Siebel Business Process Framework: Workflow Guide

Version 8.0 Rev A

May 2008

ORACLE

Contents

Chapter 8: About Workflow Process Design Options

Chapter 9: About Workflow Policies

Chapter 10: Testing a Workflow Process

Chapter 11: Administering Workflow Processes

Chapter 12: Administering Workflow Policies

Chapter 13: Example Workflow Processes

Chapter 14: Reference Materials for Siebel Workflow

1 What's New in This Release

What's New in Siebel Business Process Framework: Workflow Guide, Version 8.0 Rev A

Table 1 lists changes described in this version of the documentation to support release 8.0 of the software.

Table 1. Documentation Changes in Siebel Business Process Framework: Workflow Guide, Version 8.0 Rev A

Topic	Description
"Running Workflow in the Workflow Process Manager Server Component" on page 34	When a policy invokes a workflow process, Workflow Monitor Agent typically uses Encoded Input Arguments to pass input arguments to the Workflow Process Manager.
"Making a Workflow Process Editable" on page 60	**New topic.** Several conditions must exist to edit a workflow process.
"About the In/Out Process Property" on page 80	**New topic.** If a process property's type is Out or In/Out, then outputs are returned to the caller. For example, to SRProc.
"Passing a Process Property to an Error Workflow Process" on page 82	Describes a procedure for passing a user-defined process property to an error workflow process.
"Referencing a Process Property" on page 85	A process property can be used as a substitution variable within an expression.
"Business Service Calls to a Unix Shell Script" on page 94	**New topic.** If a workflow process invokes a business service that calls a UNIX shell script, the shell script runs as the Siebel Service Owner account.
"Copying Values from a Parent Process to a Subprocess" on page 96	**New topic.** You can copy a value to a record being updated or inserted in a subprocess by creating a process property in the subprocess and assigning the value to it.
"Example of Defining a Siebel Operation Search Specification" on page 99	A Search Specification expression can be defined with compound expressions and substitutions from Process Properties or Fields.
"Updating a Field That Resides in a non-Primary Business Component" on page 100	**New topic.** You can update a field in a non-primary business component.
"Using the Siebel Operation to Update a Process Property" on page 101	**New topic.** You can use a Siebel operation to update a process property.

Table 1. Documentation Changes in Siebel Business Process Framework: Workflow Guide, Version 8.0 Rev A

Topic	Description
"About the Upsert Operation on the Siebel Operation Step" on page 104	**New topic.** The Upsert Operation provides a way to perform either an insert or update operation based on whether a record or records exist in the database.
"Defining a Run-Time Event With a User Interact Step" on page 107	**New topic.** Branches emanating from a user interact step in an interactive flow must be associated with a run-time event.
"Invoking the Stop Step" on page 110	**New topic.** It is recommended that the Stop step be used only in a workflow process invoked from a script.
"Defining a Custom Error Message with the Stop Step" on page 110	**New topic.** If none of the predefined error messages provided in the Error Code property on the Stop Step meet your requirements, you can define a custom error message on the Stop step.
"Comparing Branching Declaratively to Programming With Script" on page 113	**New topic.** You can implement branching declaratively, as in multiple branches emanating from a single step in a workflow process. In some cases, you can achieve the same result through script, as in scripting on a business service.
"Example Requirements for Using a Run-Time Event Compared to a Workflow Policy" on page 139	**New topic.** Describes example scenarios for using a workflow policy compared with a run-time event.
"Identifying A Predefined Workflow Process Invoked by a Run-Time Event" on page 141	**New topic.** You can use the Siebel client to identify predefined workflow processes that appear in the Action Set Field of the Administration-Runtime Events screen.
"About Key-Based Routing with Workflow Process Manager" on page 144	**New topic.** Key Based Enabled can be defined for the Workflow Process Manager to assist with routing and recovery.
"About Workflow Process Manager and Automatic Record Insertion" on page 145	**New topic.** When the Workflow Process Manager inserts a record, by default, the system generated CREATED_BY field is set to the ROW_ID for employee SADMIN. It can be desirable to have the CREATED_BY field set to other than SADMIN's Id.
"Using a Run-Time Event With the User Interact Step" on page 151	The SetFieldValue event for a field that has the Immediate Post Changes property set to TRUE is supported.
"Using a Run-Time Event Within a Business Object Context" on page 151	**New topic.** A workflow process that references a run-time event is only invoked when the run-time event is detected from within the same Business Object context in which the workflow process is based.
"Using a Run-Time Event Within a M:1 Relationship" on page 152	**New topic.** When using a runtime-event to trigger a workflow based on a modification made to a record that contains a M:1 relationship with a parent, you must trigger the workflow based on the child's ROW_ID.

Table 1. Documentation Changes in Siebel Business Process Framework: Workflow Guide, Version 8.0 Rev A

Topic	Description
"Run-Time Events That Cannot Be Used to Invoke a Workflow Process" on page 154	**New topic.** Some run-time events cannot be used to invoke a workflow process.
"Defining an Error Exception to Handle an Update Conflict" on page 159	**New topic.** You can define an error exception to handle an update conflict that occurs when multiple attempts are made to write to the same record at the same time.
"About the Repeating Component Request" on page 166	**New topic.** You can use the Repeating Component Request feature in conjunction with the workflow process to schedule the workflow task to be executed at a predetermined time.
"Using Workflow Process Batch Manager When Primary and Non-Primary Business Components are Present" on page 166	**New topic.** Use the Workflow Process Batch Manager server component when you need to run the workflow process for all records of a particular business component.
"About Workflow Policies and Monitoring Tables" on page 174	Workflow Policies can monitor only Siebel tables. Do not use a Workflow policy to monitor database tables that are external to Siebel, or to monitor certain EIM tables.
"Using an Insert Operation With a Workflow Policy Action" on page 213	**New topic.** An insert operation with Workflow Policy Action does not populate the Primary Owner.
"Using the Run External Programs Type" on page 202	A substitution parameter that contains spaces must be encased in quotes. Note that the quotes are also passed as part of the parameter.
"Specialized Comparisons That Must Be Used With Batch Workflows" on page 209	If a workflow policy condition is IS ADDED or IS UPDATED, then the triggers generated do not represent all the conditions specified in the policy.
"Using the Simulator With a Workflow Subprocess" on page 227	**New topic.** To avoid a validation error, you must publish and activate subprocesses called by the workflow process you are simulating prior to invoking the simulator.
"About the Process Simulator Watch Window" on page 228	**New topic.** Describes the Watch window, including how to use it within the context of an example.
"Using the Process Simulator" on page 233	Describes Process Simulator usage with latest release.
"Run-Time Event Behavior for Mobile Clients" on page 240	There is some variation in the way run-time events behave in mobile clients compared with other Siebel client types.
"Behavior with Full Copy Nodes" on page 239	**New topic.** If you extract a regional node with the routing group set to FULL COPY then workflow process definitions with Replication set to None are routed to the MWC.

Table 1. Documentation Changes in Siebel Business Process Framework: Workflow Guide, Version 8.0 Rev A

Topic	Description
"Periodically Notifying Mobile Users Who Have Not Synchronized" on page 240	**New topic.** You can configure a workflow process to send a notification email to mobile users who have not synchronized over a given period of time.
"Deploying a Workflow Process as a Web Service" on page 240	**New topic.** Describes how a workflow process can be deployed as a Web service.
"Importing and Exporting a Workflow Subprocess" on page 247	When importing a workflow process that contains subprocesses, you must first import the subprocesses and then the parent workflow process. Also, there are name length and carriage return restrictions.
"Disabling Persistence to Avoid Excessive Records in S_WF_PROP_VAL" on page 264	**New topic.** Having Persistence defined on a large number of workflows can cause an increase in the size of S_WF_PROP_VAL.
"About the Workflow Recovery Manager" on page 268	**New topic.** You can use the Workflow Recovery Manager to recover an interrupted workflow process.
"Running Generate Triggers" on page 278	When running Generate Triggers, table names have an 18-character limit.
"About S_ESCL_REQ.REQ_ID" on page 291	Because the S_ESCL_REQ_S sequence does not have an upper boundary, the sequence for S_ESCL_REQ.REQ_ID can potentially generate a number greater than the maximum of 9 digits allowed, causing an overflow.
"Starting Workflow Monitor Agent" on page 292	**Procedure removed.** As of Siebel version 7.0, you must start Workflow Monitor Agent from the Server Manager command-line interface. Workflow Monitor Agent can no longer be run from the Siebel client.
"Checking Modifications Into the Local Database" on page 292	**New topic.** If you use Siebel Tools to modify a workflow policy object, column, or program in the local database, then you must check these objects into the server database.
"Setting the Use Action Agent Parameter" on page 300	**New topic.** In general, it is recommended that you start Workflow Monitor Agent with Use Action Agent set to False, the default setting.
"Moving a Workflow Policy to a Different Group" on page 302	**New topic.** In response to changes in the business environment or you configuration, you might find it necessary to change the workflow group for a given workflow policy.
"Expiring a Workflow Policy" on page 303	**New topic.** If you encounter a *failed to load parent rows* error message, it can be due to the presence of rows in the S_ESCL_REQ, S_ESCL_STATE, and S_ESCL_ACTN_REQ tables that reference expired policies.

Table 1. Documentation Changes in Siebel Business Process Framework: Workflow Guide, Version 8.0 Rev A

Topic	Description
"Converting a Workflow Policy to a Workflow Process" on page 310	**New topic.** In general, it is recommended you keep workflow policies simple, placing most business logic into a workflow process.
"Example Workflow Process That Traverses a Record Set to Close Obsolete Service Requests" on page 324	**New topic.** Provides a detailed example that describes how to traverse a record set.
"Example of Externalizing Properties Used by Siebel Workflow" on page 361	**Content removed.** Removed references to Universal Application Network (UAN), which is not supported in the current release.
"Example of Using the Server Request Business Service to Invoke a Workflow From Script and Pass Values" on page 378	**New topic.** You can use the Server Requests business service to invoke a workflow process from script and to pass field values to process properties.
"Example of Using the Server Requests Business Service to Invoke an EIM Task" on page 379	**New topic.** You can use the Server Requests business service to start a server task from within a workflow process.
"Using an Output Parameter With the Echo Method" on page 383	**New topic.** When using the Echo method, if you do not specify an input parameter but do specify an output parameter, the output parameters are populated with values from the active record of the business component.
"Specifying a Substitution With the Outbound Communications Manager Business Service" on page 386	**New topic.** You can specify a substitution variable when calling the Outbound Communications Manager business service from a workflow process.
"Sending an Email to the Owner of a Product Defect" on page 388	**New topic.** You can use the Outbound Communications Manager to send an email to the owner of a product defect.
"Sending Email to an Email Address Held in Any Column" on page 397	**New topic.** Workflow Manager can be configured to send an email message to an email address that is stored in a column other than S_EMPLOYEE.EMAIL_ADDR.
"Consolidating Messages Sent Through Send Broadcast Message" on page 398	**New topic.** It is not possible to consolidate multiple broadcast messages into a single message that is then displayed to a user.
"Predefined Messaging Workflow Policies" on page 411	**New topic.** You can use a predefined messaging policy to display a message in pop-up dialog box in the Siebel client.

Table 1. Documentation Changes in Siebel Business Process Framework: Workflow Guide, Version 8.0 Rev A

Topic	Description
"Passing a Constant from a Workflow Policy Action into a Workflow Process" on page 417	**New topic.** You can use the Run Workflow Process program to pass a constant from a Workflow Policy Action into a Workflow Process.
"Reference of Workflow Step and Connector Properties" on page 426	**New topic.** Provides reference information for properties of the various step types used in Siebel Workflow.

Table 2 lists changes described in this version of the documentation to support release 8.0 of the software.

Table 2. New Product Features in Siebel Business Process Framework: Workflow Guide, Version 8.0

Topic	Description
Multiple-Document Interface (MDI). See *Using Siebel Tools*.	Oracle's Siebel Tools is now a multiple-document interface application. This means that when you use Siebel Workflow, you can have multiple object editors open at any given time displaying not only workflow processes, but other objects as well.
Debug toolbar. See *Using Siebel Tools*.	This toolbar contains buttons that let you access Siebel VB and Siebel eScript debugging tools. The relevant button for Siebel Workflow is the Watch button that lets your monitor the contents of program variables in the Variable window when simulating a workflow from Siebel Tools.
WF/New Task Editor toolbar. See *Using Siebel Tools*.	In this release, you can use the debug toolbar to publish, activate, revise, and expire workflows/tasks.
Automatic Revisions. See *Using Siebel Tools*.	Workflow and task configuration options that ensure you are working on the most current workflow/task version.
"Upgrading Siebel Workflow" on page 274	Siebel Tools now supports three-way merge during upgrades for Workflow objects. This merge makes sure that the customer's extensions to these objects carry forward to the new version.
"About the Multi Value Property Window" on page 77	In this release, the Multi-value Properties Window (MVPW) replaces the list applet in most cases within the Siebel Tools side of the Business Process Designer. You use the MVPW to view, create, and delete process properties to store data for a workflow process. You can also create, delete, and assign input/output arguments for a process step. The MVPW is context-sensitive; it displays the appropriate set of properties based on the object selected.

Table 2. New Product Features in Siebel Business Process Framework: Workflow Guide, Version 8.0

Topic	Description
"State Management Type and Workflow Processes" on page 424	In this release, a new feature for Siebel Order Management—Web Channel session state management—requires that all business services are marked with a setting describing the Object Manager client-session mode when making a request to the Siebel server. Because the Workflow engine is a business service, this new feature means the addition of a new property field, called State Management Type, for all workflow processes. By default, this property is set to Stateless. You can set this field to indicate whether a workflow process is stateless or stateful.
"Argument Fields for a Process Property" on page 436	This is a new data type for defining process properties specifically used by web services. Strongly typed objects allow you more functional scripts and better performance.
"About Sequence Numbers in Workflow Steps" on page 88	Siebel Workflow automatically assigns a name and sequence number to each step and connector that you create within a workflow process. The name given is based on the type of step or connector. The sequence number differentiates instances of the same type of step or connector. When a new step of the same type is added to the workflow, if there is a gap between the sequence numbers for the type (gaps can be created when steps are deleted), the first sequence number in the first gap interval for the step type will be the sequence assigned to the new step.
"Passing a Property Set by Reference" on page 82	Subprocess methods tasked with modifying large amounts of data must copy large quantities of data. This can negatively impact performance and scalability of these method invocations. Pass By Reference is a feature that allows you to avoid passing large property sets, by passing just a pointer to the property sets. This feature employs a new user property, Business Service User Prop, which is represented by a new child object of the Business Service object.
"About the Siebel Operation Step" on page 97	In this release, the following operations were added: Delete, NextRecord, PrevRecord, and QueryBiDirectional.
"About the Task Step" on page 105.	The Task step has been introduced to accommodate the new Siebel Task UI feature in this release, which is integrated with Siebel Workflow. You use the Task step to launch tasks from within a workflow process.

Table 2. New Product Features in Siebel Business Process Framework: Workflow Guide, Version 8.0

Topic	Description
"About the Process Simulator Watch Window" on page 228	The Watch window allows you to see object and variable values during simulation. In this release, if the variable is editable, you can update it in the Watch window to feed a value to the simulation.
"Developing a Migration Strategy" on page 244	ADM is an automated deployment framework that provides a mechanism to migrate enterprise customization data (views, responsibilities, assignment rules, and so on) from one Siebel application environment to another. In this release, workflow processes and workflow policies are data types newly available for migration through ADM. Using ADM, you can perform bulk migration and activation of workflows, and you can migrate policies in batch as well.
"About the Workflow Admin Service Business Service" on page 384	A new business service that allows you to perform batch import/export, deployment, and activation of multiple workflow processes by way of a search specification. Alternatively, you can perform client-side batch activation and expiration by way of the File menu in the application.
"Workflow Deployment View" on page 251	In the enhanced Workflow Deployment view (Administration - Business Process), administrators can now view the parent-child relationship between a workflow process definition and its run-time records. Administrators can use the multi-select function in the list views to perform activation, deletion, and expiration of workflow processes in batch.

Additional Changes

This version of *Siebel Business Process Framework: Workflow Guide* also includes structural changes to the content, such as topic organization and heading arrangement, and revisions to procedures.

NOTE: The Siebel Bookshelf is available on Oracle Technology Network (OTN) and Oracle E-Delivery. It can also be installed locally on your intranet or on a network location.

2 Overview of Siebel Workflow

This chapter provides a conceptual overview of Siebel Workflow. It includes the following topics:

General Workflow Principles

This topic describes general workflow principles. It includes the following topics:

About Siebel Workflow

Siebel Workflow is a customizable business application that allows you to define, manage, and enforce your business processes, thereby creating process automation within Oracle's Siebel applications.

Siebel Workflow orchestrates the various Siebel process automation technologies. A *Siebel Workflow Process* graphically sequences a series of automation steps that support a business process, and it specifies inputs and outputs for individual steps and for the workflow process as a whole.

Challenges you can address when using Siebel Workflow to manage a business process in your organization include:

- Automating escalation of events and notification of appropriate parties.
- Routing and assigning work.
- Processing work.
- Enforcing authorization and transition rules.

Business Requirements That Can Be Addressed With Workflow Management

In theory, businesses are managed according to policies and procedures that allow efficiency, quality service, adherence to contractual agreements, and profitability. Business processes that these policies enforce include:

- Allowing that response time objectives are met for customer callbacks and open service requests.
- Specifying review policies for important processes, such as contracts, quotes, or product shipments.
- Monitoring service requests or opportunities over time.

In practice, the benefits of policies often are not realized because policies are not consistently enforced. This condition can exist due to the large number of business processes in the business environment or the dynamic nature of the information being monitored.

The management of important events is central to the enforcement of business workflow. *Workflow Management* is the timely management of an event to allow proper handling. For example, a service department has procedures for managing an open service request or making sure a measurable response time is met. A workflow process can increase the visibility of these processes within an organization and check that they are correctly handled.

A service department has sets of defined rules that match their policies:

- **Standards for processing calls.** For example, when a Severity 1 call is assigned, the new owner is automatically paged.
- **Contracted service agreements that must be adhered to.** For example, a customer can purchase a support agreement guaranteeing a callback in two hours and problem resolution in four hours.

A sales department also has rules to enforce required business practices:

- **Discount authority.** If a sales representative quotes a discount exceeding the maximum discount allowed, it requires the approval of the district sales manager or VP of Sales.
- **Pipeline management.** Each sales representative manages his or her pipeline to promote sufficient levels of prospects at each stage of the sales cycle. If an area of the pipeline needs attention, the representative or manager must be alerted.
- **Forecasting accuracy.** Opportunities that are forecasted but never closed or forecasts having wide discrepancies with the actual revenue must be flagged.

Siebel Workflow and Process Automation

Siebel Workflow brings together Workflow Processes and other repository configuration objects, including the Workflow Policies module, for creating a comprehensive workflow design. Some of the process automation technologies available to a Siebel application are described briefly in this topic.

While each of these technologies provide specific functionality to implement business process automation in the Siebel application, Workflow Processes orchestrate many of the services the technologies perform. A workflow process orchestrates services by directly invoking each technology, or by interacting with other technologies through the Siebel event model.

Table 3 describes automation technologies.

Table 3. Description of Automation Technologies

Automation Technology	Description
Workflow Process	Allows you to define your company's business processes using a familiar flowcharting interface. A workflow process consists of one or more process steps such as start steps, sub process steps, decision points, and tasks. Workflow Processes is the key technology behind building business processes in the Siebel application.
Workflow Policy	Allows you to define conditions and actions that can invoke a workflow process. When policy conditions are met, the policy action executes the relevant workflow process. Workflow Policies generates events based on database operations. Workflow Policies is effective for performing simple actions such as sending email, or creating an activity or assignment.
Task UI	Allows you to create a user interface with multiple step, interactive operations that can include branching and decision logic to guide a user through task execution. Siebel Task UI allows navigation backward and forward within task execution, and allows task execution to be paused and resumed as needed. For more information, see *Siebel Business Process Framework: Task UI Guide*.
Assignment Manager	Expresses rules to assign records to people. Allows assignment of records based on skills, workload, and availability. Supports ownership transitions within a process. For more information, see *Siebel Assignment Manager Administration Guide*.
SmartScript	Guides users through data entry activities. SmartScript is effective for call scripting and includes basic support for transaction level commits. For more information, see *Siebel SmartScript Administration Guide*.
Activity Template	Provides a predefined series of steps to be performed. Activity Template is effective for handling asynchronous/offline work. For more information on Activity Template, see *Siebel Applications Administration Guide*.
State Model	Restricts transition of record status based on a current value and the position of the user. State Models can also enforce directional progression of status. For example, Opportunities move forward but not backward through a pipeline. For more information on State Models, see *Siebel Applications Administration Guide*.

About the Workflow Processes Module

Workflow Processes is the module in Siebel Workflow you use to create and administer a workflow process.

Overview of Workflow Development

Figure 1 illustrates a high-level view of components used during workflow development.

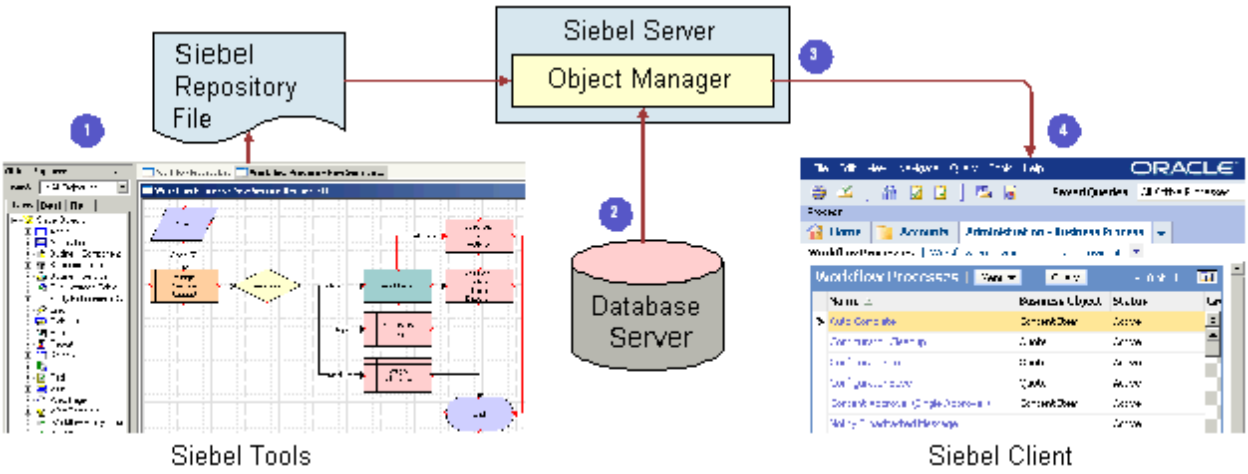


Figure 1. High-level view of Workflow Development

Components involved during development of a workflow process include:

1. **Siebel Tools.** Siebel Tools provides an integrated development environment for configuring aspects of a Siebel application. The Workflow Process Designer resides in Siebel Tools. You use the Process Designer to build a workflow process. These modifications are saved in the Siebel Repository File (srf).

   Siebel Tools provides the design interface and the debugging interface to Siebel Workflow. After a workflow process is designed and tested, it is written to repository tables for deployment from the administrative interface in the Siebel client.

2. **Siebel Server.** The Object Manager in the Siebel Server manages the workflow process components that automate business policies.

3. **Database Server.** A relational database provides the set of data that Workflow Policies act against.

4. **Siebel Client.** Workflow processes and workflow policies are administered through the Administration-Business Process views in the Siebel client.

For more information, see Chapter 3, "Architectures Used With Siebel Workflow". For a description of the Siebel Server architecture, see *Siebel System Administration Guide*.

