of a workflow definition is active at a time. Workflow template changes made in SAS Workflow Studio do not take effect until the new version is uploaded and activated. After a new workflow is uploaded and activated on the workflow server, any new instances that are started use the newly activated version. Any currently running instances continue using their respective version of the workflow definition.

CAUTION:

It is strongly recommended that administrators do not delete workflow templates from the repository. Deleting a workflow template can cause unpredictable results when instance information is accessed and displayed using SAS Workflow Administrator.

Saving a Workflow Template

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

1. Select **File** ⇒ **Save** or **Save As**.
Alternatively, click the Save icon ()
2. Navigate to the desired directory in a dialog box.
3. Enter the filename for the workflow definition, or select an existing file to replace.
4. Select **Save** to close the dialog box and return to the diagram editor.

Opening a Workflow Template

To open a workflow template on 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 diagram editor.

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 workflow templates stored in the SAS content 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. The main system configuration file, *SASHome/sassw.config*, defines the environment variable `SASENVIRONMENTSURL`. This environment variable specifies the location of the environment configuration file that defines the host properties.

3. Enter your user name and password.
4. Select **Log On**.

Note: By default, the available host properties are configured in the `sas-environment.xml` file. This file is hosted by the SAS Web Server (`SASENVIRONMENTSURL=http://host_name:port/sas/sas-environment.xml`).

```
<environment name="serverName" default="false"
  platform-version="9.4">
  <desc>SAS Environment: serverName</desc>
  <block-desc>Default SAS Environment</block-desc>
  <service-registry>
```

```

    http://serverName:7980/SASWIPClientAccess/remote/ServiceRegistry
  </service-registry>
  <service-registry interface-type="soap">
    http://serverName:7980/SASWIPSoapServices/services/ServiceRegistry
  </service-registry>
</environment>

```

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

Managing Workflow Template Authorization

Workflow Roles and Privileges, and Template Permissions

In previous versions of SAS Workflow Studio, any SAS platform user could upload, download, or activate workflow templates. Beginning with SAS 9.4, administrators can do the following to control users' access to workflow templates:

- Assign users or groups to SAS Web Infrastructure Platform roles. Each role is associated with a set of privileges. These privileges determine which actions users can perform with regard to template creation and management.
- Grant users or groups specific permissions on a particular workflow template. These permissions supplement the access that is provided through role assignment.

Administrators can use the SAS Web Administration Console to assign roles to users and groups and to edit the privileges that are associated with a role. By default, there are three workflow template management roles. The following table lists the default roles and the privileges associated with each role.

Table 5.1 Workflow Roles and Privileges

Workflow Template Management Role	Default Workflow Privileges
Workflow Viewer	Read
Workflow Editor	Read, Edit, and Manage
Workflow Administrator	Read, Edit, Manage, and Activate

The following table lists the general abilities associated with each of the default workflow privileges.

Table 5.2 Workflow Privileges and Abilities

Privilege	Abilities
Read	Log on to the server. Get a local copy of a template.
Edit	All of the Read abilities. Plus: Edit local copies of templates, and save the changes locally.

Privilege	Abilities
Manage	All of the Read and Edit abilities. Plus: Check out templates, edit the templates, manage tags, and save (upload) the templates to the repository.
Activate	All of the Read, Edit, and Manage abilities. Plus: Activate an uploaded template. Plus: Assign template ownership to other users.

See *SAS Intelligence Platform: Middle-Tier Administration Guide* for more information about SAS Web Infrastructure Platform roles and privileges and the SAS Authorization Service.

Administrators must explicitly grant users access via privileges before the users can update, upload, and activate workflow templates to the content repository. To upload and activate templates, a user must be assigned the Workflow Editor role or have the relevant privileges. The template owner is the user that uploads the first version of a specific template. In SAS Workflow Studio, a template owner can manage access to individual templates by setting the permissions on the template through the Manage Access dialog box. The template owner can set the access for a specific user or group to Read, Update, or Administer. The following table lists the general abilities associated with each permission.

Table 5.3 *Template Permissions and Abilities*

Permission	Abilities
Read	Get a local copy of the template.
Update	Get a local copy of the template. Edit the local copy of template, and save the changes locally. Check out the template, edit the template, and save (upload) the template to the repository. Activate the uploaded template.
Administer	All of the Update abilities. Plus: Assign template permissions to users and groups. Plus: Assign template ownership to another user.

By default, the template owner has Administer permission for that template and is the only user that can change access to the template. To manage template access through the Manage Access dialog box, the template owner must be assigned the Workflow Administrator role and be logged on to the server.

Note: Template ownership cannot be assigned to groups.

Administrators can use SAS Web Administration Console to review the assigned permissions. Administrators can also update permissions using the SAS Web Administration Console, but you should do so only if instructed to do so by SAS Technical Support. For details, see “[Use the SAS Web Administration Console](#)” in *SAS Intelligence Platform: Middle-Tier Administration Guide*.

Managing Template Permissions

To update template access:

1. Log on to the server as the template owner.
2. Select **Server** ⇒ **Manage Templates** (F3).
3. Select the template, right-click, and select **Get Copy**. SAS Workflow Studio opens the most recent version of the template. The version number is shown in the upper left of the diagram editor.
4. Select **Server** ⇒ **Manage Access**.
5. (Optional) Add new users or groups. To add a new user or group, follow these steps:
 - a. Click **Add Users And Groups**.
 - b. Click **User** or **Group**, depending on which one you want to add.
 - c. Enter a search string, and click . You can use the asterisk (*) to list all users or groups. SAS Workflow Studio displays the list of users or groups matching the string that you entered.
 - d. Select the user or group that you want to add, and click **OK**.
6. Select the appropriate check boxes (**Read**, **Update**, or **Administer**) for each user or group.
7. Click **OK**.

For more information about permissions, see “[Manage Web-layer Permissions](#)” in *SAS Intelligence Platform: Middle-Tier Administration Guide*.

Versioning Workflow Templates

Uploading a Workflow Template

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

To upload a workflow template, follow these steps:

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

Alternatively, you can select  in the toolbar, or press F5.

Note: These options are not active until you log on to the server and require workflow editor or administrator access.

2. Enter relevant comments in the Save to 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 that 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 SAS Workflow Services.
4. If the definition has not been previously saved locally, then the Save dialog box appears. See “[Saving a Workflow Template](#)” on page 73 for details.

Downloading a Workflow Template

Downloading a workflow retrieves the workflow template from the content repository, so that it can be viewed or revised in SAS Workflow Studio.

To download a workflow template, follow these steps:

1. Select **Server** ⇒ **Open from Repository**.

Alternatively, you can select  in the toolbar or press F7.

Note: These options are not active until you log on to the server and require workflow View access.

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

2. Select the desired template.
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 workflow definition in the diagram editor.

Overriding a Workflow 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.

Note: These options are not active until you log on to the server and require workflow editor or administrator access.

Comparing Workflow Template Versions

The Compare Workflow Template Versions dialog box displays the workflow trees for two versions of a template side-by-side. This dialog box enables you to browse through the content of both workflow trees at the same time. It displays folder names in bold if the folders contain differences between the two versions. As you browse through the content of the workflow tree, the names and values of the workflow properties are displayed at the bottom of the dialog box.

To compare two versions of the same template, follow these steps:

1. Log on to the server.
2. Select **Server** ⇒ **Manage Templates** (F3).
3. Select the template, press the right mouse button, and select **Get Copy**. SAS Workflow Studio opens the most recent version of the template. The version number is shown in the upper left of the diagram editor.
4. Close the Manage Templates dialog box.

5. Select **Server** ⇒ **Compare Versions**. SAS Workflow Studio displays the list of versions for the template.
6. Select the version from the history that you want to compare with the current version, and click **Compare**. SAS Workflow Studio displays the workflow trees for the two versions side-by-side. Folders that contain differences are displayed in bold.
7. Click **Close** to close the Compare Workflow Template Versions and Version Information dialog boxes.

Template Management

SAS Workflow Studio includes the following workflow management functions from the Manage Templates 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 workflow 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 version information for each version of the selected template.

To access these template management functions, follow these steps:

1. Select **Server** ⇒ **Manage Templates**.

Alternatively, select the Manage Workflow Templates icon  or press F3.

The Manage Templates dialog box appears. The dialog box displays a list of templates 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.

CAUTION:

It is strongly recommended that administrators do not delete workflow templates from the repository. Deleting a workflow template can cause unpredictable results when instance information is accessed and displayed using SAS Workflow Administrator.

SAS Alert Notification Templates

Editing Alert Notification Templates

For information about editing notification templates and letterheads, see “[Manage Notification Templates and Letterheads](#)” in *SAS Intelligence Platform: Middle-Tier Administration Guide*.

Note: You should keep a backup copy of the alert notification templates.

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 95 for a detailed example that uses the SAS Template Service.

Appendix 1

Workflow Loader Utility

About the Workflow Loader Utility	81
Dictionary	81
Workflow Loader Utility for SAS 9.4	81
Workflow Loader Utility for SAS 9.3	86

About the Workflow Loader Utility

The workflow loader utility enables you to load existing workflow templates directly into the content repository from the command line. Beginning with SAS 9.3, SAS Workflow supports versioning of processes. Versioned process templates are stored in content server using content services. Beginning with SAS 9.4, the workflow loader utility use the Service Registry instead of multicast to connect to the middle tier.

Dictionary

Workflow Loader Utility for SAS 9.4

Loads existing workflow templates (a single file or a directory of templates) directly into the content repository. The utility can also activate the uploaded templates.

Syntax

```
java
-Dhost=hostname
-Dport=nnnn
-Dprotocol=protocol
com.sas.workflow.util.checkin.WorkflowLoader
user-ID
password
path
-outputDir directory-name
<-replace>
<-activate>
```

Required Arguments

-Dhost=*hostname*

is the host name of the server where the SAS middle tier is deployed.

-Dport=*nnnn*

is the port number of the SAS middle tier.

-Dprotocol=*protocol*

is the protocol used by the middle tier. You must specify this property if the middle tier is configured to use HTTPS. The default protocol is HTTP.

Note This parameter is supported by the first maintenance release of SAS 9.4 and later releases.

-outputDir *directory-name*

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 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.

password

is the password for the specified user.

path

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

$$user-ID$$

is the ID of a user who is authorized to load content. See “Managing Workflow Template Authorization” on page 74 for more information.

Optional Arguments

-activate

activates the template after it is loaded into the repository. If you specify this option, the workflow loader utility uploads the workflow definition to the run-time platform.

-replace

replaces the activated workflow definitions if they already exist in the platform. In the SAS Content Server, this option updates the template version.

Examples

Example 1: BAT File for SAS 9.4 on Windows

Before running this utility, set the path to the local environment file, the user ID, the password, and the port number to the correct values for your site.

```
@echo off
```

```

: /*-----\
: | Script for executing the WorkflowLoader Utility |
: |                                               |
: \-----*/

```

setlocal

```

REM Update the following path to your local_env.bat file

call "%dp0%\..\level_env.bat"

REM Define needed environment variables

set SASINSTALLHOME=C:\Program Files\SASHome
set USERID=user
set PASSWORD=password
set PORT=port

if NOT "%1"==" " set WFTEMPLATE=%1

if "%1"==" " echo "No template argument supplied"

: Get our hostname value
FOR /F "usebackq" %i in (`hostname`) DO SET HOST=%i
set HOST=%HOST%.domain.com

set LAUNCHERJAR=%SASVJR_HOME%\eclipse\plugins\sas.launcher.jar

set UTILITIESDIR=%LEVEL_ROOT%\Web\Utilities
set PICKLISTS=%SASINSTALLHOME%\SASWebInfrastructurePlatform\9.4\Picklists\wars\sas.workflow\**\picklist
set CLASSPATH=%UTILITIESDIR%;%LAUNCHERJAR%

REM Run the query test
"%JAVA_JRE_COMMAND%" ^
-classpath "%CLASSPATH%" ^
-Djava.system.class.loader=com.sas.app.AppClassLoader -Dsas.app.launch.config="%PICKLISTS%" ^
-Dsas.app.repository.path="%SASVJR_REPOSITORYPATH%" ^
-Dsas.app.class.path="%UTILITIESDIR%" ^
-Djava.security.auth.login.config=..\Common\login.config^
-Djava.net.preferIPv4Stack=true -Djava.net.preferIPv6Addresses=false ^
-Xms64m -Xmx64m -Dhost=%HOST% -Dport=%PORT% ^
com.sas.workflow.util.checkin.WorkflowLoader %USERID% %PASSWORD% -verify
%*

REM Run the upload test
"%JAVA_JRE_COMMAND%" ^
-classpath "%CLASSPATH%" ^
-Djava.system.class.loader=com.sas.app.AppClassLoader -Dsas.app.launch.config="%PICKLISTS%" ^
-Dsas.app.repository.path="%SASVJR_REPOSITORYPATH%" ^
-Dsas.app.class.path="%UTILITIESDIR%" ^
-Djava.security.auth.login.config=..\Common\login.config^
-Djava.net.preferIPv4Stack=true -Djava.net.preferIPv6Addresses=false ^
-Xms64m -Xmx64m -Dhost=%HOST% -Dport=%PORT% ^
com.sas.workflow.util.checkin.WorkflowLoader %USERID% %PASSWORD% %WFTEMPLATE% -outputDir "%UTILITIESDIR%" -replace -activate
%*

endlocal
if [%2] EQU [exit] exit %ERRORLEVEL%

```

Example 2: Shell Script for SAS 9.4 on Linux

Before running this utility, set the path to the local environment file, the user ID, the password, the host, and the port to the correct values for your site.

```
#!/bin/sh -p
#
# WorkflowLoader.sh
#
# Update the following path to your local_env.sh script
#
. `dirname $0`/../../level_env.sh

# Define environment variables

SASINSTALLHOME=/install/cfgsas1/SASHome
USERID=user
PASSWORD=password
HOST=host_name
PORT=port

if [ -z "$1" ]
then
    echo "No template argument supplied"
else
    WFTEMPLATE=$1
fi

LAUNCHERJAR=$SASVJR_HOME/eclipse/plugins/sas.launcher.jar

UTILITIESDIR=$LEVEL_ROOT/Web/Utilities
PICKLISTS=$SASINSTALLHOME/SASWebInfrastructurePlatform/9.4/Picklists/wars/sas.workflow/**/picklist
CLASSPATH=$UTILITIESDIR:$LAUNCHERJAR

"$JAVA_JRE_COMMAND" \
-classpath "$CLASSPATH" \
-Djava.system.class.loader=com.sas.app.AppClassLoader -Dsas.app.launch.config="$PICKLISTS" \
-Dsas.app.repository.path="$SASVJR_REPOSITORYPATH" \
-Dsas.app.class.path="$UTILITIESDIR" \
-Dconfig.lev.web.appserver.logs.dir=/install/cfgsas1/config/Lev1/Web/Logs/SASServer1_1 \
-Djava.security.auth.login.config=../Common/login.config \
-Djava.net.preferIPv4Stack=true -Djava.net.preferIPv6Addresses=false \
-Dhost=$HOST -Dport=$PORT \
com.sas.workflow.util.checkin.WorkflowLoader $USERID $PASSWORD -verify

"$JAVA_JRE_COMMAND" \
-classpath "$CLASSPATH" \
-Djava.system.class.loader=com.sas.app.AppClassLoader -Dsas.app.launch.config="$PICKLISTS" \
-Dsas.app.repository.path="$SASVJR_REPOSITORYPATH" \
-Dsas.app.class.path="$UTILITIESDIR" \
-Dconfig.lev.web.appserver.logs.dir=/install/cfgsas1/config/Lev1/Web/Logs/SASServer1_1 \
-Djava.security.auth.login.config=../Common/login.config \
-Djava.net.preferIPv4Stack=true -Djava.net.preferIPv6Addresses=false \
-Dhost=$HOST -Dport=$PORT \
com.sas.workflow.util.checkin.WorkflowLoader $USERID $PASSWORD $WFTEMPLATE -outputDir "$UTILITIESDIR" -replace -activate

exit 0
```