Overview of Workflow Process Configuration

Workflow Processes allows you to define your company's business processes using the Process Designer in Siebel Tools. Hosting the Process Designer in Siebel Tools provides an integrated development environment to configure Workflow along with other Siebel objects. This integrated development environment allows a top down development framework for building business logic, starting with the creation of a workflow process, then providing pluggable services and data objects that make the process executable.

You define a process that consists of process steps such as Start steps, Decision Points, Sub Process steps, or Business Service steps to finish work. Work can be finished with a predefined business service or a custom business service. Predefined actions include updates to the Siebel database, notifications, such as an email or page, integration messages to external systems, and calls to invoke server tasks. Custom actions can be defined by using Siebel VB or Siebel eScript.

For more information, see Chapter 4, "Developing a Workflow Process".

Overview of Workflow Process Administration

A Workflow Process can be simple, such as entering a product order, or complex, such as managing call center workflow. A complex workflow process can include multiple sub processes.

Workflow Processes are administered through the Administration-Business Process views in the Siebel client. For more information, see Chapter 11, "Administering Workflow Processes."

Overview of Invoking a Workflow Process

A workflow process can be invoked from an event in the Siebel application or from an external system. The ways in which a workflow process can be invoked within the Siebel application include:

- From a workflow policy.
- From an event, such as an insert of a record or a button click.
- By a server component.

This book describes in detail how to invoke a workflow through these mechanisms. For more information, see Chapter 8, "About Workflow Process Design Options."

Scenario for Using Workflow to Promote Timely Service Request Resolution

This topic gives one example of how a workflow process can be used to promote service requests are resolved in a timely fashion. You might use workflow processes differently, depending on your business model.

This scenario underscores how basic Siebel Workflow concepts can be used to automate a business process. Prior to implementing Siebel Call Center, a service manager for a high volume service agency felt her organization was unable to resolve many customer issues in a timely manner. To better track and manage service requests, the service manager decides to implement the Service Request module to automate the company's service request management process.

The goal is to meet a Service Level Agreement commitment by making sure newly logged service requests (SRs) are resolved within a specific amount of time. The service manager needs the SRs to be assigned by the Siebel application to the most appropriate representative based on availability and matching skills. If the SR needs immediate attention, the manager needs the owner of the SR notified.

The service manager's developer uses the Process Designer in Siebel Tools to define the business process for a new service request. Figure 2 illustrates a diagram of the process as it appears in the Process Designer. The diagram includes the steps and decision points involved when a new service request comes into the organization. To see how this workflow fits within the development architecture, see "Overview of Workflow Development" on page 22.

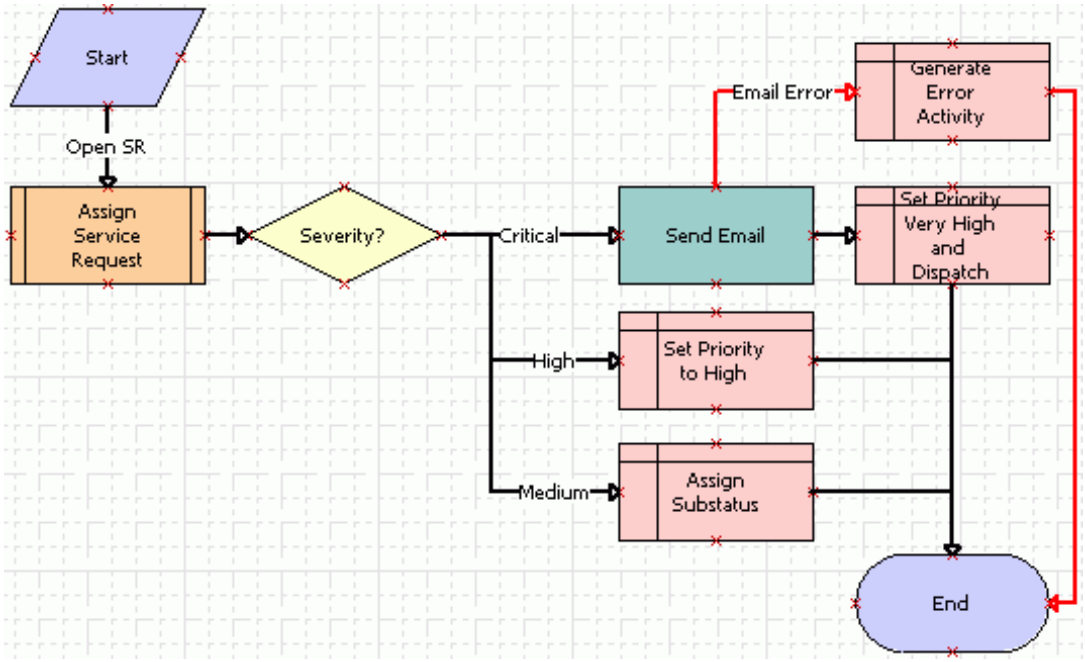


Figure 2. New Service Request Workflow Process in Process Designer

When an SR is logged, the workflow process is triggered. The workflow process calls Siebel Assignment Manager to assign the SR to the best available service representative. Based on the severity of the SR, Workflow can then send email notification to the representative, using Siebel Communications Server. Automating this process helps the company achieve faster turnaround time to resolve SRs and to meet the company's service commitments.

Step Descriptions for the Scenario

Table 4 describes each step used in the new service request scenario. Note that the descriptions are not generic. They refer specifically to the workflow process steps used in this scenario.

Table 4. Description of Steps in the New Service Request Scenario

Step Name	Step Type	Description
Start	Start	Every workflow process has a start step that is used to commence the process. Invocation is defined on the connector emanating out of the start step.
Open SR	Branch Connector	The workflow is invoked by the creation of a new service request record.
Assign Service Request	Subprocess	The service request is assigned to the appropriate agent based on assignment rules defined in the subprocess.
Severity?	Decision Point	The decision point uses service request severity to determine the next step that is taken: Critical, High, or Medium.
Send Email	Business Service	If the service request priority is critical, the business service step sends an email to the assigned agent.
Set Priority to High	Siebel Operation	The service request priority is set to High.
Assign Substatus	Siebel Operation	The sub status is set to Assigned.
Email Error	Exception Branch	If the email is undeliverable or if the Send Email business service step returns some other error, an exception branch is taken.
Generate Error Activity	Siebel Operation	An activity is generated to manage the error.
Set Priority Very High then Dispatch	Siebel Operation	The service request priority is set to Very High and the substatus is set to Dispatch.
End	End	The workflow process ends.

To view sample workflow processes

1. in Siebel Tools, navigate to the Workflow Processes Object List Editor (OBLE).
2. Query the Comments property for *Sample*.
3. Right-click an object definition in the Workflow Processes OBLE, then choose Edit Workflow Process.

3 Architectures Used With Siebel Workflow

This chapter provides an introduction to the architectures used with Siebel Workflow to automate your organization's work processes. It includes the following topics:

Siebel Workflow is an interactive software tool that allows you to automate how your organization handles workflow processes. Workflow uses as its basic model the same processes that organizations use in their sales, marketing, and service departments that determine business workflow. You can use a workflow process to promote consistency and adherence to processes through the automatic enforcement of business policies and procedures.

The Siebel Workflow product provides a graphical user interface featuring a familiar flowcharting methodology for designing a workflow process.

About Architectures Used with a Workflow Process

This topic provides information about the various architectures with a workflow process. It includes the following topics:

In addition, topics that also contain information of an architectural nature include:

Workflow Development Architecture

Figure 3 illustrates the Workflow development architecture.

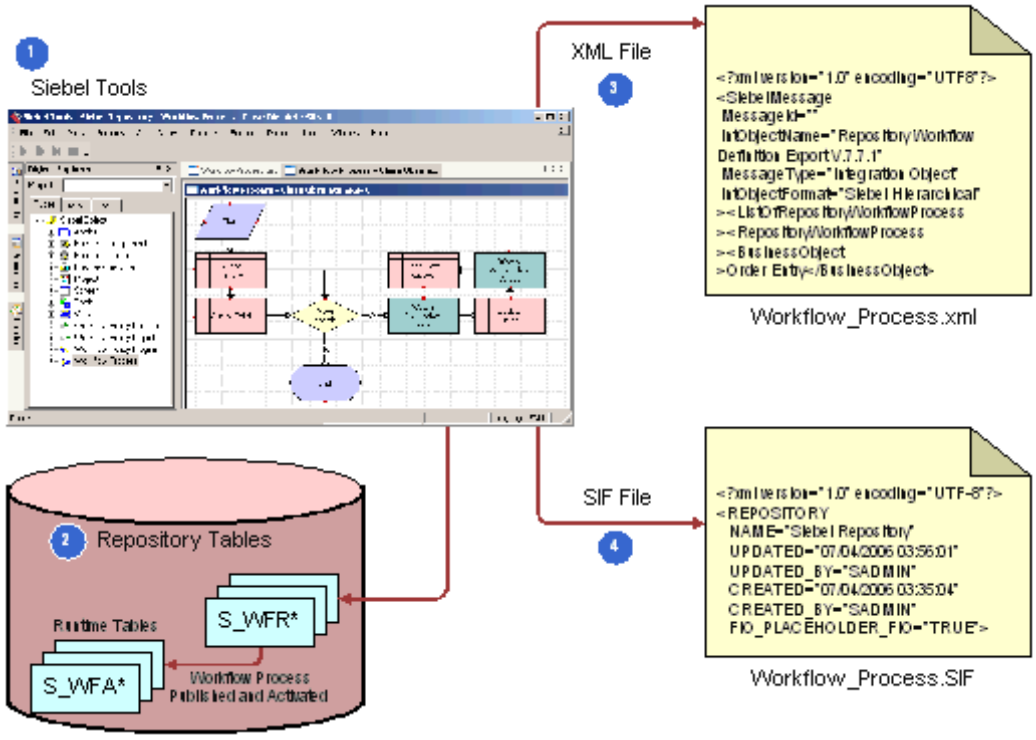


Figure 3. Workflow Development Architecture

Architectural components for workflow development include:

1 **Siebel Tools**. The Workflow Process repository object is a top level object in the Object Explorer of Siebel Tools. You use the Object List Editor (OBLE) to create a workflow process object definition. A workflow process belongs to a project. There is independent versioning of a workflow process in Siebel Tools and in the Siebel client.

   You use the Process Designer in Siebel Tools to develop a workflow process. In the Process Designer, you use process properties to create workflow definitions, or alternatively, you can enter data through unbounded picklists. In the Process Designer, configuration data is available, but run-time data is not available.

2 **Repository Tables**. A workflow process under development is stored in the Siebel repository tables and run-time tables. When you edit the process in Siebel Tools, it is stored in the repository tables. When you deploy a workflow process, it is also inserted into the run-time tables. To deploy a workflow process, you publish then activate it.

3 **XML file**. A workflow process can be exported as an XML file.

4 **SIF File**. A workflow process can be exported as a sif, or Siebel archive file.

You can also use Application Deployment Manager (ADM) to migrate workflow processes from one Siebel environment to another. For more information, see "Migrating With Application Deployment Manager" on page 245.

For a high level description of the development architecture, see "About the Workflow Processes Module" on page 22.

Functional View of Developing a Workflow Process

The typical approach for developing a workflow process is illustrated in Figure 3.

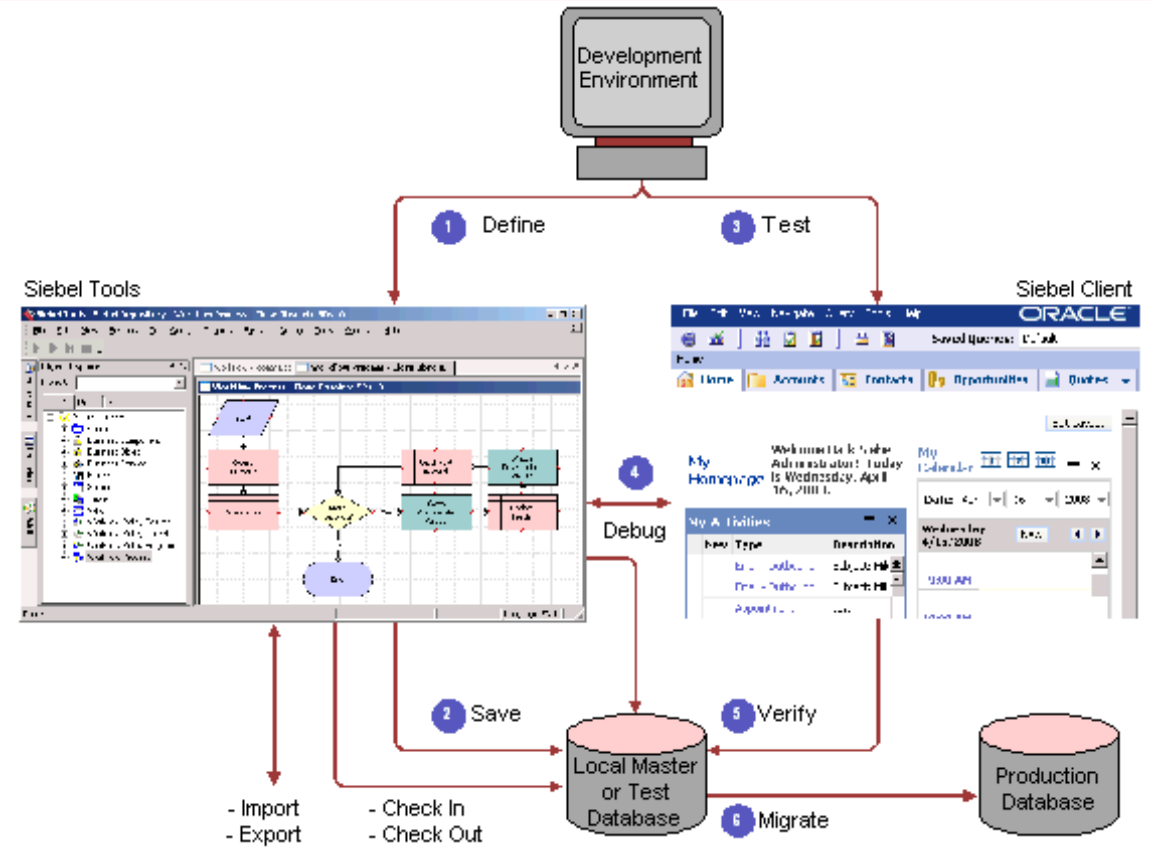


Figure 4. Typical Approach to Developing a Workflow Process

This approach outlines activities typically performed when developing a workflow process:

1 **Define.** Siebel Tools is used to define the workflow process.

When defining a workflow process, you create the object definition for the workflow process, define process properties, add steps and connectors, and so forth.

2 **Save.** The workflow process is frequently saved to the local database.

3 **Test**. The Siebel client is used to test the workflow process.

4 **Debug**. Siebel Tools, the Siebel client, and the local master or test database are used to publish, activate, and debug the workflow process.

   Workflow definitions are checked out from the repository into the local database, where they are modified and debugged locally before being checked in to the master repository. Debugging against a server or test database instead of debugging locally allows the Workflow engine to access server components, such as the Server Request Broker. During the development effort tasks that you can optionally perform in Siebel Tools include:

   - Check the workflow process into and out of your master database.
   - Export the workflow process to an XML file for backup purposes, or import the workflow process from an XML file to restore it.

5 **Verify.** To verify that the workflow process implements the functionality described during the planning phase, the workflow process is run from the Siebel client using the local master or test database.

6 **Migrate.** The workflow process is migrated from the master database to the staging or production database.

Workflow Simulation Architecture

After defining your workflow process, you can test it using the Process Simulator. Testing your workflow process before migrating it to the production environment verifies that resulting actions are accurate and useful and the results are exactly what you need.

Figure 5 illustrates the workflow simulation architecture.

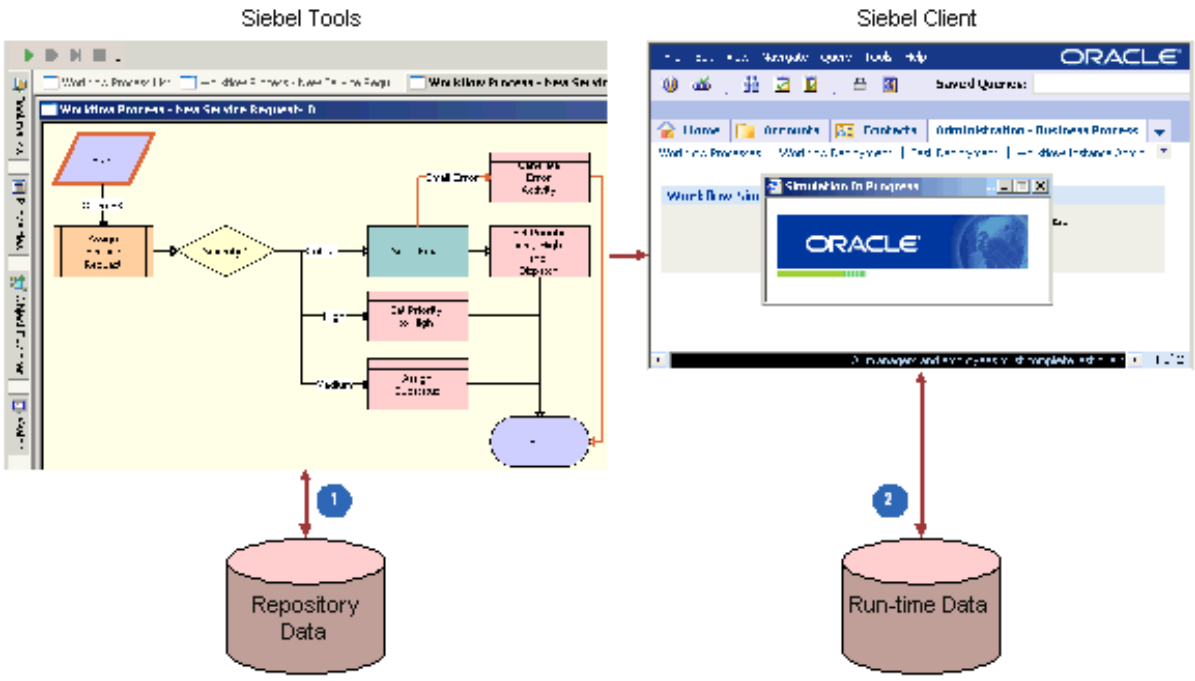


Figure 5. Workflow Simulation Architecture

Architectural components for workflow simulation include:

1. The Process Simulator accesses object definitions that are part of the workflow process or that are referenced by the workflow process.
2. Depending on the configuration you specify in the workflow definition, user data in the run-time database, such as data that resides in various fields of an opportunity record, is accessed and / or manipulated during the simulation.

When the workflow process is run from the Process Simulator, it runs in the application object manager. Actual invocation of the process can be run in the application object manager or in the Workflow Process Manager server session, depending on specific parameters.

For information about Process Simulator usage, see "About the Testing Tools" on page 224.

Workflow Deployment Architecture

After defining and testing your workflow process, the workflow is deployed.

Deploying is the act of performing a two step process:

1. **Publishing**. Reading object definitions for a workflow process that exist in the Siebel repository tables then writing those definitions along with deployment parameters into the run-time tables.

2 **Activating**. Making the workflow process available for use in the Siebel client.

Figure 6 illustrates the relationship between Siebel Tools and the Siebel client when deploying a workflow process.

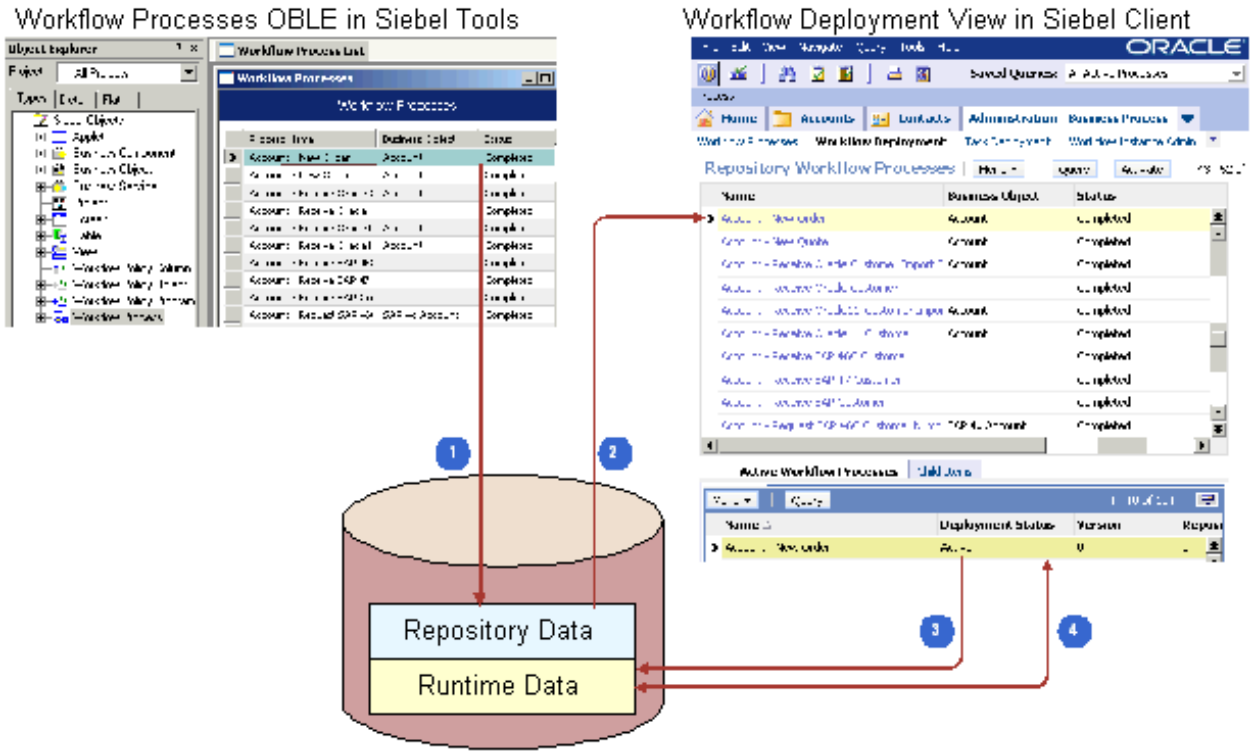


Figure 6. Workflow Deployment Architecture

Architectural components for workflow deployment include:

1 The workflow process is marked Completed for deployment.

2 The workflow process is read from the repository.

3 When activated, the workflow process is written to the run-time tables.

4 Some parameters of the deployed workflow process can be modified in the Siebel client.

To deploy a workflow process, it is not necessary to compile the SRF, nor is a merge required. Workflow components and definitions are defined as Siebel Tools objects and are stored in the Siebel Tools repository. Before you can run a workflow process as a server task or from the Siebel client, you must first publish the workflow process from Siebel Tools, then activate the workflow process from the Siebel client.

NOTE: If you use the Publish/Activate button rather than the Publish button to publish the workflow process in Siebel Tools, it is not necessary to separately activate the workflow in the Siebel client.

Workflow Run-Time Architecture

This topic includes the following topics:

Workflow Run-Time Architecture Overview

The Workflow run -time architecture is based on the Siebel Object Manager layer and the server infrastructure layer of the Siebel Business applications architecture. The run-time environment is available both as a business service and as a server component. Modes in which to invoke and resume a workflow process include:

- Local Synchronous
- Remote Synchronous
- Remote Asynchronous

Figure 7 illustrates the Workflow run-time architecture.