Example 3: Shell Script for SAS 9.4 on AIX

Before running this utility, set the path to the local environment file, the user ID, the host, the port number, and the password to the correct values for you site.

```
#!/bin/sh -p
#
# WorkflowLoader.sh
#
# Update the following path to your local_env.sh script
#
. `dirname $0`/../../level_env.sh

# Define environment variables

SASINSTALLHOME=/install/cfgsas1/SASHome
USERID=user
PASSWORD=password
HOST=host_name
PORT=port

if [ -z "$1" ]
then
    echo "No template argument supplied"
else
    WFTEMPLATE=$1
fi

LAUNCHERJAR=$SASVJR_HOME/eclipse/plugins/sas.launcher.jar

UTILITIESDIR=$LEVEL_ROOT/Web/Utilities
PICKLISTS=$SASINSTALLHOME/SASWebInfrastructurePlatform/9.4/Picklists/wars/sas.workflow/**/picklist
CLASSPATH=$UTILITIESDIR:$LAUNCHERJAR

"$JAVA_JRE_COMMAND" \
-Dsas.ext.config="$SASINSTALLHOME/sas.java.ext.config" \
-classpath "$CLASSPATH" \
-Djava.system.class.loader=com.sas.app.AppClassLoader -Dsas.app.launch.config="$PICKLISTS" \
-Dsas.app.repository.path="$SASVJR_REPOSITORYPATH" \
-Dsas.app.class.path="$UTILITIESDIR" \
-Djava.security.auth.login.config=../Common/login.config \
-Djava.net.preferIPv4Stack=true -Djava.net.preferIPv6Addresses=false \
-Dhost=$HOST -Dport=$PORT \
com.sas.workflow.util.checkin.WorkflowLoader $USERID $PASSWORD -verify

"$JAVA_JRE_COMMAND" \
-Dsas.ext.config="$SASINSTALLHOME/sas.java.ext.config" \
-classpath "$CLASSPATH" \
-Djava.system.class.loader=com.sas.app.AppClassLoader -Dsas.app.launch.config="$PICKLISTS" \
-Dsas.app.repository.path="$SASVJR_REPOSITORYPATH" \
-Dsas.app.class.path="$UTILITIESDIR" \
-Djava.security.auth.login.config=../Common/login.config \
-Djava.net.preferIPv4Stack=true -Djava.net.preferIPv6Addresses=false \
-Dhost=$HOST -Dport=$PORT \
com.sas.workflow.util.checkin.WorkflowLoader $USERID $PASSWORD $WFTEMPLATE -outputDir "$UTILITIESDIR" -replace -activate
exit 0
```

Workflow Loader Utility for SAS 9.3

Loads existing workflow templates (a single file or a directory of templates) directly into the content repository. The utility can also activate the uploaded templates.

Syntax

```
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>
```

Required Arguments

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

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

-Dmulticast.port=nnnn

is the multicast port number of the SAS middle tier.

-outputDir directory-name

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 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.

password

is the password for the specified user.

path

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

user-ID

is the ID of a user who is authorized to load content. See [“Managing Workflow Template Authorization” on page 74](#) for more information.

Optional Arguments

-activate

activates the template after it is loaded into the repository. If you specify this option, the workflow loader utility uploads the workflow definition to the run-time platform.

-replace

replaces the activated workflow definitions if they already exist in the platform. In the SAS Content Server, this option updates the template version.

Example: BAT File for SAS 9.3 on Windows

Before running this utility, set the path to the local environment file, the user ID, and the password to the correct values for you site.

```
@echo off

: /*-----\
: ||| Script for executing the WorkflowLoader Utility      |
: |||                                                    |
: \-----*/

setlocal

REM Edit this path to point to the local environment file
call "%~dp0..\..\level_env.bat"

REM Define needed environment variables

set SASINSTALLHOME=C:\Program Files\SASHome
set USERID=user
set PASSWORD=password
set MULTICAST=239.21.16.242
set MPORT=31001
set MTTL=32

if NOT "%1"==" " set WFTEMPLATE=%1

if "%1"==" " set WFTEMPLATE="Basic Approval Process.xml"

set LAUNCHERJAR=%SASVJR_HOME%\eclipse\plugins\sas.launcher.jar

set UTILITIESDIR=%LEVEL_ROOT%\Web\Utilities
set PICKLISTS=%SASINSTALLHOME%\SASWebInfrastructurePlatform\9.3\Picklists\wars\sas.workflow\**\picklist
set CLASSPATH=%UTILITIESDIR%;%LAUNCHERJAR%

REM Run the query test
"%JAVA_JRE_COMMAND%" ^
-classpath "%CLASSPATH%" ^
-Djava.system.class.loader=com.sas.app.AppClassLoader -Dsas.app.launch.config="%PICKLISTS%" ^
-Dsas.app.repository.path="%SASVJR_REPOSITORYPATH%" ^
-Dsas.app.class.path="%UTILITIESDIR%" ^
-Djava.security.auth.login.config=..\Common\login.config^
-Djava.net.preferIPv4Stack=true -Djava.net.preferIPv6Addresses=false ^
-Xms64m -Xmx64m -Dmulticast.address=%MULTICAST% -Dmulticast.port=%MPORT% -Dmulticast_udp_ip_ttl=%MTTL% ^
com.sas.workflow.util.checkin.WorkflowLoader %USERID% %PASSWORD% -verify
%*

REM Run the upload test
"%JAVA_JRE_COMMAND%" ^
-classpath "%CLASSPATH%" ^
-Djava.system.class.loader=com.sas.app.AppClassLoader -Dsas.app.launch.config="%PICKLISTS%" ^
-Dsas.app.repository.path="%SASVJR_REPOSITORYPATH%" ^
-Dsas.app.class.path="%UTILITIESDIR%" ^
-Djava.security.auth.login.config=..\Common\login.config^
-Djava.net.preferIPv4Stack=true -Djava.net.preferIPv6Addresses=false ^
-Xms64m -Xmx64m -Dmulticast.address=%MULTICAST% -Dmulticast.port=%MPORT% -Dmulticast_udp_ip_ttl=%MTTL% ^
```

```
com.sas.workflow.util.checkin.WorkflowLoader %USERID% %PASSWORD% %WFTEMPLATE% -outputDir "%UTILITIESDIR%" -replace -activate
%*

endlocal
if [%2] EQU [exit] exit %ERRORLEVEL%
```

Appendix 2

Decision Expression Examples

Operator Support in Decision Expressions	89
Function Support in Decision Expressions	91
Decision Expression Examples	91
Example Data Objects	91
Example 1: Comparing the Value of a Data Object to a String Constant	92
Example 2: Comparing the Values of Multiple Data Objects to String Constants . .	92
Example 3: Comparing the Value of a Data Object to a Numeric Constant	92
Example 4: Comparing the Values of Two Different Data Objects	92
Example 5: Expression Using the UPCASE Function	92
Example 6: Expression Using Data Object Substitution	92
Example 7: Expressions Using TODAY and Date Functions	93

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

Operator	Operation
++	increment value by 1
--	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	Returns the time part of the data object value. The TIMEPART function resolves to milliseconds, so it is recommended that you use a comparison operator other than equals (==) with this function. <i>Note:</i> This function is supported only by Date data objects.
DATEPART	Returns the date part of the data object value. <i>Note:</i> This function is supported only by Date data objects.
TODAY	Returns the current system datetime value. <i>Note:</i> You can use this function to compare values with the results of the other functions described in this table. Doing so enables you to isolate specific numeric date or time values in a Date data object. This enables arithmetic evaluation of the specific numeric values.

Decision Expression Examples

Example Data Objects

For the following examples, the workflow 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
SalePrice	Numeric data object
ShipDate	Date data object (includes both date and time values)

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

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

Example 5: Expression Using the UPCASE Function

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

Example 6: Expression Using Data Object Substitution

This example uses the `${ }` syntax so that the complete path to the data object can be specified. (See [“Data Object Substitution” on page 56](#).) This is useful if a global data object defined on the root task and a local data object have the same name, and if 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 parent workflow.

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

Note: Data determination logic is supported to ensure backward 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.

Example 7: Expressions Using *TODAY* and Date Functions

The following example compares the date portion of the ShipDate data object with the date portion of today's date and time.

```
datepart(ShipDate) == today()
```

Because the left side of the example uses only the date part of ShipDate, only the date part of today's date is used in the comparison.

The following example compares the time portion of the ShipDate data object with the time portion of today's date and time. The TIMEPART function resolves to milliseconds, so it is recommended that you use a comparison operator other than equals (==) with this function.

```
timepart(ShipDate) >= today()
```

It is also recommended that you use a comparison operator other than equals (==) if you are comparing full datetime values.

```
ShipDate > today()
```


Appendix 3

Policy Usage Examples

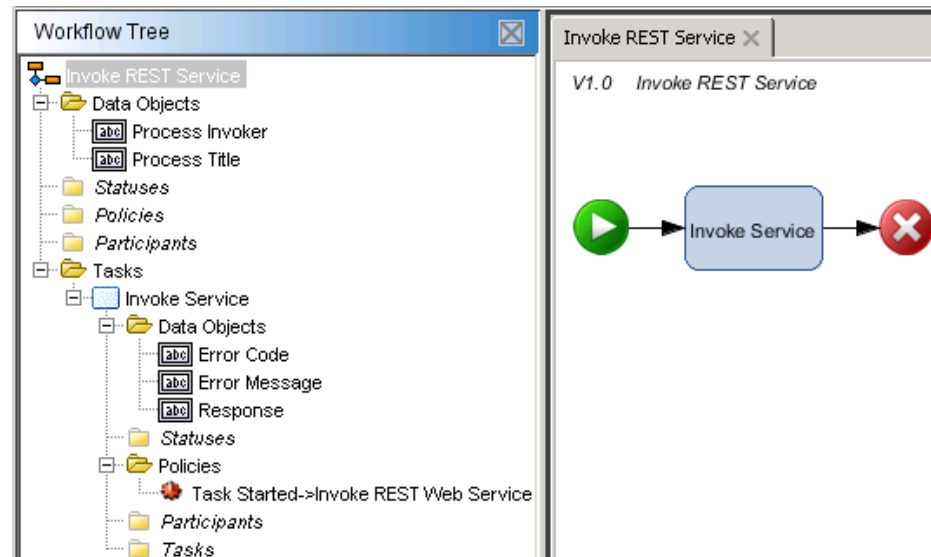
Invoke REST Web Service Policy Example	95
Invoke SAS Code Policy Example	98
Invoke Web Service Policy Example	100
Send Notification Using SAS Template Service Policy Example	103
Submit a JES Job Policy Example	105

Invoke REST Web Service Policy Example

This example illustrates a simple sequence with a single task. The task executes the Invoke REST Web Service policy, which sends a GET request to Google Maps for the geographical coordinates of a specific address. Because this example does not require a request string, it does not specify a request data object. The response is stored in output data object named Response, which is a text type object.

Note: This example invokes an external web service that might require proxy server configuration for your web application server.

The following figure shows the workflow tree and workflow diagram for this example. The workflow has task-level data objects for the response and error values.



The data objects in the Invoke REST Web Service task have the following properties:

Data Object Label	Type	Value
Response	Long Text Object	None (It is populated by the web service. If the expected response could exceed 4000, then use the Long Text data type.)
Error Code	Number	None (It is populated if an error occurs.)
Error Message	Short Text	None (It is populated if an error occurs.)

The policy definition associated with the task is as follows:

Policy Property	Value
Name	Task Started->Invoke REST Web Service
Event	Task Started
Action	Invoke REST Web Service
Description	This policy is used to invoke an unregistered web service function over REST.
Web Service Location	http://maps.googleapis.com/maps/api/geocode/json?sensor=false&address=500%20SAS%20Campus%20Drive%2C%20Cary%2C%20NC
HTTP Method	GET
Content Type	None (The default is text/plain.)
Web Service Input	None

Policy Property	Value
Web Service Output	None
Error Code	Invoke REST Web Service/Invoke Service/Error Code
Error Message	Invoke REST Web Service/Invoke Service/Error Message
User Name	None
Password	None

Note: The JSON syntax endpoint is specified, so the response is a JSON string.

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

```
{
  "results": [ {
    "address_components": [
      {
        "long_name": "500",
        "short_name": "500",
        "types": ["street_number"]
      },
      {
        "long_name": "Sas Campus Drive",
        "short_name": "Sas Campus Dr",
        "types": ["route"]
      },
      {
        "long_name": "Cary",
        "short_name": "Cary",
        "types": [
          "locality",
          "political"
        ]
      },
      {
        "long_name": "Cary",
        "short_name": "Cary",
        "types": [
          "administrative_area_level_3",
          "political"
        ]
      },
      {
        "long_name": "Wake",
        "short_name": "Wake",
        "types": [
          "administrative_area_level_2",
          "political"
        ]
      },
      {
        "long_name": "North Carolina",
```

```

        "short_name": "NC",
        "types": [
            "administrative_area_level_1",
            "political"
        ]
    },
    {
        "long_name": "United States",
        "short_name": "US",
        "types": [
            "country",
            "political"
        ]
    },
    {
        "long_name": "27513",
        "short_name": "27513",
        "types": ["postal_code"]
    }
],
"formatted_address": "500 Sas Campus Drive, Cary, NC 27513, USA",
"geometry": {
    "location": {
        "lat": 35.823976,
        "lng": -78.7579669
    },
    "location_type": "ROOFTOP",
    "viewport": {
        "northeast": {
            "lat": 35.8253249802915,
            "lng": -78.7566179197085
        },
        "southwest": {
            "lat": 35.8226270197085,
            "lng": -78.75931588029151
        }
    }
},
"types": ["street_address"]
}],
"status": "OK"
}

```

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`. Alternatively, you can use a file URL if it is local to the middle tier server(for example, `file:///temp/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 local because they are associated with the task:



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

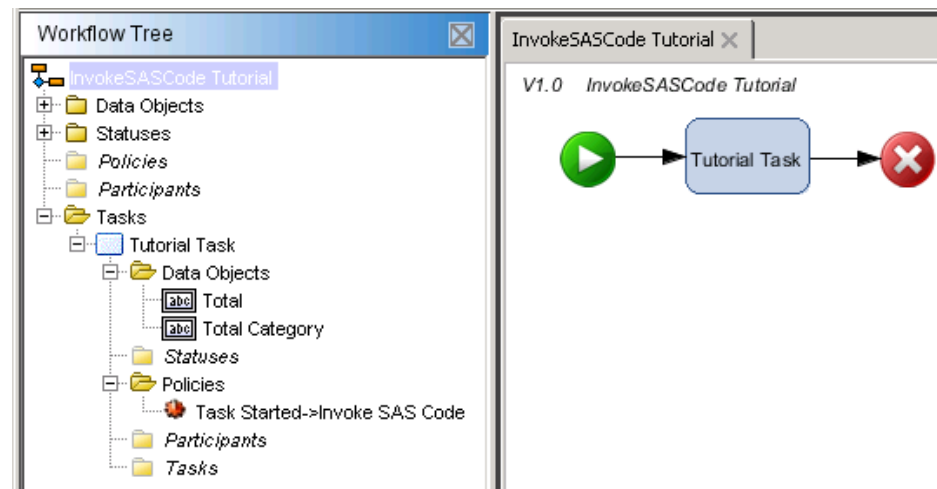
Next, associate the policy definition with the task.

Policy Property	Value
Name	Task Started->Invoke SAS Code
Description	This policy sends the var parameter value to the hosted SAS code example and assigns the result to the Total data object.
Event	Task Started
Action	Invoke SAS Code
SAS Source Code	<code>http://host-name/invokeSASCodeTutorial.sas</code>
Server	SASApp – Logical Workspace Server
Repository	Foundation
Pass all root workflow data objects	disabled

Policy Property	Value
(Output) Data Object	InvokeSASCode Tutorial/Tutorial Task/Total
(Input 1) Data Object	InvokeSASCode Tutorial/Tutorial Task/Total Category
(Input 1) Macro Variable	var
(Output) Macro Variable	result

See “Invoke SAS Code” on page 40 for more information.