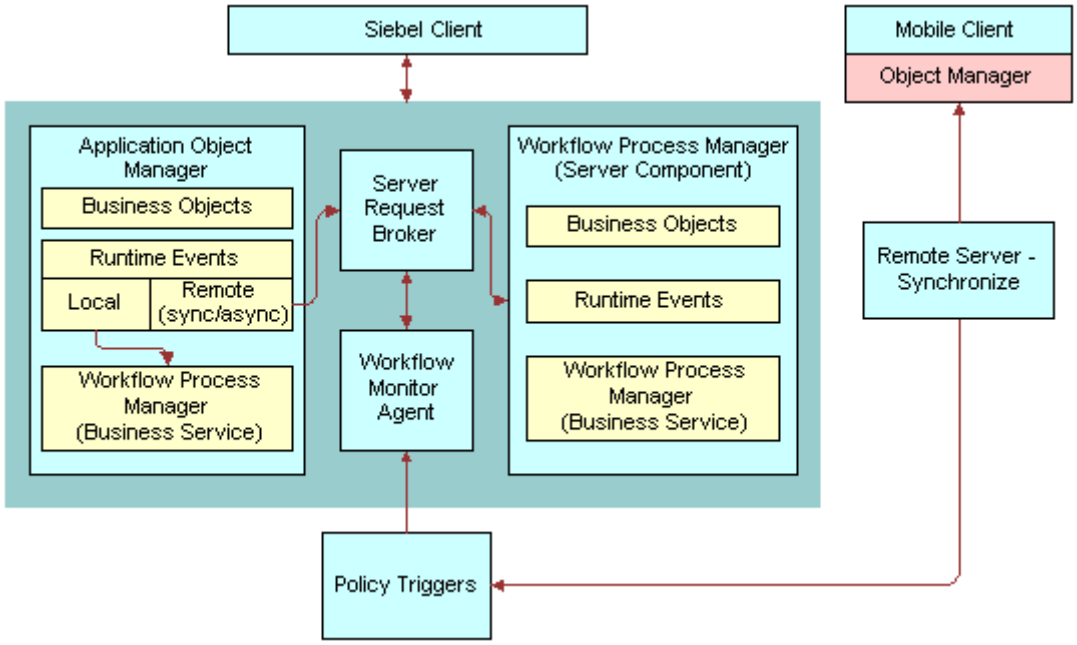


Figure 7. Workflow Run-time Architecture

About the Workflow Process Manager

The Workflow Process Manager is a server component that uses the Siebel Object Manager framework and runs workflows as a business service. The Workflow Process Manager, hosting the Business Object layer and the Data Object layer, is an architecture that provides the ability to run multiple object managers and multiple tasks for each object manager. The name Workflow Process Manager refers to both the Workflow engine and the workflow server components.

The Workflow Process Manager is active and can receive and process requests when it is in Online status. The Workflow Process Manager is inactive in other statuses, such as Shutdown and Offline. In such cases, requests to the WfProcMgr and WfProcBatchMgr server components cannot be processed. For example, if the requests have been saved to the database and the requests are submitted in DirectDB, the requests can be submitted later when WfProcMgr comes back online. Otherwise, the requests are lost.

This topic provides a conceptual overview of the Workflow Process Manager. For more information, see "Invoking a Workflow Process That Runs in the Workflow Process Manager" on page 144.

Running Workflow as a Business Service

Workflow execution in an application object manager is invoked as a business service. The *Workflow Process Manager* business service and the *Workflow Process Manager (Server Request)* business service are referred to collectively as the Workflow engine. As a business service, the Workflow engine takes input arguments and returns output arguments.

Table 5 describes how the business service used determines where the business process is run.

Table 5. Description of How The Business Service Used Determines Where the Workflow Process Is Run

When The Business Service Is. . .	. . .the Workflow Process Is Run Here
Workflow Process Manager	The object manager of the application called.
Workflow Process Manager (Server Request)	The Workflow Process Manager server component.

Running Workflow in the Workflow Process Manager Server Component

A workflow process can be executed in the background using a Workflow Process Manager server component that is configured and optimized to run the Workflow Process Manager business service. The Workflow Process Manager server component acts as the object manager to run the workflow process, including application logic within the workflow process.

Ways in which the WfProcMgr accepts the process name include:

- Through the Process Name component parameter. For example, when starting a task from Server Manager or (Repeating) Server Component Requests.
- Through the Encoded Args component parameter. For example, when submitting requests from WorkMon or the Server Requests business services.

When a policy invokes a workflow process, Workflow Monitor Agent typically uses *Encoded Input Arguments* to pass input arguments to the Workflow Process Manager. However, setting Encoded Input Arguments at the Component Request Parameters applet will fail because it is not in a format that can be recognized by the WfProcMgr server component.

Workflow Management Server Components

Table 6 describes server components in the Workflow Management server component group.

Table 6. Description of Components in the Workflow Management Server Component Group

Server Component	Alias	Description
Workflow Process Manager	WfProcMgr	Functionality provided by the WfProcMgr and WfProcBatchMgr Workflow Process Manager server components include: ■ Acts as the application object manager to run workflows. ■ Are specialized server components configured and tuned to run workflow processes. ■ Similar to server components, provides a multi-threaded environment.
Workflow Process Batch Manager	WfProcBatchMgr	
Workflow Monitor Agent	WorkMon	Executes and monitors workflow policies, then executes actions once policy conditions are met.
Workflow Action Agent	WorkActn	Process requests logged in the action request table, S_ESCL_ACTN_REQ, for a policy group and invokes actions linked with the Workflow policy being processed.
Workflow Recovery Manager	WfRecvMgr	Polls the Workflow engine to check workflow instances running on the server. Recovers failed instances and resumes instances that have been waiting beyond a due date. For more information, see "About the Workflow Recovery Manager" on page 268.
Generate Triggers	GenTrig	Functionality provided by the GenTrig component include: ■ Allows you to create database triggers that Workflow Policies uses to identify records that match policy conditions. ■ Must be rerun whenever new policies are created or deleted, and when existing policies are updated. ■ Can be run from either the Server Manager graphical user interface, or through a command-line interface.

Workflow Process Mode

Workflow process modes that characterize run-time behavior in Siebel Workflow include:

- **Service Flow.** Executes a set of operations upon event invocation.
- **Interactive Flow.** Navigates the user across Siebel views.
- **Long Running Flow.** A persistent workflow that can last for hours, days, or months.
- **7.0 Flow.** Provides backward compatibility for workflows created prior to version 7.7.

The mode is set through the Workflow Mode property in the Workflow Processes OBLE in Siebel Tools. For more information, see "About the Workflow Mode Types" on page 121.

Invoking a Workflow Process

There are a number of different ways to invoke a workflow process. A few examples include through a run-time event or a workflow policy. For more information, see "Invoking a Workflow Process" on page 137.

Administering and Monitoring a Workflow Process

You can use the Administration-Business Process views in the Siebel client to administer and monitor a workflow process. Tasks can be performed in these views include:

- Stop a workflow process.
- Delete a workflow process instance.
- Purge a workflow process instance from the database.
- Monitor a workflow process that is running.
- Recover a workflow process that is interrupted.

For more information, see "Administering Workflow Process Instances" on page 253.

About Workflow Process Interactions with Other Siebel Components

In automating your organization's business processes, Siebel Workflow interacts with various components of the Siebel Business architecture.

Siebel Server Components

The Workflow engine interacts with other server components through the Server Request Broker. Working as a business service, the Workflow engine calls server components.

To invoke server components that are exposed as specialized services, the Workflow engine calls them by their respective signature. For example, to send emails, the Workflow engine calls the Communications Server as the Outbound Communications Manager business service. To assign objects to users, it calls the Assignment Manager component as the Synchronous/Asynchronous Assignment Request business service.

To invoke server components that are not exposed as specialized services, the Workflow engine uses the predefined business service called Server Request. The Server Request business service sends a generic request to the Server Request Broker. For more information, see "About the Server Requests Business Service" on page 374.

About the Server Request Broker

The Server Request Broker (SRBroker) acts as a request broker for the Siebel application server. The Workflow engine sends requests to SRBroker, synchronously or asynchronously, and SRBroker brokers the request to the appropriate component. The actions that are involved include:

- Sending asynchronous messages from an interactive server component to the Workflow engine.
- Communication, synchronous and asynchronous, between the Workflow engine and batch components.
- Scheduling repeated tasks that are to be executed periodically in the Workflow engine.

Another job performed by SRBroker is load balancing. When SRBroker receives a request, it routes it to the server component in the current server. For Siebel workflow, if the component is not available in the current server, SRBroker then sends it to other servers on a round robin basis where the Workflow Process Manager component is activated.

Siebel workflow also uses SRBroker to resume a waiting workflow process. SRBroker pools a database table on a regular basis to see server tasks that must be resumed.

For more information about synchronous and asynchronous invocation of a workflow process, see "About the Server Requests Business Service" on page 374. For more information about SRBroker, see *Siebel System Administration Guide*.

Personalization Engine

The Personalization engine handles run-time events, such as application events, applet events, and business component events. It is through integration with the Personalization engine that Siebel workflow handles run-time events. A workflow process triggered or resumed by run-time events registers itself with the Personalization engine at the time of the process's activation. When a run-time event occurs in a user session, the Personalization engine calls Workflow in the local object manager.

Inbox

Inbox is a single screen in Siebel Business Applications that displays approval and notification items and tasks assigned users regardless of the screen where the item originated. Inbox displays enough detailed information about the item so that users can act on the item from the Inbox and not have to navigate to other screens. For more information about the Inbox, see *Siebel Applications Administration Guide*.

Tasks

Siebel Task UI allows you to create a user interface that is similar to a wizard, with multiple step, interactive operations that can include branching and decision logic to guide users through task execution. Task UI allows navigation both backward and forward within task execution, and allows task execution to be paused and resumed as needed. You can integrate tasks in a workflow process by including Task steps.

For more information, see *Siebel Business Process Framework: Task UI Guide*.

4 Developing a Workflow Process

This chapter provides information about planning a workflow process. It includes the following topics:

Roadmap to Developing a Workflow Process

Figure 8 depicts the development lifecycle for a workflow process.

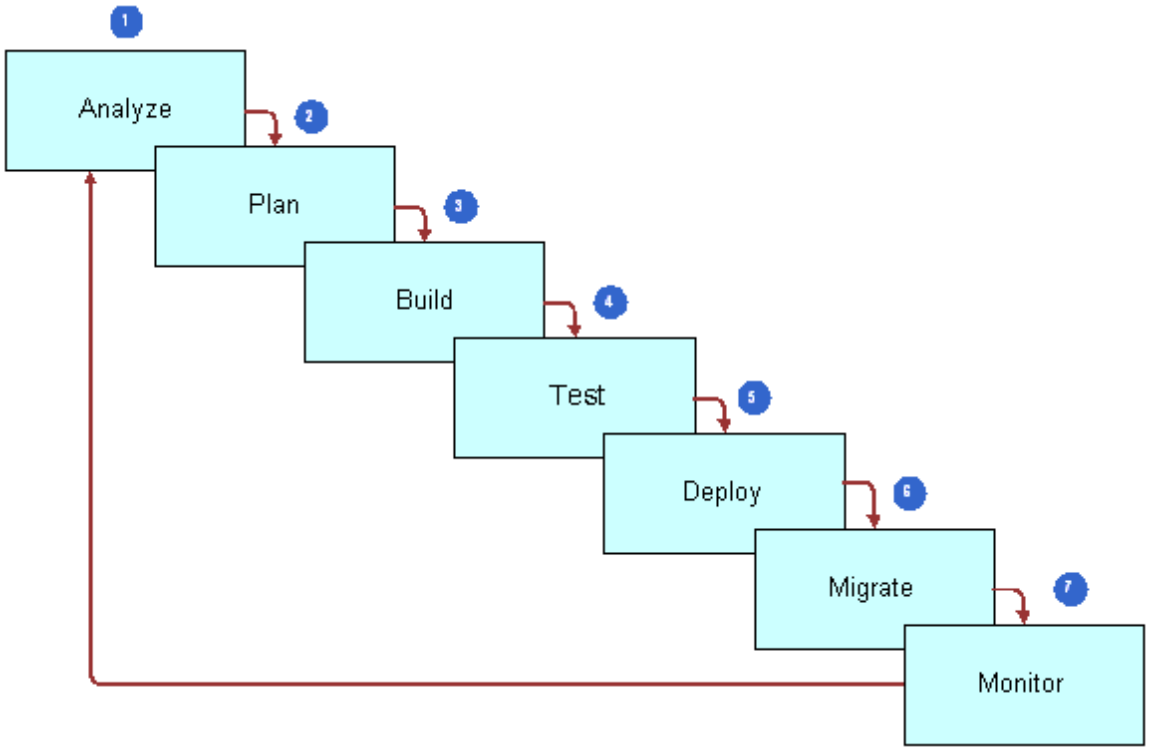


Figure 8. Development Lifecycle for A Workflow Process

The steps for developing a workflow process include:

1. **Analyze**. Analyze your business requirements and the rules and processes to be automated.
2. **Plan.** Plan for building the workflow process,
3. **Build**. Build the process by defining workflow objects in Siebel Tools. Example objects include the workflow process object definition, process properties, and workflow steps.
4. **Test.** Test your workflow process to check that the objects and exception handling you defined meet the business requirements. This includes validating and simulating the workflow process, then verifying functionality.
5. **Deploy.** Deploy your workflow process by publishing the workflow's object definitions from the repository tables to the run-time tables, then activating the workflow for use in the Siebel client.
6. **Migrate.** Migrate the tested workflow process to the production environment. You can use a utility, such as ADM, REPIMEXP, or Import/Export.
7. **Monitor**. Monitor and troubleshoot the migrated workflow process in the production environment.

Note that while the lifecycle illustrated is a linear flow, the typical development cycle of a workflow process is iterative.

Processes Involved in Developing a Workflow Process

To develop a workflow process, perform the following processes and tasks:

1. Process of Analyzing Business Requirements on page 41
2. Process of Planning a Workflow Process on page 45
3. Process of Building a Workflow Process on page 57
4. Process of Testing a Workflow Process on page 231
5. Process of Deploying a Workflow Process on page 237
6. Process of Migrating a Workflow Process on page 243
7. Monitoring Workflow Processes in a Production Environment on page 257.

Process of Analyzing Business Requirements

This process is a step in "Roadmap to Developing a Workflow Process" on page 39.

The first step in developing a workflow process includes analyzing your business requirements, identifying the rules and business processes that are automated by Siebel Workflow.

An implementation project team typically spends a fair amount of time on requirements analysis, with this step taking about 30% of the of the total implementation time. A Business Analyst defines the processes to be automated using the Siebel application, the determines the most appropriate automation solution. The developer who configures the workflow process often participates as a technical consultant during this analysis.

To analyze process requirements, perform the following tasks:

1. Gathering Information for Workflow Process Planning on page 41
2. Determining Workflow Process Requirements on page 42
3. Identifying an Automation Solution on page 43

Gathering Information for Workflow Process Planning

This task is a step in "Process of Analyzing Business Requirements" on page 41.

You start gathering information by looking at how your organization currently handles workflow issues, business processes, and overall workflow. These current processes are the basis of what you create using Siebel Workflow.

If you currently have an automated system, you must gather information on the processes handled by that system. It is also important to understand the limitations or problems that the current system has that you need to address with Siebel Workflow Processes.

Areas you might need to research for your current workflow process include:

- Existing process information.
- Measures for improvement or new process requirements.

Researching Existing Process Information

Sources from which existing process information can be derived include:

- Current automated processes
- Management guidelines
- Written guidelines of process rules or approval paths
- Internal procedures, written or unwritten

For example, assume you must document the workflow lifecycle for a new work item, such as a service request, from the moment the service request is initiated to the moment it is finished. Include information about decision points in the process, such as when a service request must be escalated or which approval path an order takes when it is high priority compared with low priority.

Researching New Processes and Areas for Improvement

After you have gathered as much information as you can about existing processes, review the information you have to determine if there are opportunities for improvement in the process or whether a new process is required. Possible requirements for improvement include:

- New management guidelines or business requirements that must be addressed.
- Current problems that must be solved.
- Areas you must make more visible.
- Customer satisfaction issues.
- Workflow processes you must automate.

Determining Workflow Process Requirements

This task is a step in "Process of Analyzing Business Requirements" on page 41.

A workflow process consists of various actions that can meet business requirements. There are many predefined actions that can be used when you build a workflow process. Some example predefined actions include:

- **Notifications.** Sending an email, page, or fax.
- **Siebel Operations.** Inserting or updating information in the Siebel database.
- **Integration Messages.** Requesting to send or receive data from an external system.
- **Assignment.** Requesting Assignment Manager to assign an object.
- **Navigation.** Navigating a user to a specific view through a User Interact Step or a Task UI call.
- **Server Request.** Requesting the Siebel Server Request Broker to run a server process.

Except for Siebel Operations, these actions are invoked by calling a method on a business service. Siebel provides predefined business services so they can be used in workflow processes to implement these actions.

You might identify a specialized action that you are interested in calling in your workflow, such as "calculate credit risk." A specialized action can be added by defining a custom business service. A workflow process can call both a predefined and a custom business service. For more information on defining a custom business service, see *Integration Platform Technologies: Siebel Enterprise Application Integration*.

Identifying an Automation Solution

This task is a step in "Process of Analyzing Business Requirements" on page 41.

Once you have determined process requirements, you can identify the automation solution that meets your business requirements. Table 7 summaries the solutions available for automating processes. When making implementation decisions, it is recommended you consider the advantages and limitations of each solution.

Table 7. Comparison of Siebel Workflow to Other Siebel Automation Solutions

Framework	Advantages	Limitations
Workflow Process	Visual representation of business logic is relatively simple to understand and maintain. Remote synchronous and asynchronous execution allows horizontal scalability and long-running transactions.	The semantics for control are not as rich as with scripting. Specific limitations include: ■ Limited control of flow for iteration through record sets. ■ Limited direct access to object methods.
Workflow Policy	Policies respond to database events regardless of whether they are initiated by an Object Manager component or by a non-Object Manager component. Policies can get higher transaction throughput on a given set of hardware for simple transactions.	Changes to policies can require database downtime to implement. Policies are more difficult to configure that other alternatives. Policies allow a limited range of executable actions.
Siebel Script	Script is familiar to many developers. Script provides a set of semantics. Script is flexible.	Limitations for scripting include: ■ Ease of maintenance. ■ Ease of upgrade. ■ Performance.

Comparing Workflow Processes and Workflow Policies

Table 8 summarizes some common requirements and recommends a workflow process or a workflow policy solution.

Table 8. Comparison of Siebel Workflow to Other Siebel Automation Solutions

Requirement	Recommended Solution	Description
Capture business layer logic.	Workflow Process	Workflow Process Manager and run-time events capture business layer logic.
Capture data layer logic.	Workflow Policy	Workflow Policy Manager captures data layer logic. Data coming into Siebel through the data layer, for example EIM or MQ channels, is not captured through the business layer. This requirement typically indicates a potential candidate for a workflow policy.
Implement features supported by a workflow policy but not a workflow process.	Workflow Policy	A workflow policy can support some features that are not available or would be more difficult to implement with a workflow process. For example, email consolidation, duration, and quantity.
Implement features supported by a workflow process but not a workflow policy.	Workflow Process	A workflow process can provide pause, stop, and error handling capabilities.
Implement complex comparison logic, or flow management.	Workflow Process	A workflow process provides a better platform for development and deployment, complex comparison logic, and flow management, such as IF, THEN, ELSE, or CASE.
Invoke a business service.	Workflow Process	A workflow process can invoke a business service.
Perform bulk data uploads.	Workflow Policy	Workflow Policy Manager is the better alternative when bulk data uploads occur through EIM .
Perform data quality cleaning in the data layer.	Workflow Policy	Workflow Policy Manager is the most appropriate solution for working at the data layer.
Use a repeating component request.	Workflow Process	You can setup a workflow process from a repeating component request but not a workflow policy.

Table 8. Comparison of Siebel Workflow to Other Siebel Automation Solutions

Requirement	Recommended Solution	Description
Repetitive, manual processing.	Workflow Process	The structure of a workflow process provides a superior solution for repetition, timeliness, and cross-functional routing through a business process.
Process an event in a timely fashion.	Workflow Process	
Perform escalations and notifications.	Workflow Process	

For more information, see Chapter 9, "About Workflow Policies".

Process of Planning a Workflow Process

This process is a step in "Roadmap to Developing a Workflow Process" on page 39.

Before building a workflow process in Siebel Tools, it is useful to plan how the workflow is built. This includes making design decisions such as which workflow mode to use, the events to define, the rules to define, actions the workflow process takes, and so forth.

To plan a workflow process, perform the following tasks:

1. Determining the Workflow Mode on page 45
2. Understanding the Workflow Constructs on page 46
3. Determining the Workflow Invocation Event on page 47
4. Determining the Workflow Rule Constructs on page 49
5. Determining the Workflow Action Constructs on page 51
6. Determining Error Handling on page 53
7. Considering a Seed Workflow Process on page 53
8. Considering Object Management in Siebel Tools on page 54
9. Considering Other Design Options on page 55

Determining the Workflow Mode

This task is a step in "Process of Planning a Workflow Process" on page 45.

A workflow process can persist for just a few moments, as in aiding a user in generating an email, or it can span days and job functions, as in a quote to cash workflow. This characterization is handled by the Workflow Mode property on the object definition for the workflow process. For more information, see "About the Workflow Mode Property" on page 121.

Understanding the Workflow Constructs

This task is a step in "Process of Planning a Workflow Process" on page 45.

To plan for building a workflow process, it is important to understand the basic constructs that can be defined. With Siebel Workflow, the run-time engine manages process flow, application flow, and integration flow. Basic constructs such as Start steps, Decision Points, Siebel Operation steps, and Connectors are available to build your workflow process. Some of the basic Siebel workflow constructs are illustrated in Figure 9.

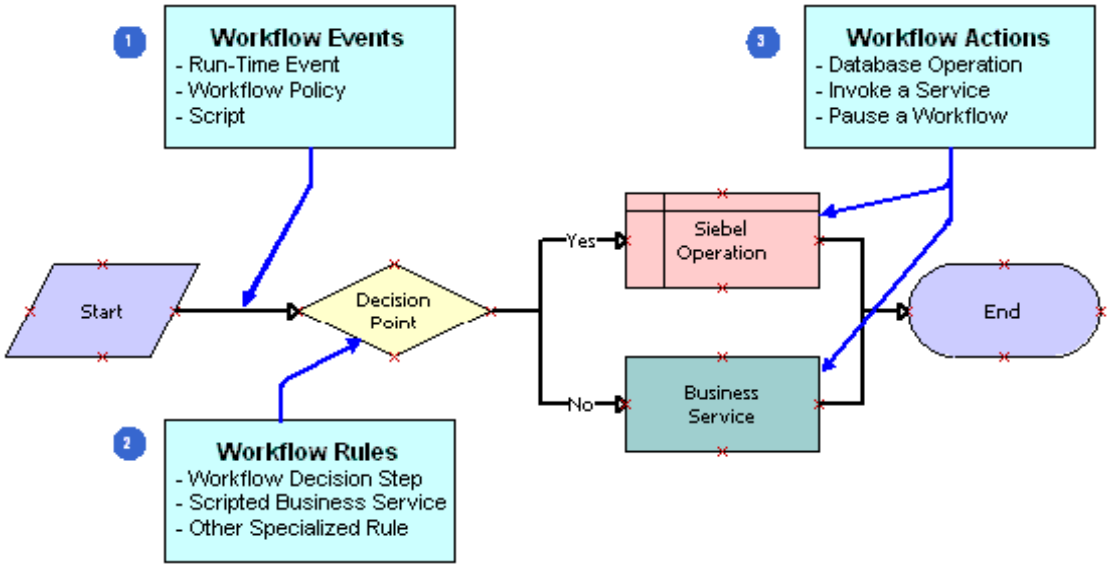


Figure 9. Basic Constructs of a Siebel Workflow Process

The basic constructs of a workflow process include:

1. **Workflow Invocation Events.** A workflow process can be invoked from a run-time event in the Siebel application, a Workflow Policy, or through script. An event can pass context from the caller, for example a user session, to a workflow process using a row id.
2. **Workflow Rules.** The flow control for a workflow process is determined by decision rules. Rules can be based on business component data or on local variables in the workflow process known as a process property. Rule expressions are defined using Siebel Query Language.
3. **Workflow Actions.** A workflow process action can perform database record operations, invoke a business service, or pause workflow process execution.

Determining the Workflow Invocation Event

This task is a step in "Process of Planning a Workflow Process" on page 45.

During the planning phase of a development effort you can determine if the workflow process is invoked by a run-time event, user event, workflow policy, or script.

For more information, see "Invoking a Workflow Process" on page 137.

Using a Workflow Policy to Invoke a Workflow Process

A workflow policy triggers a workflow process after a database change. The basic construct of a policy is a rule. If the conditions of a rule are true, then an action occurs. In some cases, the action invokes the Workflow Process Manager server component to execute a workflow process.

Processing invoked by a workflow policy does not occur in real time. Typical uses of a workflow policy include:

- EIM batch processing.
- Siebel EAI inserts and updates.
- Manual changes from the user interface.
- Assignment Manager assignments.
- Siebel Remote synchronization.

Using an Event to Invoke a Workflow Process

Types of events used by Siebel workflow include:

- **Run-time events.** A run-time event is based in the Object Manager and occurs when a change is encountered by the user interface or in the business component layer. Processing invoked by a run-time event occurs in real time.
- **User events.** A user event is a unique event internal to Workflow that triggers or resumes a long-running workflow process. A user event is generated by the User Event business service.

Events belong to three objects: Application objects, Applet objects, and Business Component objects. Events can be configured from the administrative interface.

Using Script to Invoke a Workflow Process

Scripts can call a workflow process programmatically as a business service. Using script, you can invoke Workflow from an external system. The Workflow Process Manager server component provides APIs for such programmatic invocation. Scripts are raised by the Object Manager.

Scripts belong to four objects: Application, Applet, Business Component, and Business Service.

Summary of Invocation Mechanisms

Table 9 summarizes the main uses and limitations for each of the possible workflow invocation techniques.

Table 9. Description of Advantages and Limitations of Workflow Invocation Techniques

Invocation Method	Most Useful When	Limitations
Workflow Policy	You must detect and react to data changes made outside of the Object Manager. For example, by Siebel Remote or by Siebel EIM.	Limitations for a workflow policy include: ■ Making changes requires database downtime. ■ Relatively complex to configure
Event	You must implement a basic entry point for a workflow or a simple custom action. You must avoid distributing SRFs. For example, because of the burden created for mobile users.	Limitations for an event include: ■ Cannot directly respond to an event by scripting against the event object ■ Can be more difficult to pass the event context to business logic ■ This invocation technique does not trap data changes made by non-Object Manager components
Script	You need flexibility to write script directly in response to an event. You need access to an applet event that is only exposed in Siebel Tools.	Limitations for script include: ■ Changes must be distributed through a new .SRF. ■ Does not detect data changes made by non-Object Manager components. Note that, for workflow, the script is implemented on a Siebel Tools Object Event.

Determining the Workflow Rule Constructs

This task is a step in "Process of Planning a Workflow Process" on page 45.

When planning to build a workflow process, you can determine the general rule constructs that guide the flow of control within the process.

Table 10 describes several ways to implement decision rules in a workflow process.

Table 10. Description of Ways to Implement Rules in a Workflow Process

Type	Description	When Useful	Limitations
Workflow Decision Point	A step in a workflow that arbitrates between one or more alternative branches in the flow. Each connector out of a decision point has one or more conditions. If the conditions evaluate TRUE for the connector, flow proceeds down the branch represented by the connector.	When you need a simple articulation of whether one or more alternative actions in flow must be taken.	Conditional expressions lack support for some key operators, including: ■ AND ■ OR ■ Order of precedence control, as determined by parentheses.
Scripted Business Service	Script within a business service action step that evaluates a potentially complex set of inputs and returns a simplified output that can be evaluated by a workflow decision point.	When Workflow decision point semantics are not sufficiently expressive to encapsulate decision criteria.	Undermines the readability and simplicity of the workflow by hiding logic within a business service.

Table 10. Description of Ways to Implement Rules in a Workflow Process

Type	Description	When Useful	Limitations
Other Specialized Rule Frameworks	Other rule frameworks that can be used directly or indirectly by a workflow, such as personalization rules, assignment rules, EAI Dispatch Service, or the Siebel rules engine. The Siebel rules engine allows you to maintain business process logic declaratively and in a location external to your Siebel applications. For information on implementing rules in a workflow process, see *Siebel Business Rules Administration Guide*.	When it is deemed appropriate to use a specialized rule framework, such as when you must assign work to people based on their workload.	Limitations vary depending on the rule framework used.
Wait Step	A workflow step that puts the workflow into a holding pattern until a releasing event is fired or a timeout occurs.	When you must support time-based escalations or long-running flows that can last for days or weeks. For example, waiting for a customer response.	The releasing event must be triggered through the Object Manager.

The Decision Point

A decision point can exit with multiple connectors that represent logical branches. For each connector that provides branching, a conditional statement is evaluated. A conditional statement makes a comparison between two of the items in the following list:

- Process properties.
- Business component fields.
- Literal values.

The terms of comparison include:

- Two values are equivalent.
- One value exists among a series of others. For example, child record values, One Must Match, or All Must Match.
- Greater Than (>) or Less Than (<).
- Between or Not Between.
- Null or Not Null.

For an example of the Compose Condition Criteria dialog box displaying decision criteria, see "Defining Condition Criteria for the Workflow Process" on page 319. For a description of properties in the Compose Condition Criteria dialog box, see "Defining Conditional Logic on a Branch Connector" on page 114.

The Wait Step

The Wait step allow you to suspend workflow process execution for a specified period of time or until a specific event occurs.

Determining the Workflow Action Constructs

This task is a step in "Process of Planning a Workflow Process" on page 45.

There are several ways to effect actions in a workflow. With a workflow process action, data is taken as an input, then a transformation takes place, and data is produced as output. When planning to build a workflow process, you should be able to identify the main actions the workflow process includes and the type of action constructs that are required.

About Data Manipulation With a Workflow Process

A workflow process operates on business objects and business components. Typically, a workflow process is associated with a single business object. Within the context of these data layer objects, data is created or updated as the workflow process executes. The main types of data a workflow process manipulates include:

- Business component data.
- Process property data.
- Siebel Common Object data.

A process property can be thought of as a local variable that is active during execution of the workflow process instance. The process property can be used as input and output to the various steps in a process. With a workflow process, there is a set of predefined process properties that are automatically generated when you define the workflow. One example of these predefined process properties is the Process Instance Id.

About the Main Action Types In a Workflow Process

Table 11 describes the main Action Types.

Table 11. Description of Actions in Siebel Workflow

Action Type	Description	When Useful	Limitations
Business Service Step	A workflow step that invokes a method of a business service. The business service can be a prebuilt Siebel service or a scripted business service.	When you must execute a potentially complex, but reusable set of logic.	Creating and destroying business services can be expensive. Overhead can be reduced through caching. Incorporating too much logic within a business service can limit the business service's reusability and the transparency of the workflow.
Siebel Operation Step	A workflow step that performs inserts, updates, and queries against Siebel business components.	When you must execute simple record operations within the workflow.	While it is possible to update multiple records based on a search specification, it is not possible to retrieve and iterate through a set of records such that subsequent workflow actions can execute for each record.

The Business Service Step

The Business Service step executes predefined or custom business service methods. Typical predefined business services used include Assignment Manager requests, notification through the Communications Server, server requests, and integration requests from Siebel EAI. Custom business services can be written in Siebel VB or eScript. When defining a Business Service step, you must specify the business service, the business service method, input arguments and output arguments. Input arguments are passed in a process property, business component data, or a literal value.

Some commonly used business services for workflow processes include:

- Outbound Communications Manager.
- Synchronous Assignment Manager Requests.
- Server Requests.
- Business Rule Service.
- Report Business Service.
- Audit Trail Engine.
- EAI business services, such as EAI Siebel Adapter, EAI XML Converter, and so forth.
- FINS Data Transfer Utilities and FINS Validator.

For more information, see "About the Business Service Step" on page 92, and "Predefined Business Services" on page 373.

If your business requirements require specialized functionality, you can create a custom business service for the specific activity. Business services can be defined in Siebel Tools or in the Siebel client administration screens. For more information, see *Integration Platform Technologies: Siebel Enterprise Application Integration*.

The Siebel Operation Step

The Siebel Operation step allows you to perform database operations such as Insert, Update, Upsert, Query, Delete, NextRecord, PreviousRecord, and QueryBiDirectional. The Siebel Operation step is based on a single business component. Once you have defined the Siebel Operation step, you can use the Search Specification child object to locate the records you need to manipulate.

Examples of Siebel Operation steps include creating an Activity record when a new SR is opened, or updating a comment field if an SR is open too long.

For more information, see "About the Siebel Operation Step" on page 97.

Determining Error Handling

This task is a step in "Process of Planning a Workflow Process" on page 45.

When planning a workflow process, you should be able to determine what, if any, error handling is necessary, and the error handling constructs that are required. Error handling options range from a fairly simple solution of using a stop step in a workflow process, to defining a separate error workflow process. Also, the planning phase is an appropriate time to plan for recovering a failed workflow process. For more information, see"About Handling Errors" on page 157.

Considering a Seed Workflow Process

This task is a step in "Process of Planning a Workflow Process" on page 45.

Before building a new workflow process, you should determine that a seed workflow process does not already exist that can meet your business requirements. A *seed workflow process* is a workflow process that comes predefined with the product.

Workflows that appear in the Workflow Processes Object List Editor (OBLE) immediately after Tools is installed, prior to modifications being made, are seed workflows. By default, a seed workflow process is in a Completed state with a version number of 0.

Since many seed workflows are used at run time to execute some of the base functionality of the Siebel application, they should not be edited. To use a modified form of a seed workflow, you can create a copy of the seed workflow then edit the new version or the copy. After revising a seed workflow, the workflow's version changes from 0 to 1. You then modify this version to suit your business needs, publish it, and activate it. Activating a workflow process creates the appropriate run-time events defined within the process.

For more information, see"Modifying a Workflow Process" on page 58.

Considering Object Management in Siebel Tools

This task is a step in "Process of Planning a Workflow Process" on page 45.

When planning a workflow process you can identify project management requirements, such as whether there are special requirements for merging, archiving, importing data maps or copying message tables. If a team of developers is involved in the development environment, you should consider how Project check in and check out is to be used.

When you develop a workflow process in Siebel Tools, you work on a local database where Siebel Workflow is a repository object, and where a workflow process belongs to a project.

Though compiling a Siebel Repository File is a standard practice for other repository objects, Siebel Workflow provides its own deployment mechanism. You do not compile an SRF after you have developed a workflow process. For more information, see "Process of Deploying a Workflow Process" on page 237.

Although these behaviors are standard for other repository objects, the behaviors a workflow process does not participate in include:

- **Merge.** A workflow process does not participate in three-way merge. When a workflow definition is imported into the repository, it maintains versioning provided by Siebel workflow.
- **Object Comparison.** is disabled in Siebel version 8.0.
- **Archive.** A workflow process does not participate in .sif archive. Instead, a workflow can be archived as an XML file using the Workflow export utility.

Typically, developers use a local database to develop workflows. When using a local database, you must check out workflow definitions from the master repository.

When developing workflows on a local database, the local database must have the referenced data objects. For those data objects that are not docked and hence not packaged as part of the database extract, developers must import them into the local database. Objects not docked or referenced by Workflow include:

- **Data maps.** To import data maps to the local database, you use the Siebel Dedicated Web Client connected to the local database, and the client side import utility.
- **Message tables.** You can copy message tables over to the local database. Alternatively, you can define messages using the unbounded picklist. While this technique allows for the creation of messages, it does not check the validity of the message at definition time.

By locking the project in the master repository, you can also develop or modify workflows using Siebel Tools connected to the development database. This way, it is not necessary for you to make sure lists of values are made available to the local database.

Considering Other Design Options

This task is a step in "Process of Planning a Workflow Process" on page 45.

Other design options that should be considered during the planning phase depending on your business requirements include:

- Using the Workflow Process Batch Manager to run a workflow process against records in a business component at predetermined interval. For more information, see "About Batch Processing" on page 164.
- Considering globalization, if the workflow process is implemented in a global environment, see "About Global Implementation" on page 167.

Job Roles Involved in Developing a Workflow Process

The job roles associated with developing a workflow process include:

- The *Business Analyst* considers your organization's business requirements and determines the processes to be automated using the Siebel application.
- The *Workflow Configurator* uses Siebel Tools to build workflow a process and to define objects, business services, and programs.

  Your organization can use the predefined objects, business services, or programs provided in the application. You can also use the Workflow Configurator to define custom objects, business services, and programs in Siebel Tools.

  Note that a Business Service can also be defined in the Siebel client. For more information, see *Integration Platform Technologies: Siebel Enterprise Application Integration*.
- The *Workflow Administrator* monitors workflow processes in the Siebel client using Siebel Workflow. The Workflow Administrator also activates workflow policies by generating database triggers in a script and creating them in the Siebel database. The Workflow Administrator then starts Siebel Server processes that execute workflow processes and policies. This person is typically a system administrator, database administrator, or someone from the Information Services department.
- The *End user* uses the Siebel client and causes workflow processes and policies to execute. This person is typically an employee of your organization, and can also be a customer.

5 Building a Workflow Process

This chapter describes the steps involved in building a workflow process. It includes the following topics:

To view an example that details how these steps are performed, see "Example Workflow Process That Creates an Activity for a Sales Representative" on page 315.

Process of Building a Workflow Process

This process is a step in "Roadmap to Developing a Workflow Process" on page 39.

To build a workflow process, perform the following tasks:

1. Defining a Workflow Process Object Definition on page 57
2. Diagramming a Workflow Process on page 61
3. Defining Process Properties for a Workflow Process on page 62
4. Defining Step Properties for a Workflow Process on page 63

Defining a Workflow Process Object Definition

This task is a step in "Process of Building a Workflow Process" on page 57.

The first part of building a workflow process is to define the top-level workflow process object definition. This topic describes several possible options. For more information about how objects discussed in this topic are represented in Tools, see "About the Workflow Process Object Hierarchy" on page 65.

Reviewing Workflow Processes

Review your existing workflow processes to see if the process you need is already available or if a similar process exists that you can copy and modify to meet your requirements.

The Workflow Processes Object List Editor (OBLE) provides a list of the current process definitions. You can access the Workflow Processes OBLE in Siebel Tools.

To review existing workflow process definitions

1. In Siebel Tools, in the Object Explorer, choose the Workflow Process object type.

   The right pane displays the Workflow Processes OBLE, which lists the workflow process object definitions.

2. Examine the list of existing workflow processes, perusing the Process Name, Business Object and Workflow Mode properties for a combination of properties that meet your business requirements.

3. If you find a workflow process that appears to be a potential candidate, right-click the record in the Workflow Processes OBLE, choose Edit Workflow Process, then use the Process Designer to examine the flow of steps as well as process and step properties.

4. If you find a workflow process you need to copy as the basis for a new process, right-click the process then choose Copy Record.

   For more information, see "Copying a Workflow Process" on page 58.

Copying a Workflow Process

If your intent is to create a different workflow, with a different name and a different purpose, then you can copy an existing workflow. The copied workflow does not replace the original workflow.

In contrast to revising a workflow, when you copy a workflow, you create a new workflow process that is identical to the original one, except that a unique name, such as 04-K88GQ, is generated for it.

The version of a new, *copied* workflow process is 0. When you *revise* a workflow process, however, the version is the next integer that is subsequent to the version number of the original workflow being revised.

To copy a workflow process

1. In the Object Explorer, choose the Workflow Process object type.

2. Right-click the workflow process in the OBLE, then chose Copy Record.

   A new copy of the workflow process is created for editing, and appears in the Workflow Processes OBLE with a unique numeric identifier appearing in the Process Name property.

3. Update the Process Name property so it is meaningful.

4. Modify the other properties as necessary for the new process.

Modifying a Workflow Process

You can modify an existing workflow process without restarting the Workflow Process Manager.

A workflow process definition is refreshed in the process memory as soon as it is activated. When a new definition is deployed, the cache is refreshed, so a new instance picks up the newly deployed definition.

To modify a workflow process, you must make it editable. For more information, see "Making a Workflow Process Editable" on page 60.

About Workflow Process Definition Cache Refresh

The timing of when a workflow process definition is activated impacts the process cache:

- The caches of threads in the same process are refreshed and these threads get the updated workflow process definition immediately.
- The caches of threads in other processes are not refreshed until the interval defined by the server parameter VerCheckTime expires.

For mobile clients and development clients, the server parameter called Workflow Version Checking Interval (VerCheckTime) controls how often the server component checks for new active versions of each workflow process definition. When a new workflow process definition is activated, an incoming workflow process instance created during the interval determined by Workflow Version Checking Interval uses the new definition. A workflow process instance initiated before the interval continues using the previous workflow process definition.

Scenarios to consider include:

- **Activation in thin clients.** When a workflow process is activated from a thin client, such as the Call Center application, user sessions in the *same* Call Center application server process get the updated workflow process definition immediately. User sessions in *other* server processes, however, must wait for the VerCheckTime interval defined in the server process configuration before they receive the updated definition.
- **Activation in mobile or development clients.** Each mobile or development client is a process of its own. The particular mobile/development client receives the updated workflow definition immediately. *Other* mobile/development clients must wait for the VerCheckTime interval defined in their configuration files before they can receive the updated definition.
- **Publish / Activate in Siebel Tools.** The Siebel Tools application is a process of its own. Thin clients and mobile/development clients must wait for the VerCheckTime interval before they can receive the updated definition.

Reloading Run-Time Events

Refreshing the process definition cache is necessary but not sufficient for running the updated workflow process correctly. If the workflow process contains run-time events, the run-time event cache must also be refreshed:

- To reload run-time events for thin clients, navigate from the Administration-Runtime Events view. Right-click the context menu, then choose Reload Runtime Events.
- To reload run-time events for mobile clients, log out then log back in to the mobile client. This guarantees a refresh of the process definition cache as well.

Revising a Workflow Process

To revise a workflow you modify an existing workflow process definition. When you revise a workflow, the new workflow record created is the same as the original, with the same name, except that the workflow's version is incremented by one, the Status is In Progress rather than Completed, and the workflow is editable. The revised workflow replaces the original workflow.

To revise a workflow process

1. In Siebel Tools, in the Workflow Processes OBLE, choose the definition for the workflow process you need to modify.
2. In the WF/Task Editor toolbar, click the Revise button.

   For more information, see "About the WF/Task Editor Toolbar" on page 72.
3. With this new process record chosen, right-click then choose Edit Workflow Process.
4. Modify this new version, as necessary.

If necessary, you can use the Expire button to set a workflow's status to Not In Use. For more information, see "Expiring a Workflow Process Instance" on page 256.

To make a workflow editable

1. Make sure the workflow process is either checked out or is locked by the developer who is currently attempting to edit the workflow process.
2. Make sure the project specified in the Project property for the workflow process is locked.

Defining a New Workflow Process

If you cannot locate an existing workflow process that meets your needs, you can define a new one. To view an example, see "Creating a New Workflow Process Object Definition" on page 316.

Naming A Workflow Process

When naming a workflow process, the combination of workflow process name and version must be unique. That is, you can have two workflow processes of the same name as long as their version numbers are different.

Externalizing Workflow Properties

When developing a workflow process, it is recommended you define properties for the workflow that are externalized and not hard coded. Having properties hard coded in a workflow means that you must change the definitions when the workflow is deployed between environments. For example, if a workflow is sending emails to a list of customers and the property is hard coded with the customer list, the workflow will not execute correctly in the production environment. You must make sure such input arguments are read dynamically.

For an example of how to externalize properties used by a workflow process, see "Example of Externalizing Properties Used by Siebel Workflow" on page 361.

Making a Workflow Process Editable

The background color in the workflow editor indicates whether the workflow process is editable:

- A yellow background indicates the workflow cannot be edited.
- A white background with a grid indicates the workflow can be edited.

When a workflow is editable, you can modify the workflow in a variety of ways, such as adding and removing steps and connectors, or changing step and connector properties. Several conditions must exist to make a workflow editable.

Deleting a Workflow Process

You can delete a workflow process in the Object List Editor in Siebel Tools.

To delete a workflow process

1. In the Object Explorer, choose the Workflow Process object type.
2. In the Workflow Processes OBLE, choose the workflow process you need to delete.
3. Right-click then choose Delete Record.

   Note that you cannot delete a seed workflow process. For more information, see "Considering a Seed Workflow Process" on page 53.

Diagramming a Workflow Process

This task is a step in "Process of Building a Workflow Process" on page 57.

Diagramming process steps is an important part of creating a functioning workflow process. The flowchart interface of the Process Designer allows you to build a visual representation of the entire process flow, including decision points and conditional logic.

You can choose to define the details for each step as you create each step in the Workflow Designer, or you can finish the entire flow for the process then enter details for each step.

Based on your planning results, you can diagram the steps of a workflow process.

To diagram the steps of a workflow process

1. Add a Start step to the design canvas.

   A workflow process must have one and only one Start step. For more information, see "About the Start Step" on page 91.

2. Add one or more middle steps to the design canvas.

   A workflow process can have one or more of the action step types, such as Business Service, Decision, Sub Process, Task, Stop, Wait, Exceptions, and Siebel Operation. There can be multiples of each type of step. For more information, see "About Workflow Process Step Types" on page 90.

3. Add an End step to the diagram area.

   A workflow process must have at least one End step. Details on defining an End step are in "About the End Step" on page 111.

4 Define the flow and path of the process by dragging and dropping a connector from the palette to the canvas. Position one end of the connector on one side of a step, then drag the connector's handles to connect the other end of the connector to the next step in the flow.

   NOTE: A connector's end point that is colored white is not correctly connected to a step. A red end point indicates the connector is correctly connected. Make sure both ends of every connector is red.

5 Repeat Step 4 until every step in the workflow process is connected correctly.

Note that a connector emanating from certain step types can provide conditional logic for the workflow process. For information about defining a connector that provides conditional logic, see "Defining a Branch Connector" on page 112.

Adding or Removing a Point in a Connector

To add or remove a point in a connector, use the following steps:

1 Click the connector or exception.

2 Right-click, then choose one of the following options:

- Edit > Add Point.
- Edit > Remove Point.

Defining Process Properties for a Workflow Process

This task is a step in "Process of Building a Workflow Process" on page 57.

You can define a Process Property in either the Multi Value Property Window (MVPW) or in the WF Process Props OBLE.

You can define process properties step-by-step as you diagram a workflow process, or you can diagram the entire workflow process, then define properties for individual steps. However, it is recommended you define process properties before defining an input or output argument on an individual step since some of those arguments can reference a process property. For more information, see "About Process Properties" on page 74.

To define a process property through the MVPW

1 Open Siebel Tools.

2 In the Workflow Processes OBLE, locate the object definition to which you need to define a process property, right-click the record, then choose Edit Workflow Process.

3 From the application-level menu, choose View > Windows > Multi Value Property Window.

   For more information, see "About the Multi Value Property Window" on page 77.

4 Click the canvas background, making sure no workflow process step or connector is chosen.

5 In the MVPW, make sure the Process Properties tab is chosen.

6 In the MVPW, right-click anywhere in the window below the Process Properties tab, then choose New Record.

7 Enter a name for the process property.

For more information, see "About Naming a Process Step or Process Property" on page 88.