The final workflow definition is as follows:



Multiple data objects can be passed to the SAS code via corresponding input macro variables. Because this example requires only one input variable, the following line of code is prepended to the SAS code before execution:

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

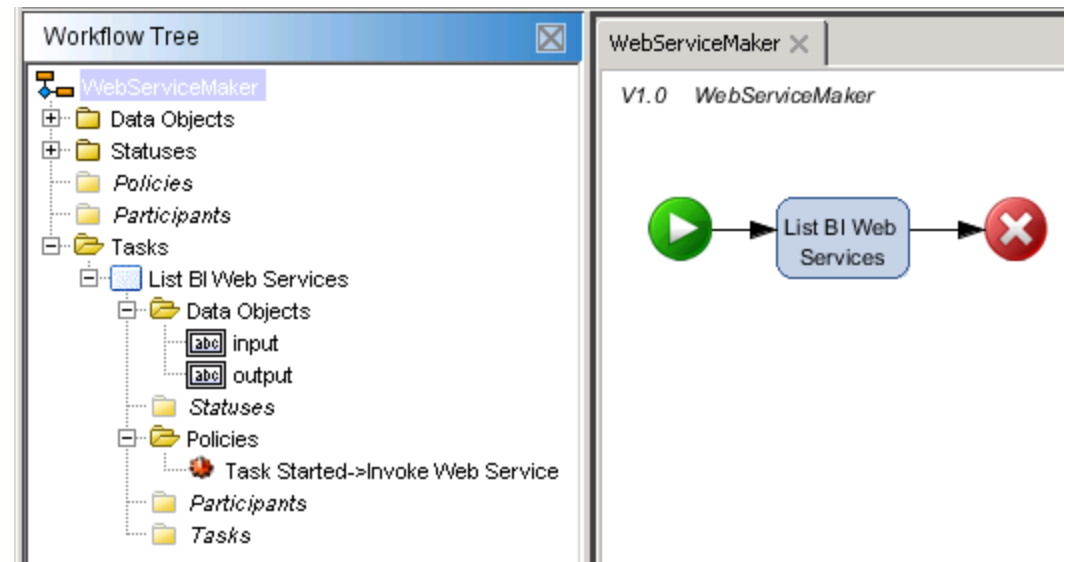
Only a single return variable is supported. See the examples provided with SAS Workflow Studio for details of this example.

Invoke Web Service Policy Example

This example illustrates a simple sequence with a single task. Once started, the task 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 task 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 the workflow diagram and workflow tree for a workflow where the policy data objects are local because they are associated with the task.



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

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

Both data objects use the following schema:

```
<schema elementFormDefault="qualified"
  targetNamespace="http://support.sas.com/xml/namespace/biwebservices/webservicemaker-9.2"
  xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:tns="http://support.sas.com/xml/namespace/biwebservices/webservicemaker-9.2"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <complexType name="StringArrayType">
    <sequence>
      <element maxOccurs="unbounded" minOccurs="1" name="string" type="string"/>
    </sequence>
  </complexType>
  <element name="MakeWebService">
    <complexType>
      <sequence>
        <element name="storedWorkflowPaths" type="tns:StringArrayType"/>
        <element name="serviceName" type="string"/>
        <element minOccurs="0" name="serviceNamespace" type="string"/>
        <element minOccurs="0" name="keywords" type="tns:StringArrayType"/>
      </sequence>
    </complexType>
  </element>
</schema>
```

```

        <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 task.

Policy Property	Value
Name	Task Started->Invoke Web Service
Event	Task Started

Policy Property	Value
Action (Policy)	Invoke Web Service
Description	This policy is used to invoke a web service function over SOAP/HTTP.
Web Service Location	http://localhost:8080/SASBIWS/services/WebServiceMaker
Web Service Action	http://support.sas.com/xml/namespace/biwebservices/webservicemaker-9.2/ListWebServices
Web Service Input	WebServiceMaker/List BI Web Services/input
Web Service Output	WebServiceMaker/List BI Web Services/output
User Name	<i>userid</i>
Password	<i>password</i>

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 a task starts or finishes by selecting a check box for the task. 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 E-mail

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 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: The format for this notification type can be customized using the SAS Template Service.

Note: Five expiry options are available: None, Remove, Reroute, Resend, and Start Workflow. For more information, see [“Send Workflow Group Notification” on page 46](#).

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.

Note: The format for this notification type can be customized using the SAS Template Service.

Note: Five expiry options are available: None, Remove, Reroute, Resend, and Start Workflow. For more information, see [“Send Workflow Notification” on page 49](#).

The Approval Escalation Workflow 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 properties are as follows:

Policy Property	Value
Recipient(s)	
Group Recipient(s)	<code>\${../Account Manager}</code>
Description	
Template	<code>SAS_Email_Message</code>
Directive	
Notification Variables	<code>../MSG, ../DESC</code>
HTTP Parameters	

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 56](#).

The default value for this data object is **US Region Accountants**, so any users assigned to this group in the SAS Authorization Service 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.

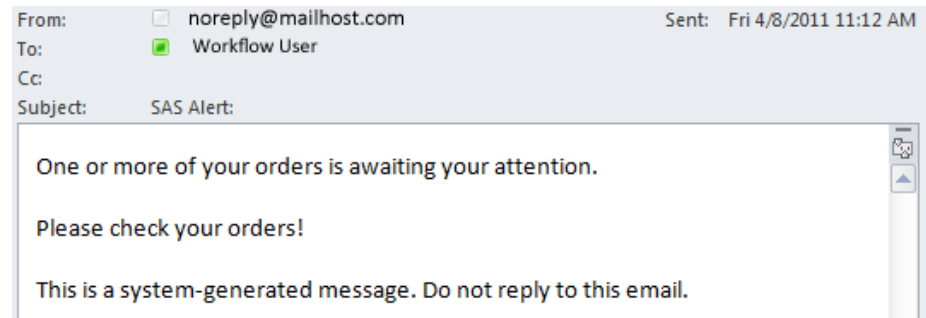
File A3.1 SAS_Email_Message Template

```

$if(_SAS_TEMPLATE_TXT)$
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:

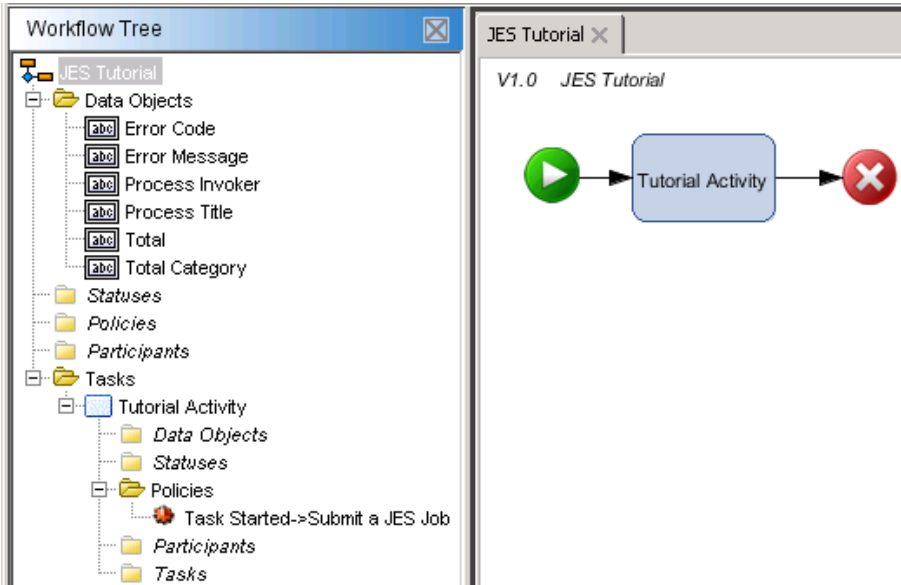


See the examples provided with SAS Workflow Studio for details of this example.

Submit a JES Job Policy Example

This example illustrates a simple workflow with a single task. The task executes the Submit a JES Job policy, which sends a single input parameter. The response is stored in the output data object named Total, which is a text type object.

The following figure shows the workflow tree and workflow diagram for this example. The workflow has root-level data objects for the input, response, and error value data objects.



The code used in “[Invoke SAS Code Policy Example](#)” on page 98 was registered in metadata as a stored process. In this example, a JES stored process task references that code. The data objects used by the Submit JES Job policy have the following properties:

Data Object Label	Type	Value
Total	Short Text	None (It is set by the policy.)
Total Category	Short Text	sales
Error Code	Short Text	None (It is populated if an error occurs.)
Error Message	Short Text	None (It is populated if an error occurs.)