8 Configure the Data Type argument field, as described in the following table:

Data Type	Description
String	If the property holds a character value.
Number	If the property holds a numeric value.
Date	If the property holds a date value.
Hierarchy	If the property holds hierarchical data, as in a property set.
Binary	If the property holds a binary value. Binary data is encoded in the original character set it was entered in. This data type code is appropriate for self describing data, such as XML.
Integration Object	If the property holds an integration object. The Siebel .srf uses the integration object definition for this object.
Strongly Typed Integration Obj	If the property holds a strongly typed integration object.
Alias	If the property holds an XPath notation for pointing to a child in a hierarchical process property.

The String data type code is appropriate for strings, text data, and non XML data, even though XML data can be stored here. String data in a workflow process is assumed to be UTF 16 String encoded. Literal or Expression data types are always UTF 16 String.

NOTE: Once a data type is chosen, it cannot be modified.

9 Define the Default String, Default Date, or Default Number property, if applicable.

The value you define in one of these properties is the value used at the start of workflow process execution. Note that you only define one of these default values if its corresponding data type is chosen. For example, if you specify the Date data type, then you can specify a Default Date.

10 Repeat Step 5 through Step 9 to define more process properties, as necessary.

Defining Step Properties for a Workflow Process

This task is a step in "Process of Building a Workflow Process" on page 57.

A workflow step's properties can include input and output arguments, search specifications, operations, and so forth. You can define some of these properties by using the WF Steps OBLE. You can also use the Properties window and the MVPW from within the Process Designer to define the step's properties. In any case, a step's properties must be defined one step at a time.

To define step properties

1. With the Process Designer open, click the step whose properties are to be defined.
2. Use the Properties window to define the step's basic properties.

   If the Properties window is not displayed, right-click then choose View Properties Window. For more information, see "About Workflow Process Steps and Connectors" on page 87.
3. Use the MVPW to define input arguments, output arguments, and search specifications.

   For more information, see "About the Multi Value Property Window" on page 77.

6 About Siebel Tools and Process Properties

This chapter describes the steps involved in building a workflow process. It includes the following topics:

About the Workflow Process Object Hierarchy

The object hierarchy used with workflow objects can be viewed in Siebel Tools. You use Siebel Tools to modify standard Siebel objects and create new objects to meet your organization's business requirements. Just as you use Siebel Tools to extend the data model, modify business logic, and define the user interface, you also use Siebel Tools to configure the workflows that the Siebel application uses to automate your organization's business processes.

Siebel Tools consists of an Object Explorer (OE) window and one or more Object List Editor (OBLE) windows, as illustrated in Figure 10.

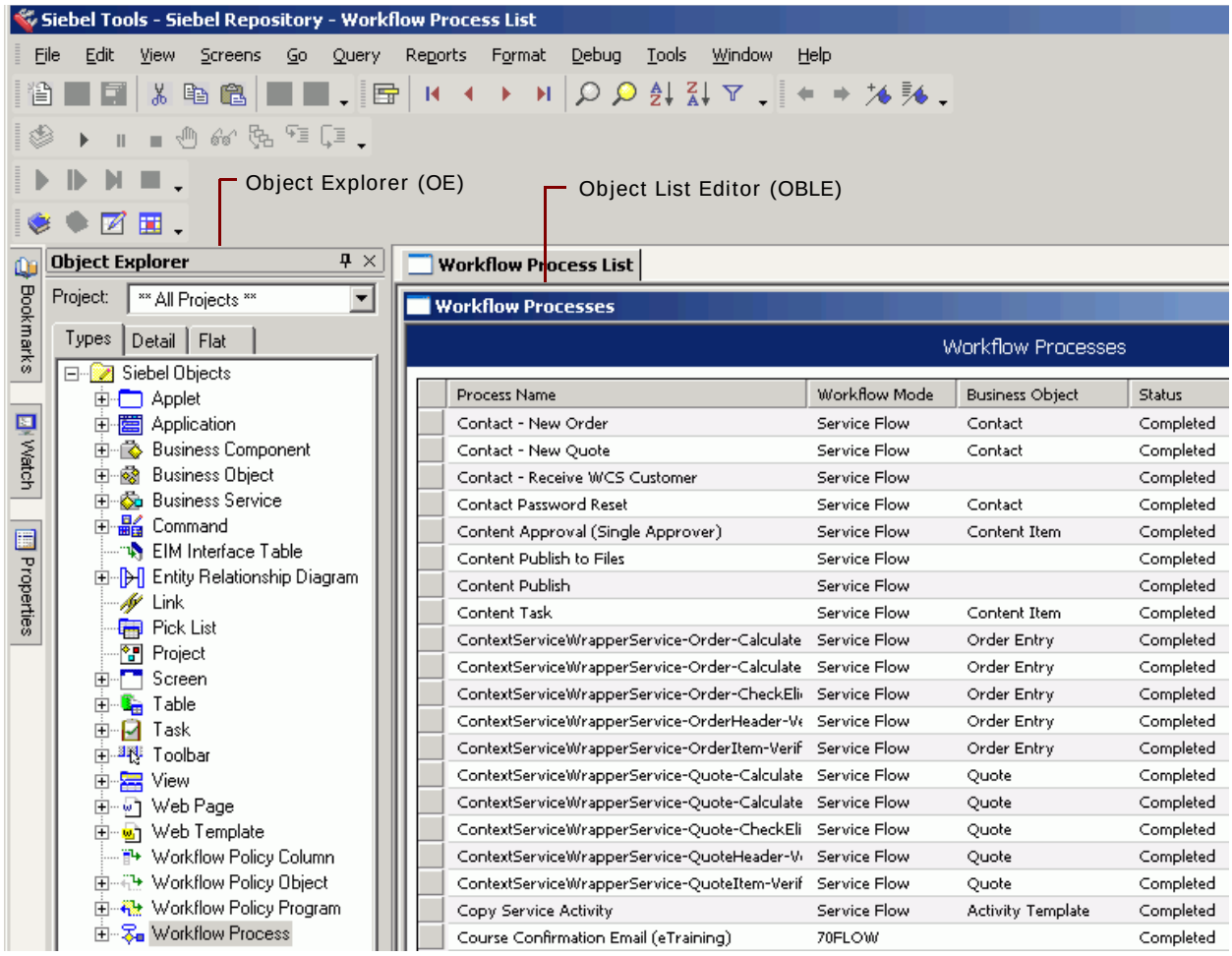


Figure 10. Object Explorer and Object List Editor Windows

The Object Explorer provides navigation between each group of object definitions of a particular object type.

An *Object Type* is an entity that is displayed as a node on the Object Explorer. An object type has a predefined set of properties, and is used as the template from which object definitions are created. Workflow Process is one object type.

An *Object Definition* implements one piece of the software. This object definition consists of *Object Properties*, which are characteristics of that piece of the software.

If the Workflow Process object type is not visible in the Object Explorer, you must expose it.

To expose the Workflow Process object type

1. From the Tools application-level menu, choose View > Options.
2. Click the Object Explorer tab.
3. Scroll down through the Object Explorer Hierarchy window, make sure the Workflow Process hierarchy has a checkmark, then click OK.

About Object Properties

When the Workflow Process object type is chosen in the OE, the right pane displays the Workflow Processes Object List Editor (OBLE) that contains a list of workflow processes currently defined in the srf. One row in the OBLE represents one object definition. For example, values in the properties in the workflow process named Contact-New Order constitute one object definition. Properties of the object definition can be edited in the OBLE.

Object properties are represented as columns in the Object List Editor window. Information entered in the columns of an OBLE window is *values*. You edit properties of the currently chosen object definition in an OBLE window by changing values in the columns. You can change the property values in an object definition. You cannot change the set of properties that constitute an object definition.

You can also use the Properties window to edit the properties of the currently chosen object definition. Changing a property value in the Properties window also changes the corresponding value of that property in the Object List Editor window, and conversely.

About the Relationships Between Workflow Process Object Types

An Object Type can have a hierarchical relationship with another Object Type. This condition is called a parent-child relationship. The parent-child relationship is represented in the Object Explorer (OE) as a hierarchical tree, as illustrated in Figure 11.

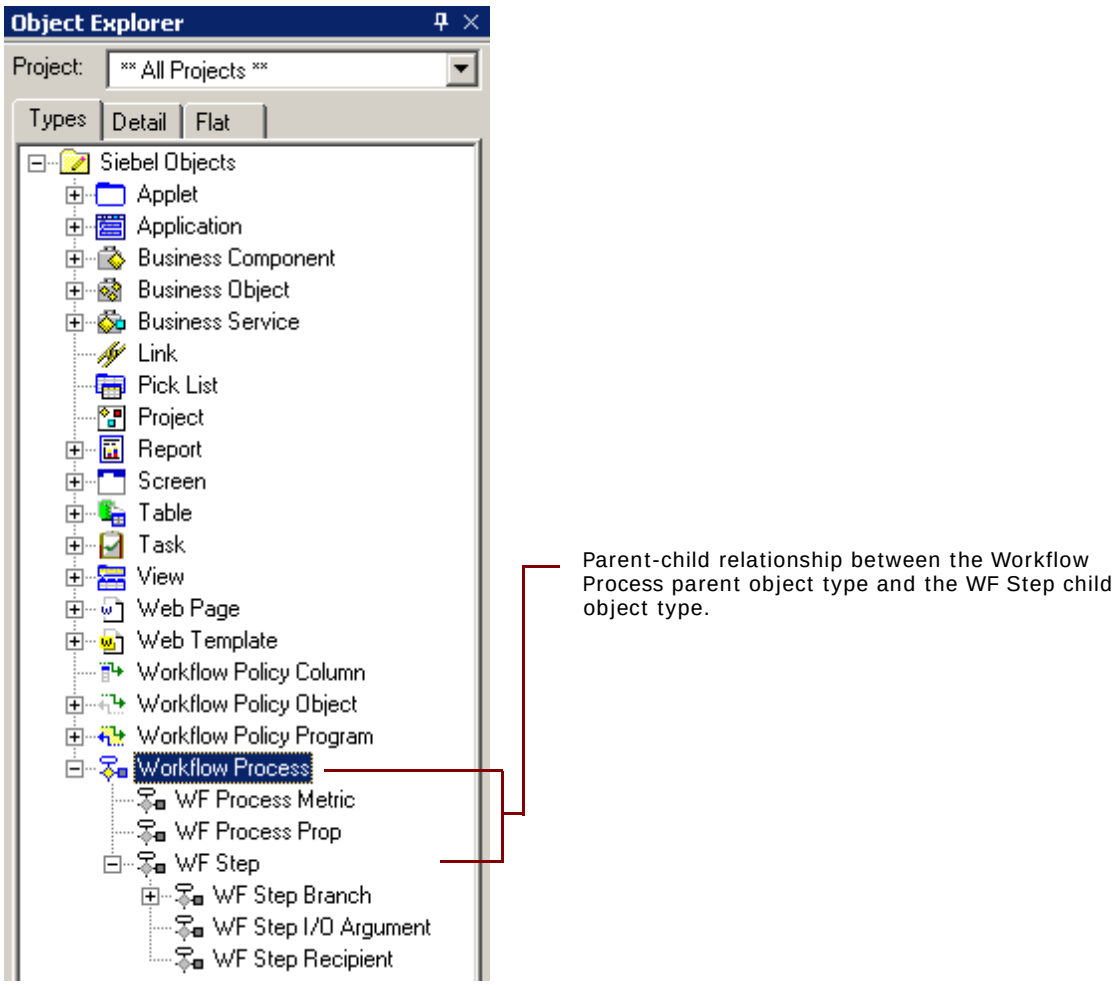


Figure 11. Parent-Child Object Type Relationship in the Object Explorer

When the OE's Types tab is chosen, the arrangement of folder icons is hierarchical. An object type that is beneath and slightly to the right of another object type is the child of the parent-child relationship. The object type that is above the *child object type* is the child's *parent object type*.

A parent object type can have multiple child object types. For example, WF Process Metric, WF Process Prop, and WF Step is each a child object type of the Workflow Process object type.

This workflow guide assumes that you understand the basics of using Siebel Tools. The information you should know includes:

- Using Siebel Tools application window usage, primarily the Object Explorer and the Object List Editor.
- Configuring object properties, applets, and applet controls.
- Using the Siebel Tools menu bar.
- Checking out and checking in projects.
- Working with projects.

For more information, see *Using Siebel Tools*, and *Configuring Siebel Business Applications*.

About the Process Designer

This topic describes the Process Designer used to configure a workflow in Siebel Tools. It includes the following topics:

About the Process Designer GUI

Figure 12 illustrates the main elements in the Process Designer GUI.

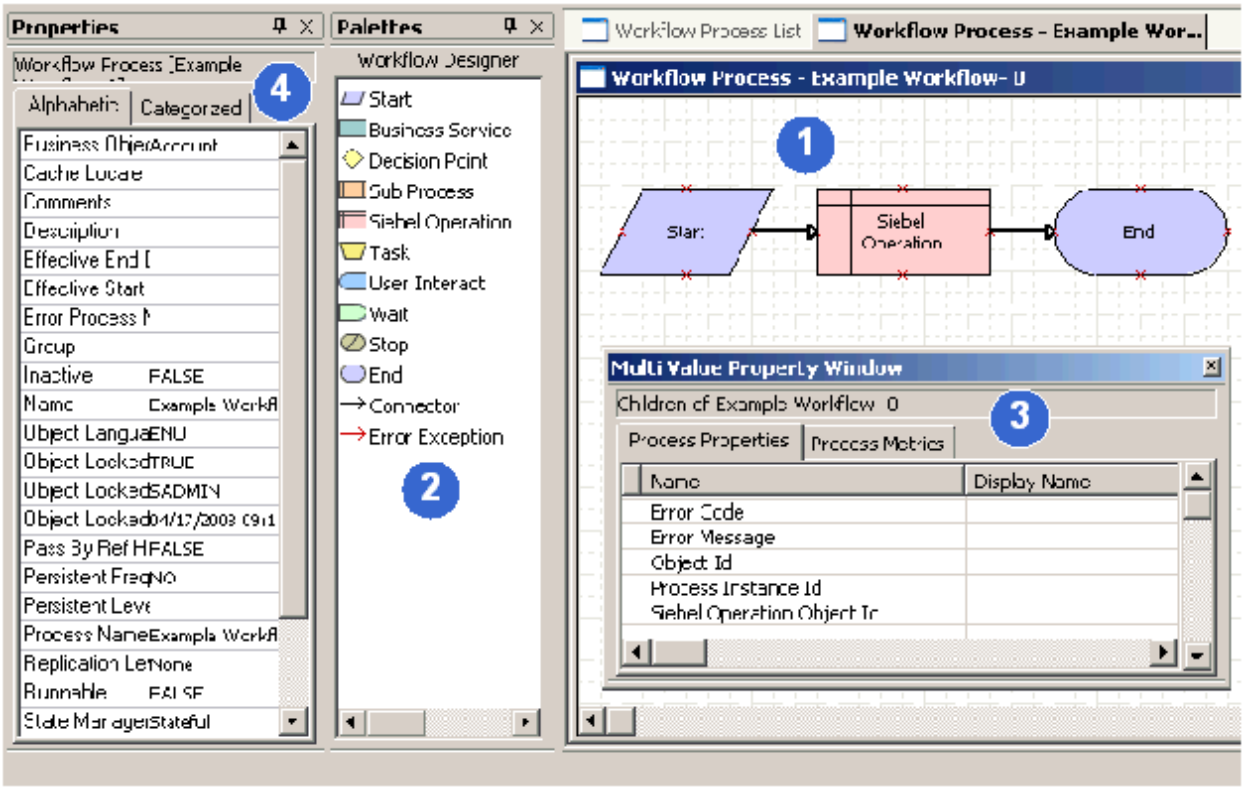


Figure 12. Screen Capture of Process Designer Graphical User Interface

The Process Designer includes elements you use frequently when building a workflow process:

1. **Process Designer Canvas**. A work area where you build the workflow process. You right-click the canvas to access a context-sensitive menu that allows you to take several actions related to the workflow process displayed in the canvas.
2. **Workflow Designer Palette**. A window that contains icons which represent the various step types you can add to a workflow process. To add a step to a workflow, you drag then drop an icon from the palette to the design canvas.
3. **Multi Value Property Window (MVPW)**. A window that allows you to define arguments used with a process property. For more information, see "About the Multi Value Property Window" on page 77.

4 **Properties Window**. A window that allows you to define properties for an individual workflow process step, or for the overall workflow process. The window is context-sensitive. If a step or connector is chosen in the canvas, properties for the step or connector are displayed in the Properties window. If no step or connector is chosen in the canvas, properties for the workflow process are displayed in the Properties window. For more information, see "About Workflow Steps" on page 87.

About Process Designer Functionality

You use the Process Designer in Siebel Tools to build your workflow process. While diagramming a workflow process in the Process Designer, tasks you can perform include:

- **Copy and paste.** Copy and paste objects in the Workflow Designer canvas as you are building a workflow process.
- **Edit shape properties and layout.** Define shape colors and other attributes such as the look of the line, the fill pattern, and the font for labels. Create consistency by controlling alignment of shapes and by making shapes the same size as other shapes in the workflow process.
- **Zoom.** Zoom in and out on the Process Designer canvas to view the workflow process you are diagramming at various magnifications.
- **Copy drawings.** Copy a workflow diagram into another application, such as a Microsoft Word document. In the canvas, you right-click and choose Copy Drawing.
- **Print.** Print a diagram of the workflow process.
- **Hide connector and error exception names.** You can choose to hide the names of connectors and error exceptions within a workflow process. Hiding a connector name can be helpful to clarify the meaning of conditional branching emanating from a Start step or a Decision Point.

The copy and paste functionality works as Windows applications do with the CTRL+C and CTRL+V key combinations. The rest of the design functionality is employed by right-clicking the Process Designer Canvas.

About the Data Available for Configuration

Because the Process Designer is in Siebel Tools, data objects are available for use as you design your workflow process. Changes to repository data are immediately available for use in a workflow process without the need for a compile of the SRF. You can use configuration data, such as business component fields and other repository information, while you are building your workflow.

For example, if you have a List of Values (LOV) such as Account Status with values of Gold, Silver, and Bronze, you can use a newly added LOV in the definition of your workflow process conditions as you are defining them. As another example, if you add a field to a business component, that new field is readily available for use in the Process Designer.

The type of data that is not available for use in designing a workflow process is the run-time data such as an account name or a ZIP code and other transactional data. To use run-time data as you are building a workflow process, make the data available through a process property. If necessary, you can also hardcode run-time data into your workflow process by using unbounded picklists. For more information, see "About Process Properties" on page 74.

About the WF/Task Editor Toolbar

The WF/Task Editor Toolbar is used to manage deployment of a workflow process. Upon first use, you must expose the WF/Task Editor Toolbar. From the application-level menu, choose View > Toolbars > WF/Task Editor Toolbar. To identify a button in the WF/Task Editor Toolbar, position your mouse over an icon in the toolbar, then note the button's name appears in a small pop-up message.

Table 12 describes buttons on the WF/Task Editor Toolbar.

Table 12. Description of Buttons on the WF/Task Editor Toolbar

Button Name	Description
Publish	Moves the workflow process definition from the repository tables into the Siebel client run-time tables. When you click the publish button, the workflow's status changes from In Progress to Completed and is available: ■ If you are connected to the server data source, the finished workflow process is available at run time to be activated. ■ If you are connected to the local data source, check in the workflow process. After you check in the workflow, it is available at run time to be activated.
Publish/Activate	Publishes and activates a workflow process with a single button click from within Siebel Tools while in local mode. To use this feature, you must also set the [Workflow] VerCheckTime parameter to -1 for the Siebel client on which you are testing. .
Revise	Creates a new version of the workflow process for editing. The new version appears in the Workflow Processes OBLE with its version property incremented by 1.
Expire	Expires a workflow process.

For procedural information on publishing and activating a workflow process, see "Process of Deploying a Workflow Process" on page 237.

Manual Deployment of a Workflow Process

You can publish a workflow manually from within Siebel Tools or from within the Siebel client. To publish a workflow manually in Siebel Tools, you use the Publish button or the Publish/Activate button found in the WF/Task Editor toolbar.

The Publish/Activate button allows you to both deploy and activate a workflow process with one click. If you use the Publish/Activate button rather than the Publish button to publish the workflow process while in local mode, it is not necessary to separately activate the workflow in the Siebel client.

About the Activate Button in the Siebel Client

If you choose to publish a workflow but not activate it, you can use the Siebel client to activate the workflow by clicking the Activate button in the Workflow Deployment view.

The Activate button chooses the active version of the workflow process in the run-time tables. Only one version of a workflow process can be active at a time for a Siebel Enterprise.

Activate provides version control to address an environment where multiple developers can publish multiple versions of the same workflow process. Activation occurs from within the Siebel client, where multi-select is supported.

Validation of a Workflow Process

When you click the Publish button or the Publish/Activate button, the workflow process is validated before it is published. If there are validation errors for the workflow, a dialog box appears, giving you the opportunity to correct errors before publishing.

This dialog box is modeless, meaning that you can keep it open to see the error messages while working on the workflow using the Process Designer to correct the problems reported. However, you can proceed with deployment despite validation errors if you choose to do so.

Deployment Changes Beginning with Siebel Version 8.0

A workflow process is deployed by using buttons in Siebel Tools called *Publish* and *Publish/Activate*. In earlier releases, the button was called *Deploy*, and activation can only take place in the Siebel client. Starting with Siebel version 8.0, you can activate a workflow in Siebel Tools, given you have set the VerCheckTime parameter in the Workflow section of the .cfg file to -1 ([Workflow] VerCheckTime = -1).

If there are no validation errors, you do not see this dialog box, nor do you see a message stating that the validation was successful. The validation is carried out without user visibility, unless errors are detected.

Comparing Copying to Revising a Workflow Process

In some cases, you can chose to copy an existing workflow process rather than revising it. The copy feature allows you to continue to work on the original workflow process should you need to use at some future point. To view an example, see “Copying then Modifying an Existing Workflow Process” on page 358.

About Process Properties

This topic provides information about process properties. It includes the following topics:

A *process property* is a container that stores values you can use in steps as input arguments, output arguments, or for performing evaluations. Process properties store values the workflow process retrieves from the database or derives before or during processing. You can base decision branches on the values in a process property and pass process properties as step arguments. When a workflow process finishes, the final result of the process properties are available as output arguments. You can also use process property values in expressions.

For more about:

- How a process property is defined in Siebel Tools, see "Defining Process Properties for a Workflow Process" on page 62.
- An example that uses process properties, see "Example Workflow Process That Traverses a Record Set to Close Obsolete Service Requests" on page 324.
- Process properties, see "Reference of Process Properties and Arguments" on page 436.
- A detailed description of property sets, see *Integration Platform Technologies: Siebel Enterprise Application Integration*.

About Process Properties and Property Sets

Siebel workflow uses a structure known as the property set to pass data to and from a workflow process step. A *property set* is a hierarchical structure that contains name/value pairs, known as properties, at each level in the hierarchy. A process property is a container used to store data associated with the property set.

Figure 13 illustrates how a process property works within a workflow process.

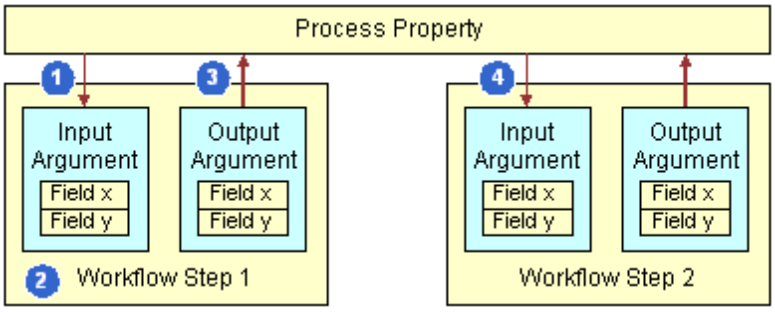


Figure 13. How Process Properties Are Used Within a Workflow Process

Example usage within a workflow process is described in the following sequence:

1. A process property is used to send data to Workflow Step 1 as an input argument.
2. Workflow Process Step 1 manipulates the data according to its internal configuration.
3. An output argument on Workflow Step 1 sends data from the step to the process property.
4. An input argument on Workflow Step 2 provides a way to bring the data manipulated in Step 2 into Workflow Step 2 where it can be manipulated according to the internal configuration defined for workflow Step 2.

The process property therefore manages property sets in two ways, as described in Table 13.

Table 13. Description of How the Process Property Is Used

Construct	Description	How Defined
Process Property	Used to store values that apply to the entire workflow process.	Click the canvas background, then view the MVPW.
Step Argument	Used to communicate information between a process property and an individual workflow process step.	Click a workflow process step, then view the MVPW.

Initial Process Property Values

When a workflow process is started, process properties of type string, number, or date with an In/Out type of In or In/Out are initialized to the input property with the same name, if one exists. Hierarchy process properties with an In/Out type of In or In/Out are initialized with child input property sets that have a matching name. Process properties with their Default String set to [Value] are initialized with the value in the Value property of the input property set.

Concluded Process Property Values

When a workflow process concludes, process properties of type string, number, or date with an In/Out type of In or In/Out are stored as properties in the output property set. Hierarchy process properties with an In/Out type of In or In/Out are stored as child property sets. If a process property with the name [Value] is defined, the property's value is stored in the Value property of the output property set.

NOTE: The process property [Value] can also be used to pass binary data in a Sub Process step. The subprocess input argument [Value] can be initialized with data in the main process then retrieved in the Sub Process step.

For more information, see "Passing Data To and From a Workflow Process" on page 416.

About Arguments for a Workflow Process Step

This topic describes arguments that can be defined in the Multi Value Properties Window for several types of workflow process steps.

An *argument* is a part of the process property hierarchy that allows you to define values for argument fields. An *argument field* is a variable on an argument that allows you to specify values that shape the parameters for a process property.

Table 14 describes arguments that can be defined in the MVPW to pass data to and from an individual workflow process step.

Table 14. Description of Arguments That Can be Defined on a Workflow Process Step

Argument	Description
Input Argument	Used to pass an argument to a workflow step.
Output Argument	Used to retrieve an argument from a workflow step.
Search Specification	Used to specify a search specification for a Siebel Operation step.
Recipient	Used to specify a recipient for a Task or Sub Process step.

A given workflow step can have several types of arguments. The type of step determines the possible arguments that can be defined.

Input Arguments for the Business Service Step, Sub Process Step, and Wait Step

An input argument allows you to define values for argument fields that you need to pass to a service method. Many methods require input arguments. For reference information, see "Argument Fields for an Input Argument" on page 439.

Output Arguments for the Business Service Step, Sub Process Step, and Siebel Operation Step

An Output argument is the result of a business service method. An output argument can be stored in a process property. Calculated fields are not available as values for input or output arguments. If you need to use a calculated value, use an expression. For reference information, see "Argument Fields for an Output Argument" on page 440.

Process Properties and the Business Service Step

Process properties with the business service have a data type of hierarchy, integration object, or strongly typed integration object and can be used as input and output arguments for business service method arguments that have a data type of hierarchy, integration object, or strongly typed integration object.

When you need to call a workflow as a business service, you can map the data contained in the input and output property sets to and from process properties. This is useful when you need to run a workflow within a script.

About the Multi Value Property Window

The ways in which the Multi Value Property Window (MVPW) is used include:

- To view and configure properties that apply to the entire workflow process. These are called *process properties*.
- Provide *arguments* as to what information should be passed in and out of a workflow process step.

Figure 12 on page 70 illustrates the MVPW displaying predefined process properties, as it appears when the canvas is first opened for editing a new workflow process.

To view an example that uses the MVPW, see "Defining the Workflow Process" on page 326.

Process Properties in the MVPW

When you click the canvas with no step or connector chosen, the child objects of the overall workflow process are displayed. The Process Properties tab in the MVPW displays the workflow's process property definitions.

To view process properties associated with a workflow process

1 Click anywhere in the design canvas of a workflow process, with no steps or connectors chosen.

   If necessary, expose the MVPW by choosing View > Windows > Multi Value Property Window from the Tools application-level menu.

2 With the Process Properties tab chosen, examine the records displayed in the MVPW.

Step Arguments in the MVPW

When you click a step in the process designer canvas, the step's input and output arguments are displayed in the MVPW. Depending on the type of step chosen, the MVPW displays one or a combination of tabs.

To view step arguments displayed through the MVPW

1 Click a step in the design canvas of a workflow process.

   If necessary, expose the MVPW by choosing View > Windows > Multi Value Property Window from the Tools application-level menu.

2 Choose the required tab in the MVPW, then examine the arguments displayed.

Defining the Type Argument Field in the MVPW

When you pick the Type argument field in the MVPW, other argument fields must be specified, depending on the value you pick. Table 15 describes the argument fields that must be specified.

Table 15. Description of Defining The Type Argument Field in the MVPW

If The Type Argument Field is. . .	. . .Then Define This Argument Field. . .	. . .With This Value
Literal	Value	Type in a string.
Process Property	Property Name	Choose a property from the pick applet.
Business Component	Argument fields you must define include: ■ Business Component Name ■ Business Component Field	In the Business Component Name argument field, choose a business component from the picklist then, in the Business Component Field argument field, choose a business component field from the picklist.
Expression	Value	Type in an expression.

Changes for Siebel Version 8.0

With Siebel version 8.0, the Multi Value Properties Window (MVPW) replaces the list applet in most cases in the Process Designer. Rather than viewing a step's properties by clicking the step then accessing a list applet below the canvas, you view the step's properties in the Properties window, and the step's child items in the MVPW. Rather than right-clicking a step to see the step's input and output arguments, you view them in the MVPW. You can add and delete values from within the MVPW in the same manner as was performed in previous releases in the list applets.

Predefined Process Properties

Some default process properties automatically defined for each workflow process include:

- **Object Id.** The Siebel row ID of the record against which the process is invoked.
- **Error Code.** An error symbol of the process instance if a step returns an error. This process property is automatically populated when an error occurs.
- **Error Message.** An error message text of the process instance if a step returns an error. This process property is automatically populated when an error occurs.
- **Siebel Operation Object Id.** The Siebel row ID of the record that is updated, created, or queried on during a Siebel Operation step. This process property is automatically populated when a Siebel Operation step is executed.
- **Process Instance Id.** A unique number assigned to the currently running instance of the process. This process property is automatically populated when a process is executed and persistence is activated.

About the Object Id Process Property

The Object Id process property is the Siebel Row Id of the primary business component record against which a process is invoked.

Run-Time Event Behavior

When a workflow is invoked by a run-time event, the active business component instance that triggered the run-time event is automatically passed to workflow. Run-time events are received on the row specified by the Object Id property. Workflow receives and processes this event only when the business object that the active business component instance belongs to is the same as the business object in the workflow definition. If workflow can receive this event, the Object Id process property is automatically set to the active row of the *primary* business record in the business object.

If the business component that actually triggered the run-time event is not the primary business component, the active row of the business component is not reflected in the Object Id process property, and it needs to be retrieved through some extra processing.

Long-Running, Interactive, and Service Workflow Behavior

Behavior for the long-running, interactive, and service workflow processes include:

- The Object Id must match the Row ID of the primary business component's active row. The Workflow engine does not allow the active row of the primary business component to be different from the Object Id process property. If the Object Id process property is different from the active row, the primary business component is executed again to make the active row the same as the Object Id.
- It is possible to change the active row by assigning a new Row Id to the Object Id property. When Workflow detects that an assignment is made to the Object Id process property, Workflow executes the business component again and makes the new Row ID the active row.
- You can set the Object Id to an empty string then Workflow no longer enforces the must match rule. However, the parts of Workflow that require an Object Id, such as run-time events and Siebel Operation steps, cannot be used until the Object Id is set to a new Row ID.

Changing the Active Row in a Workflow Step

If you need to change the active row in a step of a workflow, you can do so by using an appropriate Business Service step or Siebel Operation step. However, you must promptly update the Object Id process property to the new active Row ID in the output arguments of the step that changes the active row.

Once a step finishes and output arguments have been evaluated, Workflow checks to make sure the Object Id matches the active row. So changes to the active row must be reflected in the Object Id property within the affected step.

About the In/Out Process Property

If necessary, you can run the workflow process and avoid receiving response data back. For example, to avoid inserts to the S_SRM_DATA table that can cause a heavy backlog.

Configuration within a workflow process determines whether response data is returned. If a process property's type is Out or In/Out, then outputs are returned to the caller. For example, to SRProc. If the caller receives a non null response from a callee, it writes the response into the S_SRM_DATA table. If no response data is received, the response is not written into the S_SRM_DATA table.

When a request is inserted into the S_SRM_REQUEST table, one or more rows are inserted into the S_SRM_DATA table for the component request parameters and input arguments used by the request. During submission of a request, the S_SRM_DATA table column MSG_TYPE_CD contains a value of REQ_DATA, indicating the type of data is to request data or inputs for the request. When the request finishes execution, a set of rows is also inserted into the S_SRM_DATA table with the MSG_TYPE_CD column having value of REQ_RESPONSE, indicating the request has returned a response back to the caller. In this case, since the request is in the S_SRM_REQUEST table, the caller is the Server Request Processor component.

To run a workflow while avoiding response data inserts

1 Configure one of the following options:

- Until immediately before the workflow ends, leave the In/Out property on the process property with the value In/Out.
- Right before the workflow ends, call another step that nulls out values stored in the process properties. In this way, SRProc receives no response text and an insert is not made into the S_SRM_DATA table.

2 Review the workflow's process properties and set the In/Out property to NONE.

Passing a Process Property In and Out of Workflow Steps

In long-running, interactive, and service workflow processes, when a hierarchical process property is passed to a step in a workflow process, the Type argument field of the top level process property is overwritten by Workflow for the duration of the call to match the name of the argument as specified by the input argument's configuration.

For example, assume MyTree is a process property with data type Hierarchy. MyBusSvc is a business service that has a hierarchical Input Argument named SomeTree. Consider the process property described in Table 16.

Table 16. Example of a Process Property Record in the MVPW

Input Argument	Type	Property Name
SomeTree	Process Property	MyTree

If the input argument is configured with values displayed in Table 16, then the call into MyBusSvc receives a child in its input process property with the Process Property Type field set to *SomeTree* , instead of *MyTree*:

The rest of the data in the child process property is the same as the contents of the process property MyTree.

Similarly, Workflow expects an output argument of a step in a workflow that passes out a hierarchy to send it out as a child of the output property set. Workflow locates the child by examining the Type argument field of the child.

The string in the Type argument field must match the Output Argument name as specified in the output argument applet of the step. Once Workflow finds such a child, the data is copied into the indicated destination process property. However, the Type argument field on the incoming hierarchy is discarded and instead the name of the process property overwrites the Type argument field.

To summarize, when passing hierarchies into and out of steps, the Type argument field of the top level of the data is used as the name of the argument.

NOTE: It is recommended the Type argument field of the top level of a hierarchical argument not contain data.

Passing a Property Set by Reference

Subprocess methods tasked with modifying large amounts of data must copy large quantities of data as input and output arguments. This condition can negatively impact the performance and scalability of these method invocations. If you can avoid unnecessary copying of data, you can achieve improvements in performance and scalability.

Pass By Reference is a feature that allows you to avoid passing large property sets by passing just a pointer to the property sets. You can use Pass By Reference for a workflow process by using a Sub Process step. For more information, see "About the Pass By Reference Feature for a Business Service" on page 94.

Passing a Process Property to an Error Workflow Process

You can pass more than just a system-defined process property to an error workflow process:

- To pass the original process instance to a user-defined process property to the error process, you must explicitly recreate those user-defined process properties in your error process, giving them the same name and data type.
- To pass a property set from the original process to the error process, you must create a common user-defined hierarchy process property in the original workflow process and in the error process, then use this common hierarchy property to pass the property set.
- To get the name of the original workflow process, you must create a common user-defined process property in the original workflow process and in the error process, then pass the original process name through this common user-defined process property.

To create a process property that passes data to an error workflow process

1. Perform the procedure described in "Defining an Error Exception" on page 159.
2. Click the Stop step, then define an input argument in the MVPW using values described in the following table:

Argument Field	Value
Name	%1
Type	Literal
Value	Error Message (or a valid string.)

3 In the Workflow Processes OBLE, define a new workflow process that references the error workflow process.

Concatenating a Process Property

You can use process property values in your expressions by concatenating workflow process properties with other process properties or with text.

The following example illustrates how four process properties can be used in a workflow to concatenate three string values. The three process properties have values in the Default String property of *Welcome*, *to*, and *Siebel*.

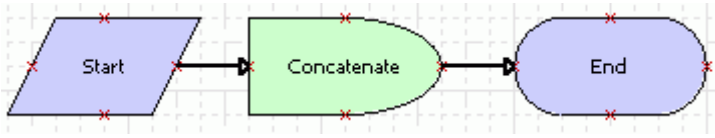
To create a workflow process that concatenates process properties

1 In Siebel Tools, in the Workflow Processes OBLE, create a new workflow process object definition with the following values:

Property	Value
Process Name	Concatenate
Business Object	Account
Workflow Mode	Interactive Flow

To view an example, see "Creating a New Workflow Process Object Definition" on page 316.

2 Open the Process Designer for the workflow process you created in Step 1, then create a workflow that resembles the workflow in the following diagram:



For more information, see "About Workflow Process Steps and Connectors" on page 87, and "Diagramming a Workflow Process" on page 61.

3 In the MVPW, create four process property records using values from the following table:

Name	In/Out	Business Object	Default String	Data Type
ProcessProperty1	In/Out	Account	Welcome	String
ProcessProperty2	In/Out	Account	to	String
ProcessProperty3	In/Out	Account	Siebel	String
ProcessProperty4	In/Out	Account	(no value)	String

For more information, see "About the Multi Value Property Window" on page 77.

4 Click the wait step then click the Output Arguments tab in the MVPW.

Note that the wait step is often used for testing and development. For more information, see "About the Wait Step" on page 108.

5 Define an Output Argument for the wait step using values from the following table:

Type	Output Arguments	Value
Expression	ProcessProperty4	[&ProcessProperty1]+' '+[&ProcessProperty2]+' '+[&ProcessProperty3]

The ampersand (&) identifies the text immediately following the ampersand as the name of a process property. The process property you indicate can also be the name of a business component field. ProcessProperty1, ProcessProperty2, and ProcessProperty3 can be of many different types, such as string and date. The Value property in this example must be set to String type.

6 Validate then Simulate the workflow process.

For more information, see "Process of Testing a Workflow Process" on page 231.

7 After control returns to Tools, right-click the Simulation window, then choose Watch Window.

8 In the Watch window, expand PS:Property Set, then note the values for the four process properties you defined in Step 3.

9 Click the Simulate Next button. In the Watch window, note that ProcessProperty4 now contains a concatenation of values from ProcessProperty1, ProcessProperty2, and ProcessProperty3.

Referencing a Process Property

A process property can be referenced from within an expression. The syntax is [&PropName]. For example:

```
[&Object Id] like '99-28B1T'
```

Where:

& indicates a process property, and *Object Id* is the name of the process property.

The following example illustrates how a process property can be used as a substitution variable within a more complex expression. This example uses an input argument to establish a message body:

```
The Activity #" + [&Object Id] + ", owned by " + [&First Name]+" "+[&Last Name] + "
has passed three days from the Due Date.
```

Where Object Id, First Name, and Last Name are defined as process properties.

For more examples of using a process property as a substitution variable, see "Example of Defining a Siebel Operation Search Specification" on page 99 and "Substitution Variables Commonly Used in Expressions" on page 120.

7 About Workflow Process Steps and Connectors

This chapter describes the various workflow process step types you use to build a workflow process. It includes the following topics:

About Workflow Process Steps and Connectors

This topic includes the following topics:

For an inventory of properties for each type of step and connector, see "Reference of Workflow Step and Connector Properties" on page 426.

About Workflow Steps

You can specify properties for a workflow process step. If working in the Process Designer, some of these properties can be specified in the Properties window. If working in the Object List Editor (OBLE), properties can be specified in the WF Steps OBLE.. For reference information about workflow step properties, see Table 113 on page 433.

Adding a Step to a Workflow Process

You use the Process Designer to add a step to a workflow process. Once a step is added, you can use the Properties window to define properties for the step.

In some cases, you can choose to use the WF Steps OBLE to define properties for a step. Once a workflow process is saved, a record for the new step appears in the WF Steps OBLE, where you can modify properties.

To add a step to a workflow process

1. In the Workflow Processes OBLE in Siebel Tools, choose the workflow process to which you need to add a step.

   If the status of the workflow process is *Completed* or *Not In Use*, click the Revise button in the WF/Task Editor toolbar.

   For more information, see "About the WF/Task Editor Toolbar" on page 72.

2. Right-click the workflow process record in the OBLE then choose Edit Workflow Process.

   The Workflow Designer Palette and Properties windows open along with the Process Designer. The workflow is editable. If the palette is not visible, you can display it by choosing View > Windows > Palette in the Tools application-level menu.

3. Drag then drop the step type you need to add from the Workflow Designer Palette to the canvas.

4. In the Properties window, enter or modify the Name property.

   When you step out of the property, the name updates on the step in the canvas. For more information, see "About Naming a Process Step or Process Property" on page 88.

5. (Optional) enter a description of the purpose of the step.

6. Define other properties for the workflow step, as necessary.

   For more information, see "Reference of Workflow Step and Connector Properties" on page 426.

About Naming a Process Step or Process Property

A name given to a workflow process step or a process property must be unique within a workflow process. When you create a new process step, the step's Name property is automatically populated with a name/number combination that you can change. If you change the name, the new name, including the number, must be unique from the names of the other steps in the process.

The name given automatically to a step or a connector is based on the step or connector's type. For example *Business Service 0* for a business service step, or *Siebel Operation 0* for a Siebel Operation step. The number given automatically in the name for a step or a connector is an integer, such as 0, 1, 2, 3, and so forth. This number differentiates instances of the same type of step or connector. For example, Business Service 0, and Business Service 1. This number, called *sequence*, is stored as part of the name.

Note that some symbols, such as the period (.), cannot be used in a process property name.

About Sequence Numbers in Workflow Steps

When a new step of the same type is added to the workflow, if there is no gap between the sequence numbers for the type, the next number in the sequence is used. Otherwise, the first sequence number in the first gap for the step type is the sequence assigned to the new step. Note that a gap can be created when a step is deleted.

For example, assume are five business service steps in a workflow process: Business Service 0, Business Service 1, Business Service 2, Business Service 3, Business Service 4. If Business Service 1 and Business Service 2 are deleted, the next business service step assumes the sequence number 1, and it is automatically named Business Service 1. This sequencing approach is the same for connectors.

About the Wait Step Processing Mode Property

There are performance considerations when choosing between synchronous and asynchronous for the Processing Mode property on the wait step. For more information, see "About the Server Requests Business Service" on page 374.

SetFieldValue and the Processing Mode

It is recommended that when SetFieldValue is used to invoke a workflow, that the Processing Mode property be set to Local Synchronous. Since SetFieldValue can occur without data being committed, setting the Processing Mode to Remote Synchronous or Remote Asynchronous when using SetFieldValue can result in Workflow running out of process, and Siebel Workflow is not able to access the process's uncommitted data.

About Workflow Connector Properties

You can specify properties for a workflow process connector in the Properties window from within the Process Designer. For reference information about connector properties, see Table 113 on page 433.

For information about defining a run-time event on a connector, see "Invoking a Workflow Process from a Run-Time Event" on page 138.

About Workflow Process Step Types

There are several different types of workflow process steps you can use when developing a workflow process. This topic provides details for each of the step types. It includes the following topics:

Overview of Step Types

This topic provides an overview of the various step types. Figure 14 displays the Workflow Designer Palette.

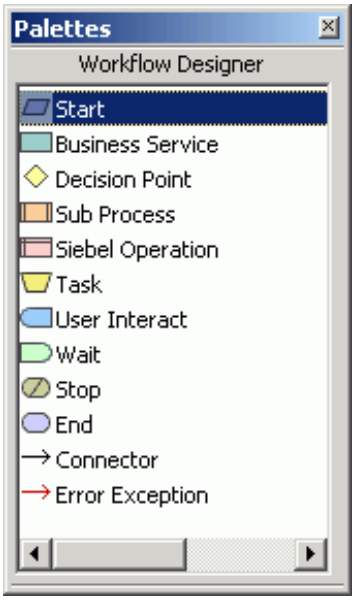


Figure 14. Step Types and Connectors in the Workflow Designer Palette

Table 17 describes the step types and connectors available in the Workflow Designer Palette.

Table 17. Description of Step Types and Connectors in the Workflow Designer Palette

Step Type	Description
Start	Defines the input conditions that must be met to execute an instance of a business process. Note that conditions are actually defined on the connector emanating from the Start step, not the start step itself.
Business Service	Calls a business service that allows you to execute predefined or custom actions in a workflow process.
Decision Point	Evaluates a condition to determine the next step to execute.
Sub Process	Calls a workflow process. The called workflow process can be a standalone workflow process, having its own process definition and flow of workflow steps.
Siebel Operation	Performs actions on a business component, such as Insert, Update, or Query.
Task	Calls a Task UI object definition.
User Interact	Controls the flow of Siebel views within an application. Guides the user through a specified flow of Siebel views based on the user's actions, or executes a specified set of actions.
Wait	Suspends execution of the workflow process for a specific period of time or until a specific event occurs.
Stop	Terminates the workflow process instance. Presents an error message to the end-user.
End	Indicates when process execution is finished.
Connector	Determines direction of flow between steps in a workflow process.
Error Exception	Traps for a deviation from normal processing, such as a system error or a user-defined error.

About the Start Step

A start step indicates the starting point for the workflow. A workflow process must have one, and only one, start step.

Conditional logic and run-time events can be defined on the connector emanating from the start step, but not the start step itself. Purposes that the connector emanating from the start step can fulfill include:

- To establish the conditional logic that describes input conditions that must be met for a process to execute. For example, to handle open service requests, you can define a condition of Status equals Open on the connector. For more information, see "About Conditional Logic on a Branch Connector" on page 112.

- To define a run-time event that is used to invoke the workflow process. For example, to generate an activity when an opportunity's revenue is greater than $10,000, you can define a WriteRecord run-time event, then configure a workflow process that inserts the activity record when an opportunity record is saved that meets the condition. For more information, see "Invoking a Workflow Process from a Run-Time Event" on page 138. To view an example that uses a run-time event, see "Example Workflow Process That Creates an Activity for a Sales Representative" on page 315.

Defining a Start Step

You define a Start step in the Process Designer in Siebel Tools.

To define a Start step

Perform the procedure described in "Adding a Step to a Workflow Process" on page 87 with the following modifications:

- If a run-time event is used to invoke the workflow, attach a connector to the Start step, then define the run-time event on that connector.

About the Business Service Step

A business service allows you to execute a predefined or custom action in a workflow process. Some examples of predefined business services include:

- **Notifications.** Notifications can be sent to employees or contacts using the Outbound Communication Server business service.
- **Assignment.** Assignment Manager can assign an object in a workflow process by calling the Synchronous Assignment Manager Request business service.
- **Server tasks.** You can run a server component task using either the Asynchronous Server Requests or the Synchronous Server Requests business service.

For a list of some of the more commonly used predefined business services, see "Predefined Business Services" on page 373.