The policy definition associated with the task is as follows:

Policy Property	Value
Name	Task Started->Submit a JES Job
Event	Task Started
Action	Submit a JES Job
Description	This policy is used to submit a JES Job.
Job Name	SAS Projects/MidTier Services/Workflow/TestSPJESPolicyJob
Input	JES Tutorial/Total Category
Output	JES Tutorial/Total
Error Code	JES Tutorial/Error Code

Policy Property	Value
Error Message	JES Tutorial/Error Message

The final response stored in the Total output data object is **33851566**.

Appendix 4

Timer Examples

Timer Expression Examples	109
Timer Expired and Tool Policy Example	111

Timer Expression Examples

The following example defines a timer that is triggered in one day and does not reoccur. The timer expression shown in the workflow diagram is **+1d**.

Timer Property	Value
Time value	+1
Time unit (for Time value)	Days
This even occurs only one time	selected

The following example defines a timer that fires every day at 12 p.m. (noon). The timer expression shown in the workflow diagram is the cron expression, **0 0 12 * * ?**.

Timer Property	Value
Specify cron expression	0 0 12 * * ?
No end date	selected

The following example defines a timer that is scheduled to fire one day after the workflow is started and repeats every 24 hours. No end date is specified, so the timer repeats indefinitely. The timer expression shown in the workflow diagram is **+1d**.

Timer Property	Value
Time value	+1

Timer Property	Value
Time unit (for Time value)	Days
Recurrence interval	+24
Time unit (for Recurrence interval)	Hours
No end date	selected

The following example defines a timer that is scheduled to fire 1 day after the workflow is started and repeats every 24 hours. The end date is specified as an offset of 1 week, so the timer terminates after 1 week. The timer expression shown in the workflow diagram is **+1d**.

Timer Property	Value
Time value	+1
Time unit (for Time value)	Days
Recurrence interval	+24
Time unit (for Recurrence interval)	Hours
Specify date and/or offset	selected
Offset time value	+1
Time unit (for Offset time value)	Weeks

In the following example, the start date is specified with a negative offset. The timer expression shown in the workflow diagram is **-3d**. In this example, the first time the timer fires is 3 days before the date specified by Date data object, so the timer starts firing on July 7, 2013 at 09:42:06 AM. The timer fires every 24 hours for 1 week and expires on July 14, 2013 at 09:42:06 AM.

Note: You can enter negative offsets for the starting **Time value**, but you cannot specify a negative offset for the end date **Offset time value**.

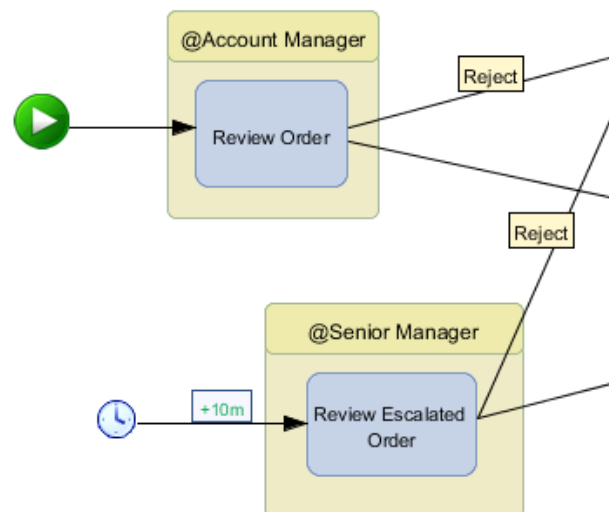
Timer Property	Value
Date	Approval Workflow/Initial Date , which contains the value July 10, 2013 09:42:06 AM
Time value	-3
Time unit (for Time value)	Days

Timer Property	Value
Recurrence interval	+24
Time unit (for Recurrence interval)	Hours
Specify date and/or offset	selected
Offset time value	+1
Time unit (for Offset time value)	Weeks

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. The users are notified by e-mail up to three times before the order is escalated and reassigned to a senior manager. The remaining workflow tasks are similar to those in [“Basic Approval Workflow”](#) on page 115.

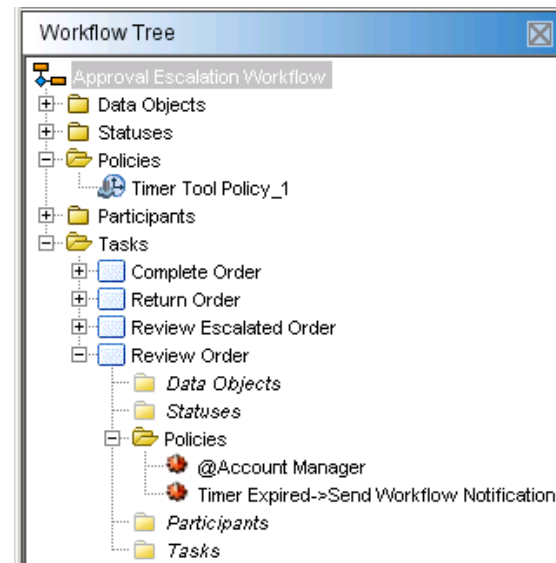
Figure A4.1 Timer Expired Tool



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

The following figure shows the workflow tree for the Approval Escalation workflow:

Figure A4.2 Approval Escalation Workflow Tree



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

Policy Property	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 Properties	
Action on Expiry	None
Policy Label	Timer Expired->Send Workflow Notification

The timer settings in the following table trigger an e-mail notification after three minutes and twice more after two minutes.

Timer Setting	Value
Time value	+3
Time unit (for Time value)	Minutes
Recurrence interval	+2
Time unit (for Recurrence interval)	Minutes
End after	2

The following table lists the timer settings for the timer policy associated with the main workflow. These settings trigger a transition to the Review Escalated Order task after 10 minutes. Another policy is associated with this task that stops the initial Review Order Task because the order has been automatically reassigned.

Timer Setting	Value
Time value	+10
Time unit (for Time value)	Minutes
This event occurs only one time	selected

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

Appendix 5

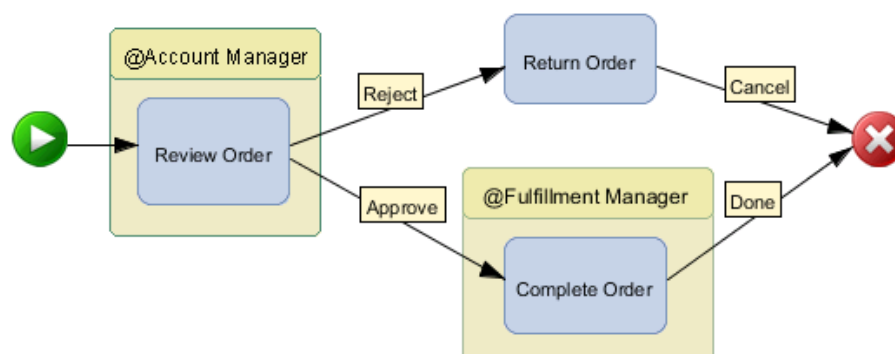
Basic Workflow Examples

Basic Approval Workflow	115
Order Fulfillment Workflow	118

Basic Approval Workflow

The Basic Approval Workflow 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.

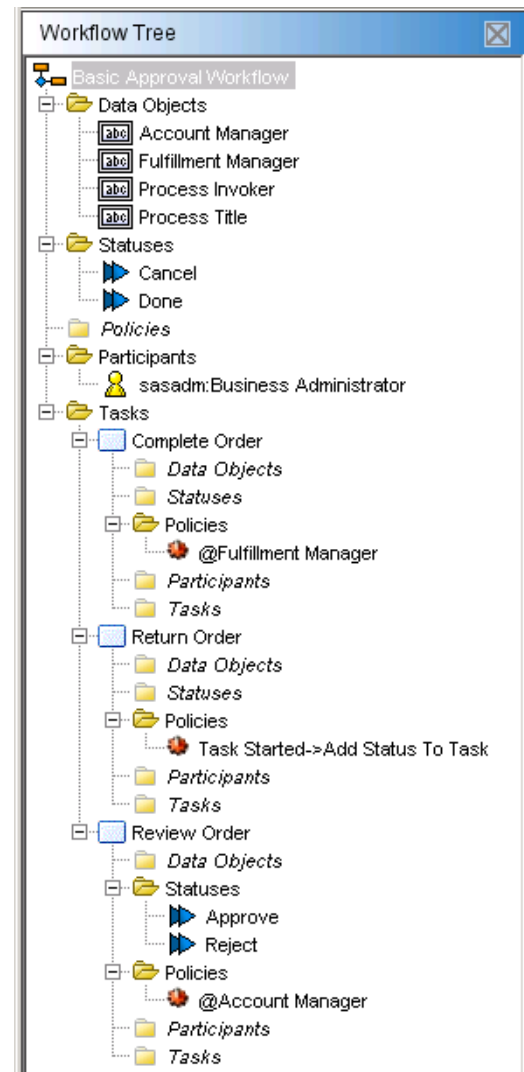
Figure A5.1 Basic Approval Workflow Diagram



This workflow uses the Exclusive Choice workflow pattern.