You can also define your own custom business services using Siebel Tools or the Administration-Business Service view in the Siebel client. You can use Siebel VB or Siebel eScript to define your own custom business services that you can invoke from a workflow process.

CAUTION: A Business Services invoked from a workflow process cannot include browser scripts. A Business Service only works with server scripts. A business service with browser scripts fails if it is executed from a workflow process on the Siebel Server.

Defining a Business Service Step

You define a Business Service step in the Process Designer in Siebel Tools.

To define a Business Service step

Perform the procedure described in "Adding a Step to a Workflow Process" on page 87 with the following modifications:

1. In the Business Service Name property, pick the name of the business service to be invoked from the picklist.

   The picklist contains the business services defined in Siebel Tools or the Siebel client.

   For information about creating customer defined services, see *Integration Platform Technologies: Siebel Enterprise Application Integration*.

2. In the Business Service Method property, choose the method for invoking the service. The choices available for this property depend on the business service you defined in Step 1.

3. With the business service step chosen, use the MVPW to create new input and output arguments.

   For more information, see "About Process Properties" on page 74.

Making a Business Service Visible to a Workflow Process

Business service objects, which include the business service, business service method, and business service method argument, have Display Name and Hidden properties in Siebel Tools. For these objects to be displayed in a picklist, the Hidden flag for the object must be set to FALSE. For example, the methods and arguments you define in your business service appear in the picklists in the Arguments list applets for the business service. Note that, by default, a business service defined in the Siebel client is not hidden.

When creating a workflow definition in Siebel Tools, the value specified for the Name property for business services objects is the value that also appears in the picklists.

To make a Business Service visible to a workflow process

1. In Siebel Tools, from the Object Explorer applet, click the Business Service object type.

   This action displays a list of predefined business services.

2. In the Business Services OBLE, click the business service you need to modify.

3. In the Properties window, change the Hidden property to FALSE.

4. In the Object Explorer, expand the Business Service object type, then click the Business Service Method object type.

5. In the Business Service Methods OBLE, click the method you need to modify, then change the Hidden property to FALSE in the properties window.

6. Repeat Step 5 for each method, if applicable.

7. In the Object Explorer, expand the Business Service Method object type, then click the Business Service Method Arg object type.

8. In the Business Service Method Arguments OBLE, click the argument you need to modify.

9. Change the Hidden property to FALSE in the Properties window.

10. Repeat Step 9 for each method argument, if applicable.

About the Pass By Reference Feature for a Business Service

The _SupportsPassByRef_ user property is available for use on a Siebel business service. You cannot add this user property to a custom business service.

If, in a workflow process, you configure a Business Service step that is not marked for Pass By Reference, the Workflow engine returns an error. For the Workflow engine to honor Pass By Reference, the business service must have the Business Service User Prop setting to TRUE and the Business Service step in the workflow must be marked for Pass By Reference.

CAUTION: Use of the Pass By Reference feature is not supported for a custom business service.

For more information, see "Defining a Sub Process to Support Pass By Reference" on page 96.

Business Service Calls to a Unix Shell Script

A Siebel process running on UNIX runs as the user that launches Siebel Services. If a workflow process invokes a business service that calls a UNIX shell script, the shell script runs as the Siebel Service Owner account.

About the Decision Point

A Decision Point is a type of workflow process step that evaluates one or more defined conditions to determine the next step of a process instance.

A Decision Point evaluates a business component's user data in a record when the workflow process is executed. For example, assume the workflow process is triggered by a workflow policy and multiple violations of a policy condition occur during the Workflow Monitor Agent's action interval. In this case, when the workflow process is executed the Decision Point determines which branch to take based on the current value of the business component field.

Defining Branching Logic in a Workflow Policy

If the branch connector logic is moved from the workflow process to a workflow policy level, then the policy generates unique events within the defined action interval. In this way, the workflow process is triggered by a workflow policy violation. For more information, see "Building Expressions with Expression Builder" on page 116.

Defining a Decision Point

You define a Decision Point in the Process Designer in Siebel Tools.

To define a Decision Point

Perform the procedure described in "Adding a Step to a Workflow Process" on page 87 with the following modifications:

- Continue with "Defining Conditional Logic on a Branch Connector" on page 114 to create conditional logic for the Decision Point.

About the Sub Process Step

This topic describes the main parts for creating a Sub Process step. A Sub Process step allows you to invoke a separate process from within a process. A workflow process can have one or more Sub Process steps.

You can also define a Sub Process step to support Pass By Reference. For more information, see "Passing a Process Property In and Out of Workflow Steps" on page 81 and "Defining a Sub Process to Support Pass By Reference" on page 96.

When using the Process Simulator, you must publish and activate a subprocess called by the workflow process you are simulating prior to invoking the simulator. For more information, see "Using the Simulator With a Workflow Subprocess" on page 227.

For a description of Sub Process step properties, see Table 106 on page 429.

Defining a Sub Process Step

Before you define a Sub Process step, you must define the workflow process that calls the step.

Once a Sub Process step is created within a workflow, to directly edit the corresponding subprocess definition, double-click the Sub Process step to reach the design canvas for that subprocess. You can also right-click the Sub Process step, then choose Edit Sub Process.

To define a Sub Process step

Perform the procedure described in "Adding a Step to a Workflow Process" on page 87 with the following modifications:

1. Use the Subprocess Name property in the Properties window to choose the name of the workflow process the Sub Process step calls.
2. In the MVPW, create new input and output arguments.

   For more information, see "About Process Properties" on page 74 and "Defining an Input Argument for a Sub Process Step" on page 95.
3. Create recipient arguments in the MVPW.

   For more information, see "Argument Fields for a Recipient Argument" on page 442.

Defining an Input Argument for a Sub Process Step

An input argument allows you to populate process properties in the Sub Process step. For example, the Object Id is passed from the main workflow process to the sub-workflow process through an input argument. If the subprocess is based on a different business object, you must pass the relevant row ID of the target object as the subprocess Object Id Process Property.

NOTE: The Object Id passed to a subprocess must not contain the Object Id of the parent process if the subprocess creates a child object. When a subprocess creates a child object, it is recommended the Object Id passed to a subprocess be null.

Copying Values from a Parent Process to a Subprocess

You can copy a value to a record being updated or inserted in a subprocess by creating a process property in the subprocess and assigning the value to it through the Sub Process Input Arguments applet in the Sub Process step. The process property can then be used in the subprocess's Siebel Operation step.

In the example below, you copy an order's Order Name into the Name field of a newly created Opportunity.

To copy a value to a record being updated or inserted in a subprocess

1. Add a process property to the subprocess called Opportunity Name.
2. In the Sub Process step, map Order Parent Product Name to Opportunity Name.
3. Assign Opportunity Name to the Name field of the Opportunity record created within the Siebel Operation step.

Defining a Sub Process to Support Pass By Reference

In some instances a subprocess can modify large amounts of data, requiring the subprocess to copy large quantities of data that negatively impacts performance and scalability. In such cases, you can use the Pass By Reference feature so that a pointer to the data is passed, but not the data itself.

The example displayed in Figure 15 displays an example workflow process that passes a large property set created by the Create Siebel Message step to the 11i Process Order step.

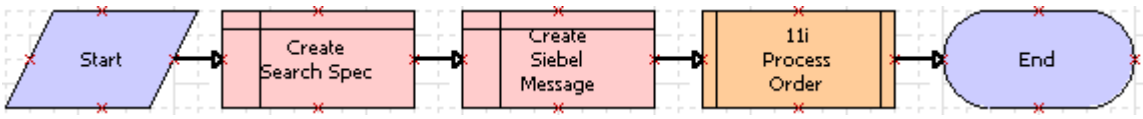


Figure 15. Example Workflow for Which Pass By Reference with a Sub Process Step is Useful

When you configure a Sub Process step to support Pass By Reference, it is not necessary to map the output argument in the Sub Process step for the hierarchical properties that you are passing. The Sub Process step overwrites the passed input hierarchical argument. Optionally, you can have the Sub Process step modify the passed input hierarchical argument.

To define a Sub Process step to support pass by reference

1. In the Workflow Processes list, choose the workflow process that is your subprocess.
2. In the Properties window, set the Workflow Process property Pass By Ref Hierarchy Argument to TRUE.

   In the Worfklow Process list, the Pass by Ref Hierarchy property is now checked for the child workflow.

About the Siebel Operation Step

The Siebel Operation step includes operations such as Insert, Update, or Query. These operations are performed on business components. This topic includes the following topics:

Defining a Siebel Operation Step

You can define a Siebel Operation step for a business component associated with the business object defined for the workflow process. If you need to update a business component not associated with the business object, you can invoke a subprocess or associate the business component to the business object using Siebel Tools.

To define a Siebel Operation step

Perform the procedure described in "Adding a Step to a Workflow Process" on page 87 with the following modifications:

1. In the Operation property, choose the type of operation.

   For updates or inserts of fields that have dependencies, make sure the fields are valid. For example, if you have a service request process and your workflow process is updating the area and sub-area fields, you must make sure the values chosen for the subarea field are valid for that associated area.

2. For the Business Component property, choose the name of the business component whose records this business service is interacting.

3. (Optional). In the MVPW, create new input and output arguments.

   If an Insert or Upsert operation is performed, make sure you add the required arguments in the MVPW. Make sure you specify the name of the field to be updated in the Field Name argument field. In the Type argument field, choose an input argument type then define other argument fields, depending on the type you specify.

   For more information, see "About Process Properties" on page 74.

4 (Optional). Create a search specification in the MVPW.

a In the Type argument field, choose a search specification type, as described in the following table:

Type	Description
Literal	A a static string where the run-time value is the same as the one defined in the Process Designer.
Expression	A Siebel expression, often based on other run-time variables, where the run-time value can only be determined at run time by evaluating the expression.

b In the Search Specification argument field, enter search specifications.

c If the search specification Type is Expression, choose the applicable business component name.

For more information, see “Defining a Siebel Operation Search Specification” on page 98.

Defining a Siebel Operation Search Specification

You can define search specifications to identify the specific data on which to perform the operation. Search specifications are used when the business component has multiple records and you need to perform the operation on only some of the records. For example, if you have a process for the Account object and you need to update only those Opportunities with a lead quality of Poor, you define search specifications to access only those Opportunities.

CAUTION: Define your Siebel operation search specification as efficiently as possible, so that only the smallest necessary set of rows match. Search specifications that identify a large set of rows can cause severe performance degradation.

Defining a Literal Search Specification

A search specification of type Literal is executed as written. For example, [Status] LIKE '*Open*'. A search specification of type Expression allows you to construct a search specification dynamically. For example, "[Contact ID] = ' " + [&New ID] + " ' " is evaluated to [Contact ID] = '1-ABC' if the New ID process property is 1-ABC at run time.

Referencing a Business Component Field in the Search Specification

Both sides of an expression can reference a business component field. The Filter Business Component defines the business component referenced on the left side, and the Expression Business Component defines the business component referenced on the right side. For example, consider the configuration described in Table 18.

Table 18. Example of Arguments Used in a Search Specification

Argument Field	Value
Filter Business Component	Account
Expression Business Component	Contact
Expression	"[Id] = [Account Id]"

In this example, the expression evaluates as described in the following syntax:

```
"[Account.Id] = [Contact.Account Id]"
```

Example of Defining a Siebel Operation Search Specification

An expression for a search specification can be defined with compound expressions and substitutions from process properties or fields.

For example, consider the following generic syntax:

```
"([Field1] > '" + [&Process Property Name] + "') OR ([Field2] IS NULL) OR ([Field3]
IS NULL)"
```

This syntax can be translated to:

```
([Field1] > 'value') OR ([Field2] IS NULL) OR ([Field3] IS NULL)
```

Expressions that now include literal representations of the business component fields, where 'dFromDate' is a custom process property, include:

- `"([Open] > '" + [&dFromDate] + "') OR ([Open] IS NULL)"`
- `"([Open] > '" + [&dFromDate] + "') AND ([Status] IS NULL)"`

To view search specification usage in an example workflow, see “Defining the Workflow Process” on page 326.

For more information about using a process property as a substitution variable, see “Referencing a Process Property” on page 85.

Updating a Field That Resides in a non-Primary Business Component

You can update a business component field that resides in a non-primary business component. To update a field in a non-primary business component, one of the following conditions must exist:

- A join must exist between the base table for the primary business component and the field you need to update.
- A link must exist between the primary business component and the business component that contains the field you need to update.

For example, assume you need to update sales stage data in a workflow whose business object property is set to Opportunity. The predefined Sales Stage join on the opportunity business component provides the capability to update the sales stage field.

Considerations to weigh when using joins and links include:

- If a predefined join or link does not exist, you can create one.
- In some cases, to update a field in a non-primary business component, you can choose to copy an existing business object then specify it in the business object property for a workflow process. If you choose a different business component as the primary business component for this business object, then you must make sure link or join information specified for that business component is removed or corrected. Otherwise, an error can result.

For more information, see "Defining a Primary Business Component for a Business Object" on page 105.
For more information about modifying joins and links, see *Configuring Siebel Business Applications*.

Updating a Field That is Based on a Multi-Value Group

If you need to update a business component field that is based on a multi-value group, you can define a business component for the field then link the business component to the object using Siebel Tools.

For example, assume you need to update the Account Team business component field. Account Team is based on a multi-value group, so it cannot be updated by choosing the Account business component. However, you can create a business component called *Account Team*, then associate it with the Account business object using Siebel Tools. You can then choose Account Team as the business component to update with the Siebel Operation step.

For more information about multi-value groups, see *Configuring Siebel Business Applications*.

Updating a Calculated Field

A Siebel Operation step cannot be used to update a calculated field because typically the calculated field requires values from other business component fields. Instead, use an expression to perform calculations.

Using the Siebel Operation Step to Traverse a Record Set

The NextRecord, PrevRecord, and QueryBiDirectional operations on the Siebel Operation step can be used in conjunction with the Update, Query, and Insert operations to traverse a record set.

Using NextRecord and PrevRecord, you can navigate through the records of a child business component of the business object associated with the workflow:

- The NextRecord operation changes the active row of the target business component to the next record in the current workset.
- The PrevRecord operation changes the active row to the previous record.

You must perform a query on the target business component before using subsequent Next/Previous Record operations. If the workflow is traversing active records or if your workflow uses the PreviousRecord operation, query bidirectionally. Otherwise, use the standard Siebel query operation.

NextRecord, PrevRecord, and QueryBiDirectional cannot be used to navigate through records of the primary business component. These operations are designed to navigate records of a child business component in a workflow process. Therefore you must use a child business component.

To see a detailed example that uses the traversing a record set technique, see "Example Workflow Process That Traverses a Record Set to Close Obsolete Service Requests" on page 324.

Output Arguments Used With Traversing a Record Set

When you use Next Record and Previous Record to traverse records of a child business component, an output argument called NoMoreRecords in the Siebel Operation is set to TRUE when the Siebel Operation attempts to read the next record but finds there are no more records in the record set to traverse. NoMoreRecords is set to TRUE in the forward direction for Next Record or in the backward direction for Previous Record.

NoMoreRecords can be assigned to a process property, and in conjunction with a Decision Point, can be used to exit the loop after navigating through every record in the record set. To support this output argument, the Process Designer displays an Output Argument column in the output arguments tab for the Siebel Operation step.

Another output argument, NumAffRows (number of affected rows) exists for Query, Update and Upsert operations:

- When used with Query, NumAffRows provides the number of rows returned as a result of the query.
- When used with Update and Upsert, NumAffRows provides the number of rows updated.

Using the Siebel Operation to Update a Process Property

You can create a test workflow process that updates a process property. This technique can be useful during development. For example, you can take the sum of two business component integer fields that provide input to a decision downstream in the workflow process. By updating a process property in the Siebel operation step with the sum of the two required fields, you can use this property on a subsequent branch connector.

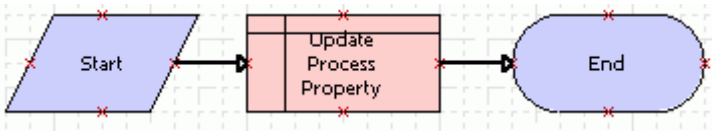
To create a test workflow that updates a process property

1 In Siebel Tools, create a new workflow process object definition using values described in the following table:

Property	Value
Process Name	Updating a Process Property
Business Object	Action
Workflow Mode	Service Flow

To view an example, see "Creating a New Workflow Process Object Definition" on page 316.

2 Add steps and connectors until your workflow process resembles the workflow illustrated in the following diagram:



For more information, see "About Workflow Process Steps and Connectors" on page 87, and "Diagramming a Workflow Process" on page 61.

3 Click anywhere on the canvas background.

4 In the MVPW, right-click then add a new process property using values described in the following table:

Argument Field	Value
Name	Custom Process Property
Display Name	Custom Process Property
In/Out	In/Out
Business Object	Action

For more information, see "About Process Properties" on page 74.

5 Click the Update Process Property step, then click the Search Spec Input Arguments tab in the MVPW.

6 Right-click, choose New Record, then define a new record using values described in the following table:

Argument Field	Value
Expression Business Component	List of Values
Filter Business Component	Query

Argument Field	Value
Search Specification	[Name] = 'Personal' AND [Type] = 'TODO_TYPE'
Type	Literal

7 With the Update Process Property step still chosen, click the Output Arguments tab in the MVPW.

8 Right-click, choose New Record, then define a new record using values described in the following table:

Argument Field	Value
Property Name	Custom Process Property
Type	Expression
Value	[Class Code]
Business Component Name	List Of Values

9 Validate then Simulate the workflow process.

For more information, see "Process of Testing a Workflow Process" on page 231.

10 Implement this technique in your production workflow.

About the Siebel Operation Object Id Process Property

After executing an Insert or Upsert operation, the Siebel Operation Object Id process property automatically stores the row ID of the record that was created. The Object Id for the process is automatically passed to Siebel Operation steps. Because this automatic passing occurs, it is not necessary to define a search specification unless you are updating child records. For example, if you have a workflow process based on the service request business object and you need to update the service request, it is not necessary for you to enter a search specification. However, if you need to update activities for the service request, you can enter a search specification to query the specific activity you need to update. Otherwise, the update step updates every activity associated with the service request.

The Object Id cannot be null if you are executing a Siebel operation, unless you are inserting into the primary Object Id. If the process has no Object Id, the Siebel Operation step returns an error.

Values returned by the Siebel Operation Object Id process property when performing a query operation for child records include:

- The row ID if one record matches.
- Null/no value if no records match.
- An asterisk (*) if multiple records match. Provided to distinguish from values that return a unique record or no record. Typically, either a unique record matches or no records match. In most cases, multiple records will not match and an asterisk will not be returned.

The *only* option returns a row ID of a matching row.

The Insert, Update and Upsert operations update the Siebel Operation Object Id process property of the record's row ID.

About the Upsert Operation on the Siebel Operation Step

The Upsert Operation provides a way to perform either an insert or update operation based on whether a record or records exist in the database. Hence the name, *Upsert*. The operation can perform *up*date or in*sert* operations.

For example, assume you have a search specification on a Siebel Operation step that queries the database. Table 19 describes how Upsert works.

Table 19. Description of What the Upsert Operation Does

If a record exists in the database. . .	If a record does not exist in the database. . .
Upsert performs similar to an Update operation, and updates each record with modified values as determined by the workflow process.	Upsert performs similar to an Insert operation and inserts a new record.

Examples Where Upsert is Useful

In the Siebel client, assume the user is taken to a view where the user clicks a checkbox to create a new contact then enters the relevant contact information. Assume the Yes default for the checkbox creates a new contact record. If the user creates a new contact, navigates away from the view, then returns to the view, when the user returns there is the potential that another contact is created. In this example, an evaluation must be made whether the contact already exists or not. If a record does exist, then the contact record must be updated. If the record does not exist, then a new contact record must be created.

In another example, assume a workflow runs in the background at midnight, processing orders submitted from an external system. If a given order does not exist, Upsert creates a new record. If a given order already exists but must be updated, Upsert updates the record.

About the Relationship Between a Siebel Operation and Siebel Object Layers

Workflow Policy programs and Siebel Operation steps use different object layers to update data. For example, you can have a workflow policy that calls a Workflow Policy program to update a Service Request record. This technique goes through the data layer, where the state model does not apply.

Conversely, if you have a workflow policy that calls a workflow process action and in the workflow process you have defined a Siebel Operation step to update a Service Request record, this technique goes through the object layer, where the state model does apply.

Defining a Primary Business Component for a Business Object

The business object defined for a workflow process must have a primary business component defined in Siebel Tools.

To designate a primary business component for a Business Object

1. In Siebel Tools, in the Object Explorer choose the Business Object object type.
2. In the Business Objects OBLE, choose the business object to be modified.
3. In the Properties window, use the picklist in the Primary Business Component property to pick the appropriate component name.

   Configure a primary business component by choosing the key business component for the specific business object.
4. Compile the SRF.

Once a primary business component is defined, the business object appears in the Workflow Processes list editor.

About the Task Step

A Task step is similar to a Sub Process step in that the task step represents a container for a further set of steps below the level of the steps in the workflow in which the task step is defined. Task steps launch tasks from within a workflow process.

Once a workflow calls a task, the task is visible in the user's Inbox. Until the user finishes the task, the workflow is in a waiting state. Once the task is finished, the workflow moves to the next workflow step, continuing the workflow's execution.

The main parts of creating a Task step for a workflow process include:

- Defining the Task step's input arguments, including associating the Task step with a task flow.
- Defining the Task step's output arguments.
- Assigning a recipient for the task.

Once you have created a Task step in the Process Designer and you have associated the Task step with a task flow, you can open the Task Editor by double-clicking the task step on the Process Designer canvas.

Considerations to weigh when using a task step include:

- To create the definition for a Task step in a workflow process, you must already have created the relevant task flow in the Task Editor. For information on creating tasks, see *Siebel Business Process Framework: Task UI Guide*.
- The Mode property for the workflow process must be set to Long Running Flow if the workflow launches a Siebel Task UI.

Defining a Task Step

You define a Task step in the Process Designer in Siebel Tools.

To define a Task step

Perform the procedure described in "Adding a Step to a Workflow Process" on page 87 with the following modifications:

1. In the Task Name property, use the picklist to choose a task flow to associate with the Task step added to the workflow.
2. In the Properties window, make sure the Inactive property is set to FALSE.
3. In the MVPW, create new input, output, and recipient arguments.

   For more information, see "About Process Properties" on page 74, and *Siebel Business Process Framework: Task UI Guide*.

About the User Interact Step

The User Interact step provides you a way to configure the flow of Siebel views within an application. Siebel Workflow guides users through a specified flow of Siebel views based on the user's actions, or executes a specified set of actions. This flow can be modified as business conditions change.

The User Interact step can take process properties as input arguments. In this way, you can dynamically set view names as you design your interactive workflow process. The view name property is set in the View property, an unbounded picklist, of the User Interact step.

User Interact Step Behaviors

User Interact step behaviors include:

- Sends a request to the Siebel Web Engine to build the view, then brings up the required view in the user session.

  Only one view can be built at a time. You cannot combine a User Interact step with another action, such as bringing up a message box or building another view simultaneously. You also cannot combine a UserInteract step with the invocation of a task.

- Waits in the memory of the user session for a run-time event to resume processing. In cases where there is no run-time event defined, the workflow process continues to the end.
- If, after a User Interact step, the user manually navigates out of the view, the workflow process remains in the memory of the user session. The process is deleted when the user session is terminated or when another workflow process is instantiated in the same user session.
- Automatically queries the primary business component of the workflow's business object if it is not already in the executed state. The query searches for a record for which the ID matches the value of the Object Id property.

- Is resumed after a User Interact step only if the ID of the record on which the run-time event registered in the conditions defined on the connector match the value of the Object Id process property.