The following figure shows the workflow tree for the Basic Approval Workflow.

Figure A5.2 Workflow Tree for the Basic Approval Workflow



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

Table A5.1 Basic Approval Workflow Elements

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

Element Name	Element Type	Details
Account Manager	Data Object	Data object of type Short Text defined for all instances. Holds the group name used by the @Account Manager swimlane policy.* The default value is US Region Accountants.
Fulfillment Manager	Data Object	Data object of type Short Text defined for all instances. Holds the group name used by the @Fulfillment Manager swimlane policy.* The default value is US Region Supply Managers.
Cancel	Status	Default status value defined for all templates.**
Done	Status	Default status value defined for all templates.**
sasadm	Participant	This is a User type participant assigned to the root level and each task with the workflow role of Business Administrator. This is a default SAS administrator user configured with the platform in the SAS Authorization Service. The Business Administrator role supports certain administrative tasks such as adding comments, delegating or transferring the task, or releasing the task locked by another user.
Review Order	Task	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 task is selected based on the status value as assigned to the transition.
@Account Manager	Policy/Swimlane	The swimlane corresponds to a group participant defined in the workflow 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 the SAS Authorization Service. For example, the default value is US Region Accountants. When starting a workflow instance, an initial value must be made for this data object. Otherwise, there must be a group defined with that name in the SAS Authorization Service. 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 task.***
Approve	Status	Status value defined for Review Order.†

Element Name	Element Type	Details
Reject	Status	Status value defined for Review Order. [†]
Return Order	Task	This is a system task (that is, automated task) that initiates if the order is rejected. Once completed, the workflow instance ends with a status of Cancel because this task transitions to the terminate node.
Workflow Started->Add Status to Workflow	Policy	This policy assigns the Cancel status because it is an automated task (that is, is not explicitly performed like the previous manual task). This status value stops the Return Order task and triggers the final transition into the terminate node.
Complete Order	Task	This is a manual task that represents the user interacting with the system. This task starts if the order is approved. Then, the workflow instance ends with a status of Done because this task transitions to the terminate node.
@Fulfillment Manager	Policy/Swimlane	The swimlane corresponds to a group participant defined in the workflow 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 the SAS Authorization Service. For example, the default value is US Region Supply Managers. If an initial value assignment is not made for this data object when starting a workflow instance, there must be a group defined with that name in the SAS Authorization Service. 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 task.^{***}

* This data object value is defined at the root level. This allows access to the value by decisions and policies at run time by all contained tasks and subflows. However, the value is not accessible as a local copy of the data object.

** This status value is defined at the root level. This allows access to the value by decisions and policies at run time by all contained tasks and subflows. 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 a SAS platform authorization group or role type must be assigned as Potential Owner and their user members must claim the task to become Actual Owner.

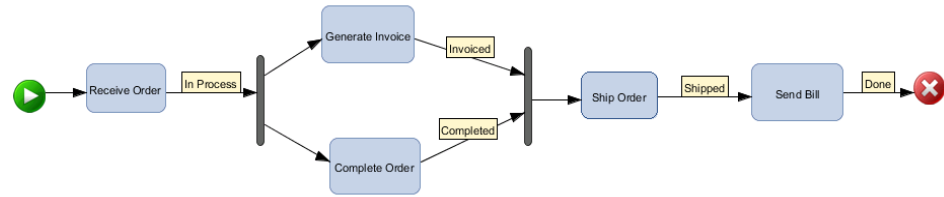
† This status value is defined at the task level. This allows access to the value by decisions and policies at run time by all contained tasks and subflows. However, the value is not accessible as a local copy of the status.

Order Fulfillment Workflow

The Order Fulfillment Workflow template illustrates parallel flow control where two paths are processed concurrently, and the converged path does not start until both of the parallel tasks 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 Workflow Diagram of the Order Fulfillment Workflow

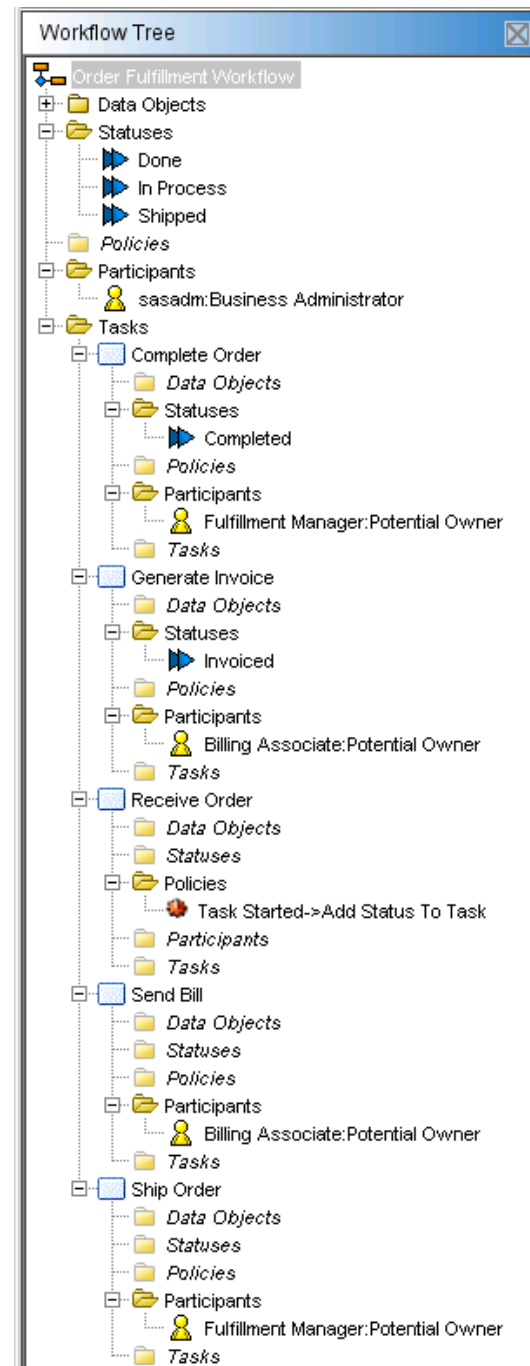


This workflow uses the following workflow patterns:

- Parallel
- Join (Merge with Synchronization)

The following figure shows the workflow tree for the Order Fulfillment Workflow:

Figure A5.4 Workflow Tree for the Order Fulfillment Workflow



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

Table A5.2 Order Fulfillment Workflow Elements

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

Element Name	Element Type	Details
Invoiced	Status	Status value defined for Generate Invoice. [†]
Complete Order	Task	This is a manual task that represents the user interacting with the system. This task 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	This is a group type participant assigned to the task level of the workflow 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 the SAS Authorization Service. 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 task.^{***}
Completed	Status	Status value defined for Complete Order. [†]
Ship Order	Task	This is a manual task that represents the user interacting with the system. This task 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	This is a manual task that represents the user interacting with the system. This task 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 level. This allows access to the value by decisions and policies at run time by all contained activities and subflows. However, the value is not accessible as a local copy of the data object.

** This status value is defined at the root level. This allows access to the value by decisions and policies at run time by all contained activities and subflows. 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 a SAS platform authorization 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 task level. This allows access to the value by decisions and policies at run time by all contained activities and subflows. However, the value is not accessible as a local copy of the status.

Glossary

activity

See task

activity status

See task status

annotation

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

authorization

the process of determining the permissions that particular users have for particular resources. Authorization either permits or denies a specific action on a specific resource, based on the user's identity and on group memberships.

data object

an object that holds the business data that is required to execute workflow tasks.

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

instance

See workflow 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. See also gateway.

policy

a workflow element that associates event-driven logic with a task or subflow. Policies are usually triggered automatically by an event such as a status change or a timer event.

sequence flow

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

subflow

a workflow that is a child of another workflow.

swimlane

a workflow diagram element that enables you to group tasks that are assigned to the same participant.

task

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

task status

the outcome of a task in a workflow. The status of a task (for example, Started, Canceled, Accepted) is typically used to trigger the next task.

workflow

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

workflow definition

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

workflow instance

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

workflow template

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

Index

A

actual owner participant [30](#)
 Add Status to External Workflow policy
 properties [35](#)
 Add Status to Task policy
 properties [35](#)
 alignment modes [25](#)
 arithmetic operators [90](#)

B

bitwise and bit shift operators [90](#)
 business administrator participant [30](#)
 business process management, description
 [1](#)

C

clip explorer [20](#)
 comparison operators [89](#)
 conditional operators [90](#)
 Copy Data Object from External
 Workflow policy
 properties [36](#)
 Copy Data Object policy
 properties [35](#)
 Copy Data Object to External Workflow
 policy
 properties [36](#)
 Copy Participants to Task policy
 properties [36](#)
 copying elements
 maintaining object properties [20](#)
 using the clip explorer [20](#)

D

data object
 adding [54](#)
 assigning to a task [56](#)
 copy from external workflow, policy for
 [36](#)
 copying between data objects, policy for
 [35](#)

 copying to external workflow, policy for
 [36](#)

 deleting [55](#)
 description of [54](#)
 editing [55](#)
 Extract from XML Data Object policy
 [37](#)
 functions for use with [91](#)
 in expressions, examples [91](#)
 Increment Data Object policy [38](#)
 localizing text for [56](#)
 substitution, example [92](#)
 types of [54](#)

Data Object Updated event [34](#)
 decision expression, examples [91](#)
 decision gateway, adding [61](#)
 decision operators
 See operators
 diagram editor [16](#)
 diagram elements, description [17](#)

E

e-mail
 Send E-mail policy [46](#)
 Send E-mail policy, example [103](#)
 events
 Data Object Updated [34](#)
 initiating automatically [32](#)
 Participants Updated [34](#)
 Status Addition [34](#)
 Status Removal [34](#)
 Task Finished [34](#)
 Task Started [34](#)
 Timer Expired [34](#)
 excluded owner participant [29](#)
 exclusive choice
 example [115](#)
 workflow pattern [60](#)
 expiry options for notification policies
 [48](#), [50](#)
 expression examples, decision [91](#)
 Extract from XML Data Object policy
 properties [37](#)

F

functions, in decision expressions 91

H

HTTP Request policy
properties 38

I

Increment Data Object policy
properties 38
Invoke REST Web Service policy
example 95
properties 39
Invoke SAS Code policy
example 98
properties 40
Invoke Web Service policy
example 100
properties 42

L

localization 56

N

notification participant 30
notifications
expiry options for policies 48, 50
Send Workflow Group Notification
policy 46
Send Workflow Notification policy 49
sending, example 103
Notify Participant policy
properties 43

O

operators
arithmetic 90
bitwise and bit shift 90
comparison 89
conditional 90
unary 90

P

parallel, workflow pattern 60
participant 29
adding 30
assigning to tasks 31
configuring 32
copying between tasks, policy for 36
deleting 31

editing 31

Notify Participant policy 43

roles 29

Set Multiple Participant policy 51

Set Workflow Participant policy 52

Participants Updated event 34

policy

actions 34

Add Status to External Workflow 35

Add Status to Task 35

adding 33

assigning to a task 34

Copy Data Object 35

Copy Data Object from External
Workflow 36

Copy Data Object to External Workflow
36

Copy Participants to Task 36

deleting 33

description of 32

editing 33

events that initiate automatically 32

expiry options for notification policies
48, 50

Extract from XML Data Object 37

HTTP Request 38

Increment Data Object 38

Invoke REST Web Service 39

Invoke REST Web Service, example 95

Invoke SAS Code 40

Invoke SAS Code, example 98

Invoke Web Service 42

Invoke Web Service, example 100

Notify Participant 43

properties 34

properties, application-specific values
56

properties, data object substitution 56

Remove Status from Task 44

Schedule Task 45

Send Workflow Group Notification 46

Send E-mail 46

Send E-mail, example 103

Send Workflow Notification 49

Set Multiple Participant 51

Set Overdue Status 51

Set Workflow Participant 52

Start Workflow 52

Start Workflow with Label 52

Stop Workflow 53

Stop Workflow with ID 53

Submit a JES Job 53

Submit JES Job, example 105

timer tool, example 111

potential owner participant 29

process management

definition 1
 propertys, policy 34

R

Remove Status from Task policy
 properties 44

S

SAS code
 Invoke SAS Code policy 40
 Submit a JES Job policy 53
 SAS workflow architecture
 description 5
 diagram 6
 SAS Workflow Services 5
 SAS Workflow Studio
 benefits of 2
 clip explorer 20
 diagram editor 16
 drawing tools 13
 features 6
 format toolbar 14
 inteface 9
 menus 10
 navigation pane 15
 SAS workflow architecture 5
 workflow tree 19
 zoom toolbar 14
 Schedule Task policy
 properties 45
 Send E-mail policy
 example 103
 properties 46
 Send Workflow Group Notification policy
 properties 46
 Send Workflow Notification policy
 properties 49
 sequence flow, connecting tasks 23
 sequence, workflow pattern 59
 Set Multiple Participant policy
 properties 51
 Set Overdue Status policy
 properties 51
 Set Workflow Participant policy
 properties 52
 Start Workflow policy
 properties 52
 Start Workflow with Label policy
 properties 52
 status
 adding 26
 adding to any workflow, policy for 35
 adding to current task, policy for 35
 assigning to a connection 27

default status 26
 deleting 27
 editing 27
 local to a task 28
 localizing text for 56
 purpose 26
 Remove Status from Task policy 44
 Set Overdue Status policy 51
 workflow path based on status 60
 Status Addition event 34
 Status Removal event 34
 Stop Workflow policy
 properties 53
 Stop Workflow with ID policy
 properties 53
 subflow, adding 24
 Submit a JES Job policy
 properties 53
 Submit JES Job policy
 example 105

T

tags
 adding 57, 58
 description of 57
 removing 57, 58
 task
 Schedule Task policy 45
 Task Finished event 34
 task initiator participant 29
 task stakeholder participant 30
 Task Started event 34
 tasks
 adding 22
 aligning vertically or horizontally 25
 alignment modes 25
 connecting 23
 deleting 24
 editing 23
 in the diagram editor 16
 localizing text for 56
 templates
 See also workflow
 loading with the workflow loader utility 81
 versioning 7
 timer
 automatically traversing transition,
 example 70
 expired, example 111
 expression examples 109
 inside task border, example 69
 on task border, example 69
 outside a task, example 70
 tool policy, example 111

Timer Expired event [34](#)

U

unary operators, decision expressions [90](#)

W

web service

 Invoke REST Web Service policy [39](#)

 Invoke Web Service policy [42](#)

workflow

 adding a subflow [24](#)

 adding decision gateways [61](#)

 creating [22](#)

 elements of [16](#)

 example, basic approval workflow [115](#)

 example, order fulfillment [118](#)

 loading templates [81](#)

 localizing text for [56](#)

 Start Workflow policy for subflows or tasks [52](#)

 Start Workflow with Label policy for workflow instance [52](#)

 Stop Workflow policy for subflows or tasks [53](#)

 Stop Workflow with ID policy [53](#)

 versioning [7](#)

workflow group

 Send Workflow Group Notification policy [46](#)

workflow instance management [8](#)

workflow lifecycle management [7](#)

Workflow Loader utility [81](#)

workflow path

 based on expression evaluation [61](#)

 based on status [60](#)

workflow pattern

 alternate paths [60](#)

 exclusive choice [60](#), [61](#)

 exclusive choice, example [115](#)

 inclusive choice [61](#)

 join, example [118](#)

 parallel [60](#)

 parallel, example [118](#)

 sequence [59](#)

workflow tree [19](#)