User Interact Step Restrictions

Restrictions when configuring a User Interact step include:

- A workflow process running in the Workflow Process Manager server component must not contain User Interact steps. That is, if the workflow is running in background mode or in batch mode, it cannot include User Interact steps. In this situation, if the Workflow Process Manager encounters a User Interact step, an error results.
- The User Interact step is only supported if the process is invoked through a script or run-time event and the process is run locally in the application object manager.
- It is not possible to use an interact step in a workflow process to move to a view and invoke the Search Center.

Defining a User Interact Step

You define a User Interact step in the Process Designer in Siebel Tools.

To define a User Interact step

Perform the procedure described in "Adding a Step to a Workflow Process" on page 87 with the following modifications:

1. Use the picklist in the User Interact View property to specify the view name to which you must navigate the user.
2. In the Description property, enter a description of the purpose of the User Interact step.
3. If the User Interact step includes conditional logic, see "Defining Conditional Logic for a User Interact Step" on page 107.

Defining Conditional Logic for a User Interact Step

Conditional logic can be defined on a User Interact step. This involves establishing conditional logic on one or more connectors emanating from the User Interact step. For more information, see "About Conditional Logic on a Branch Connector" on page 112.

Defining a Run-Time Event With a User Interact Step

Branches emanating from a user interact step in an interactive flow must be associated with a run-time event.

If the mode property for the workflow process is set to Interactive Flow, and the workflow has a User Interact step that does not have a run-time event defined on the outgoing branch, it results in an error during validation. To avoid this error, the Workflow Mode property for the workflow process should be changed to 7.0 Flow if the run-time event is not required or not used in the user interact step. For more information, see "About the Workflow Mode Property" on page 121.

Using a User Interact Step to Create Substitute View Names

You can associate view names with process properties so that they can be set dynamically at run time. You do this by assigning a view name to the run-time value of a process property.

To associate a view name with a process property to set view names dynamically at run time

In the User Interact View property of a User Interact step, type the following string: `[&ProcessPropertyName]`.

The Workflow engine recognizes this string and assigns the view name at run time.

About the Wait Step

The Wait step allows you to suspend process execution for a specific period of time or until a specific event occurs. Workflow administrators can specify to pause a process instance in units of seconds, minutes, hours, or days. In addition, administrators can specify a service calendar to account for business hours and days when waiting a specified duration.

If a workflow process includes a Wait step, by default it is persisted.

Note that the Wait step is often used for testing and development purposes. Unlike the Siebel Operation step, the Wait step can be used without impacting business component metadata or user data. To view an example that uses the wait step for testing, see "Using a Run-Time Event Within a M:1 Relationship" on page 152.

Defining a Wait Step

You can define a Wait step to pause a process instance.

To define a Wait step

Perform the procedure described in "Adding a Step to a Workflow Process" on page 87 with the following modifications:

1. Make sure the Workflow Mode property for the workflow process is set to Interactive Flow.

   A wait step can only be used in an Interactive Flow. For more information, see "About the Workflow Mode Property" on page 121.

2. With the Wait step chosen in the canvas, define input and output arguments for the step in the MVPW.

   For more information, see "About Process Properties" on page 74.

   When defining a duration argument that is greater than 60 seconds, specify minutes or a larger unit of measure so that business component data is refreshed. Workflow is resumed from Workflow Process Manager when units of minutes or higher are specified. Wait steps with durations measured in anything other than seconds are automatically persisted.

About SleepMode and the 7.0 Flow Workflow Process

When a workflow process has the property Workflow Mode set to 7.0 Flow, the input argument SleepMode allows you to specify how the Wait step resumes the process:

- When set to the default Local value, the process is resumed within the session that launched the process.
- When set to Remote, the process is resumed on the server, by the Workflow Process Manager server component. A component request for the WfProcMgr server component is created to be run at the time determined by the Wait step's input argument called Duration.

Note that when a workflow process has the property Workflow Mode set to anything other than 7.0 Flow, the input argument SleepMode is ignored and the mode is always Remote. For more information, see "About the Workflow Mode Property" on page 121.

About the Stop Step

The Stop step is used to raise an error to the user and terminate the workflow process instance. The main parts of creating a Stop step for a workflow process are:

1. Defining a Stop Step on page 109
2. Defining a Custom Error Message with the Stop Step on page 110

NOTE: When using a Stop step in a subprocess within a workflow process, the Stop step stops the subprocess and it also stops the parent, invoking process. It is not necessary for you to configure a Stop step within the parent process to stop subprocess logic from executing.

Table 20 describes the way the Stop step is handled, depending on how it is called and in which object manager it is running.

Table 20. Description of How Workflow Process Manager Handles a Stop Step

Stop Step Condition	Where Process Runs	Action Taken by Workflow Process Manager
Workflow policy calls a process that contains a Stop step.	(This cell is intentionally empty.)	Exits. Writes an error message to the log file.
A script or run-time event calls a process that contains a Stop step	Workflow Process Manager Object Manager	Writes an error message to the log file.
	Application Object Manager	Flags an error message to the user.

Defining a Stop Step

This topic describes how to define a stop step.

To define a Stop step

Perform the procedure described in "Adding a Step to a Workflow Process" on page 87 with the following modifications:

1. In the Error Code property, choose a predefined error code from the picklist.

   For information about defining a custom error message, see "Defining a Custom Error Message with the Stop Step" on page 110.

2. In the Error Message property, enter an error message.

3. With the Wait step chosen in the canvas, define input arguments for the step in the MVPW.

   For more information, see "About Process Properties" on page 74.

   Note that no picklist is available for the Name argument field. The input arguments for a Stop step are the substitution variables that appear in the error message. Substitution variables are identified by a percentage symbol: %. To define the substitution value, enter the substitution variable in the input argument's Name argument field, such as *%1*.

Invoking the Stop Step

It is recommended that the Stop step be used only in a workflow process invoked from a script.

For example, consider a workflow process that displays a custom error message in a stop step. When the workflow is run, the custom error message is displayed that includes stack information that you need to suppress. It is not possible to suppress stack information with a stop step.

However, a workflow process invoked from a script can have an end step that defines the error message in a process property. When the workflow encounters the required condition, the process property that contains the error message is sent. Since the subsequent step is an end step that does not display any messages, control is returned to the calling script that checks for the value set in the process property, then uses RaiseErrorText() to display the message. The error dialog displays the error text but does not display workflow or stack trace information.

Defining a Custom Error Message with the Stop Step

If none of the predefined error messages provided in the Error Code property on the Stop Step meet your requirements, you can define a custom error message on the Stop step.

To define a custom error message on the Stop step

1. Add a Stop step to your workflow process.

2. Click the Stop step, then use the Properties window to set the Error Code property to WF_ERR_CUSTOM_1.

   Note that the Error Message property defaults to %1.

3. Create a new input argument in the MVPW for the Stop step.

   For more information, see "About Arguments for a Workflow Process Step" on page 76.

4 Set the Name argument field to the value displayed in the Error Message property in the Properties window. In this case, that value is %1.

5 Enter the custom text message in the Value argument field.

When you create the input argument in this way, the value you specify is used in the %1 variable.

Defining Multiple Custom Error Messages

Several customizable codes are available in the Error Code property on the stop step. These are indicated by WF_ERR_CUSTOM_x. Since each WF_ERR_CUSTOM_x is unique, it can only be used once. If you need to display multiple custom error messages, use WF_ERR_CUSTOM_2, WF_ERR_CUSTOM_3, and so forth, instead of using %1, %2 for the same WF_ERR_CUSTOM_x.

About the End Step

An End step specifies when a process instance is finished. It also provides one last chance to store output arguments to a process property. Each workflow process definition must have at least one End step.

For more information, see "Defining an End Step" on page 111.

Differences Between the End Step and the Stop Step

An important difference between the Stop step and the End step is that the Stop step sets the workflow state to In Error while the End step sets the workflow state to Completed. This is important to keep in mind when calling a workflow process using Workflow Monitor Agent.

If the WorkMon parameter Ignore Errors is set to False, a workflow process that encounters a Stop step causes WorkMon to exit with error. If the workflow process encounters an End step, WorkMon does not exit with error.

Defining an End Step

You define an End step in the Process Designer in Siebel Tools.

To define an End step

Perform the procedure described in "Adding a Step to a Workflow Process" on page 87 with the following modifications:

- With the end step chosen in the canvas, define output arguments for the step in the MVPW.

  For more information, see "About Process Properties" on page 74.

About Conditional Logic on a Branch Connector

This topic includes the following topics:

Conditional logic for a workflow process is implemented by defining conditions and values that affect the flow of your workflow process execution. Different actions can occur depending on which path is followed. For example, you can define a condition based on the value of a priority field, so that if the priority is equal to *high*, the process execution proceeds along a branch leading to an action that sends an email to a vice president. However, if the priority is equal to *medium*, the email is sent to an engineer.

Workflow steps on which conditional logic can be defined include:

- Start step.
- Decision point.
- Wait step.
- User Interact step.

Table 21 describes example workflow processes you can view that use conditional logic.

Table 21. Examples of Workflow Processes that use Conditional Logic

Conditional Logic Description	Action Taken	Location
An IF/THEN condition, where the Revenue field in the Opportunity business component is greater than $10,000.	Insert an activity record for the parent Opportunity if the Revenue is greater than $10,000.	"Defining Condition Criteria for the Workflow Process" on page 319
A CASE statement, where one of five different branches is taken depending on the value of the Priority field in the Service Request business component.	Add an activity record is added with a commit time value that is based on the result of the CASE statement.	"Example Workflow Process That Manages Service Request Creation" on page 349

Defining a Branch Connector

Conditional logic is defined on connectors emanating out of a workflow process step, not on the step itself. Therefore, to establish conditional logic in a workflow process, you define a branch connector, which includes adding a connector to the workflow process then defining conditional logic on that connector. Typically, a *branch connector* is a connector that emanates from a Start step, decision point, Wait step, or User Interact step.

To add a branch Connector

1. Drag then drop a connector from the palette to the canvas, connecting the Decision Point with the new next step.
2. Click the connector to access the connector's properties in the Properties window.
3. Enter or modify the connector name.

   The connector name must be unique to the workflow process. If it is not unique, you cannot commit the record.
4. Choose a connector Type.

   In most cases, this is Condition or Default. Choose Condition if you define conditional logic on the connector. Choose Default if the connector is being used as an exit route. Other values can also be set. For more information, see Table 113 on page 433.

   If you are defining multiple branches on this step, for caution information see "Defining Multiple Branches on a Single Step" on page 113.
5. Enter comments.
6. Make sure in the Process Designer that the connector is correctly attached to the next step in the workflow process.

   The handles on both ends of the connector are red if they are correctly attached to adjacent steps.
7. Define the conditional logic, if necessary.

   Use the Compose Condition Criteria dialog box if the connector is used to implement conditional logic. For more information, see "Defining Conditional Logic on a Branch Connector" on page 114.

Defining Multiple Branches on a Single Step

A Start step, Decision Point, Wait step or User Interact step can each have multiple branch connectors.

To define multiple branches on a single workflow step, perform the procedure described in "Defining a Branch Connector" on page 112 for each branch you must define for the step. Note that each branch can have its own conditional logic.

CAUTION: When defining multiple branch connectors, always define at least one connector with the Type property set to Default. This provides an exit route in case a work item does not meet the conditions defined.

Comparing Branching Declaratively to Programming With Script

You can implement branching declaratively, as in multiple branches emanating from a single step in a workflow process. In some cases, you can achieve the same result through script, as in scripting on a business service.

For example, assume you have a STATUS field that references a List of Values (LOV) and this LOV itself has 120 values. When the user updates the STATUS field, a workflow process performs 100 different updates in three related business components. This workflow contains a Start step that checks for 120 conditions then uses 100 different branches that, in turn, updates the relevant business component.

As an alternative, you can configure a workflow process that contains a scripted business service. In this workflow, the STATUS is sent to the business service. The business service computes then propagates the outcome to the business components either through the same business service or through a separate Siebel operation step in the workflow process.

Conclusions that can be drawn from this example include:

- Effectively, there are no limitations on the number of outgoing branches on a start step or a decision step.
- 120 steps with 120 branches in a workflow process should provide the same performance as implementing the same logic through eScript on a business service.
- For this kind of programmatically intense operation, eScript is probably a better choice. A workflow process with 120 branches is cluttered, while the implementation would be cleaner and easier to maintain in eScript. Conversely, it is recommended to use a declarative technique to address more common requirements.

Branching and Parallel Processing

In this release, a workflow process does not support parallel processing. Make sure you define conditions so the workflow can only proceed along one connector. If conditions are defined in such a way that flow can proceed along multiple connectors, the exact run-time behavior of the Workflow engine cannot be accurately predicted.

Defining Conditional Logic on a Branch Connector

You define conditional logic on a branch connector by using the Compose Condition Criteria dialog box. A branch connector can be a connector that emanates from a start step, decision point, wait step, or User Interact step. Values listed in the Compose Condition Criteria dialog box are constrained by the value defined in the Business Object property for the workflow process. To display this dialog, you right-click a branch connector, then choose Edit Conditions.

To view an example that uses conditional logic on a connector, see “Defining Condition Criteria for the Workflow Process” on page 319.

Fields in the Compose Condition Criteria Dialog Box

Table 22 describes fields you set in the Compose Condition Criteria dialog box to define criteria for a connector that contains conditional logic.

Table 22. Description of Fields in The Compose Condition Criteria Dialog Box

Field	Description	Possible Value
Compare To	Indicates where the comparison value is coming from.	(Required). The choices available include: ■ Business Component ■ Process Property ■ Expression ■ Applet
Operation	Identifies the comparison operation.	**This Must Match.** The current value must match exactly, including case. **One Must Match.** One or more values must match exactly, including case. **All Must Match.** All of the values must match exactly, including case. **None Can Match.** None of the values can match exactly, including case. **This Must Match (ignore case).** The current value must match without regard to case. **One Must Match (ignore case).** One or more values must match without regard to case. **All Must Match (ignore case).** All of the values must match without regard to case. **None Can Match (ignore case).** None of the values can match without regard to case. **Greater Than.** Value must be greater than the comparison value. **Less Than.** Value must be less than the comparison value. **Between.** Value must be between a range of values. **Not Between.** Value cannot be between a range of values. **Is Null.** Value must be null. **Is Not Null.** Value cannot be null.

Table 22. Description of Fields in The Compose Condition Criteria Dialog Box

Field	Description	Possible Value
Object	The name of the associated business object.	This value is chosen from a picklist of business objects.
Field	This is a required value when Applet is the Compare To value.	The name of the field within the named applet.
Values	(This cell is intentionally empty.)	The Values property is dynamic based on the Compare To property. The Values property is for storing data to be used in the condition evaluation.

To define conditional logic on a branch Connector

1. In the Process Designer, right-click the connector on which you need to define conditional logic.
2. Choose Edit Conditions.

   The Compose Condition Criteria dialog box displays.
3. Compose a condition.
4. Click OK.

Building Expressions with Expression Builder

The Compose Condition Criteria dialog box uses the Expression Builder to create conditional logic. This topic describes parts of the Expression Builder, including Compare To options, operations, and patterns.

Compare To Options

Table 23 describes Compare To options and their required and optional values.

Table 23. Description of Compare To Options in the Compose Criteria Dialog Box

Compare To	Operation	Object	Field	Value
Applet. Compare the run-time value of an applet column to a literal(s).	Comparison operation. All operations allowed.	The name of the applet.	The column in the applet.	One or more literals for comparison.
Business Component. Compare the run-time value of a buscomp field to a literal(s).	Comparison operation. All operations allowed.	The name of the business component.	The field in the business component.	One or more literals for comparison.
Expression. Evaluate expressions and determine whether they return true.	Specifies how effects of multiple expressions stack. Only applicable when multiple values are specified in the Values property. Operations allowed include: ■ All Must Match ■ None Can Match ■ One Must Match ■ This Must Match ■ Four Ignore Case variations of these operations. For more information, see "Comparison Operations" on page 118.	(Optional) The name of a business component, if business component fields are referenced in the expressions. Fields of at most one business component can be referenced in the expressions.	(This cell is intentionally empty.)	One or more expressions
Process Property. Compare the run-time value of a process property to a literal(s).	Comparison operation. All operations allowed.	The name of the process property.	(This cell is intentionally empty.)	One or more literals for comparison.

Comparison Operations

This topic describes the comparison operations available.

Simple Comparison Operations

Simple comparison operations involve comparisons that can be expressed in simple expressions without the need for iterative operations. Values for these operations include:

- **Between.** The comparison succeeds if the run-time value of interest is between two predefined literal values. This comparison requires exactly two values in the Values property, which is enforced by the Process Designer.
- **Not Between.** The comparison succeeds if the run-time value of interest is not between two predefined literal values. This comparison requires exactly two values in the Values property, which is enforced by the Process Designer.
- **Greater Than.** The comparison succeeds if the run-time value of interest is greater than a predefined literal value. Logically, only one value is needed in the Values property.
- **Less Than.** The comparison succeeds if the run-time value of interest is less than a predefined literal value. Logically, only one value is needed in the Values property.
- **Is Null.** The comparison succeeds if the run-time value of interest is null/empty. No value is required in the Values property.
- **Is Not Null.** The comparison succeeds if the run-time value of interest is not null/empty. No value is required in the Values property.

Iterative Comparison Operations

Iterative comparison operations involve comparisons that involve multiple iterations on the literal values or expressions in the Values property, or iterations on child business component records, to derive a Boolean result.

The ways in which an iterative operation can be defined that results in different iterative behavior include:

- When multiple values are specified in the Values property.
- When multiple child business component records exist in the current workset. A child business component is a business component that is not the primary business component in the current business object that the workflow is working on. A child business component is involved in the comparison if the Compare To option is “Business Component” or “Expression,” and the Object field is specified as the name of a child business component. Multiple records can exist in the current workset for the child business component if multiple records were returned when the child business component was last executed.

Iterative comparison operators that are available include:

- **All Must Match**

  Multiple value behavior. If the Compare To option is anything but Expression, a literal is defined and this comparison succeeds if at least one value matches. If the Compare To option is Expression, then this comparison succeeds if every expression evaluates to true.

  Multiple child buscomp record behavior. Child business component records are used for comparison and the overall comparison succeeds only if the comparison succeeds for every one of the records.

  Consider an example where All Must Match and multiple child business components are used. In this example, Account is the business object, and the condition is described in the following table:

Compare To	Operation	Object	Field	Values
Business Component	All Must Match	Contact	First Name	John, Sam

  In this example, Contact business component records in the current workset must have a first name of John or Sam for the comparison to succeed. In this example, the workset is the child Contact record for the particular Account record the workflow is processing.

- **This Must Match**

  For the comparison to succeed, field values of the active row of the current child business component instance must match the condition values specified for the connector

  When Remote Asynchronous is used, the workflow is resumed from WfProcMgr. Because the child business component instance is not preserved in WfProcMgr, the child business component's active row becomes lost. For this reason, the conditional connector stops working. If you must use Remote Asynchronous, then use a condition operation other than *This Must Match*. Otherwise, use Local Synchronous.

  Multiple value behavior. If the values involved are literal values, the comparison succeeds when at least one value matches. If the values involved are expressions, the comparison succeeds if at least one expression is evaluated to true.

  Multiple child buscomp record behavior. Only the current child business component record is used for comparison and the overall comparison succeeds if the comparison succeeds for the current record.

- **None Can Match**

  Multiple value behavior. If the values involved are literal values, the comparison succeeds when none of the values matches. If the values involved are expressions, the comparison succeeds when none of the expressions is evaluated to true.

  Multiple child buscomp record behavior. Child business component records are used for the comparison and the overall comparison succeeds only if the comparison fails for every one of the records.

- **One Must Match**

  Multiple value behavior. If the values involved are literal values, the comparison succeeds when at least one value matches. If the values involved are expressions, the comparison succeeds when at least one expression is evaluated to true. This is the same as This Must Match.

  Multiple child buscomp record behavior. Child business component records are used for comparison and the overall comparison succeeds if the comparison succeeds for at least one of the records.

- **All Must Match (Ignore Case)**

  Same as its All Must Match counterpart except string comparisons are case insensitive.

- **This Must Match (Ignore Case)**

  Same as its This Must Match counterpart except string comparisons are case insensitive.

- **None Can Match (Ignore Case)**

  Same as its None Can Match counterpart except string comparisons are case insensitive.

- **One Must Match (Ignore Case)**

  Same as its One Must Match counterpart except string comparisons are case insensitive.

Substitution Variables Commonly Used in Expressions

A process property or business component field is often used as a substitution variable in an expression. For more information see "Referencing a Process Property" on page 85 and "Referencing a Business Component Field" on page 120.

For more information about the Operators, Expressions, and Conditions that can be used in a workflow process, see *Siebel Developer's Reference*.

Referencing a Business Component Field

The syntax is [FieldName]. For example, for the business component Contact:

```
[First Name] like 'John'
```

Where:

First Name is the name of the field.

The business component itself is specified in the Object field of the Expression Builder.

Example of Using an Expression to Compare Values

An expression can be used to compare the value of a process property Object Id to that of the Contact business component and Account Id field, as described in the following table:

Compare To	Operation	Object	Values
Expression	This Must Match	Contact	[Account Id] like [&Object Id]

8 About Workflow Process Design Options

This chapter describes several options you have when developing a workflow process. It includes the following topics:

For other topics that are also of a design nature, see “Manipulating and Processing Data” on page 413.

About the Workflow Mode Property

There are four workflow process modes that can be used to characterize run-time behavior. The mode is set by modifying the Workflow Mode property in the Workflow Processes Object List Editor (OBLE). This topic includes the following topics:

About the Workflow Mode Types

The types of available workflow modes that are described in this topic include:

- Service Flow.
- Interactive Flow.
- Long Running Flow.
- 7.0 Flow.

About the Service Workflow Process

The *Service Flow* is a type of workflow process that executes a set of operations, performing a unit of work from start to finish. A Service Flow workflow process is a *transient workflow*. A transient workflow runs to completion in a short period of time without stopping or pausing for an event or activity. A service workflow is the lowest common denominator of the workflow process mode types, in that it has no special behavior associated with it and it can be a subprocess in another service workflow, an interactive workflow, or a long-running workflow, but not a 7.0 flow. The service workflow mode simply executes a set of operations upon invocation of an event.

One example of a service workflow is a workflow that sends an email.

Restrictions when using a service workflow process include:

- A service workflow process cannot wait for run-time events or by pausing for time.
- A service workflow process cannot have User Interact steps or Wait steps. The Process Designer does not allow you to build User Interact steps or Wait steps into a service workflow process. You cannot drag then drop these step types onto the design canvas if the workflow's mode is Service Flow.

For an example, see "Example Workflow Process That Creates an Activity for a Sales Representative" on page 315.

About the Interactive Workflow Process

The *Interactive Flow* is a type of workflow process that assists and controls user navigation across Siebel views and screens. An interactive workflow includes one or more User Interact steps, and usually includes a run-time event.

For an example, see "Example Workflow Process That Manages Service Request Creation then Navigates the User" on page 358.

Running an Interactive Workflow in a UI Context

An interactive workflow can run only in the context of a user session. An interactive workflow cannot run in the Workflow Process Manager server component, nor in other server components that do not have a UI context, such as Business Integration Manager.

To be run in the context of a user session means the workflow must run within an Application Object Manager that has UI context-such as Siebel Call Center, Siebel Service, Siebel eSales, or Siebel eService. These applications can support view navigation, view display, and so forth. This is necessary for interactive workflows because they perform view display through the User Interact step and they require interaction with the user for clicking buttons and hyperlinks.

Synthetic Event Usage Within An Interactive Workflow Process

An interactive workflow process can be controlled through the use of a synthetic event attached to an explicit user interface button. A *synthetic event* is a specialized run-time event that is dedicated to controlling workflow navigation.

Examples of synthetic events include Suspend, Resume, Next, and Back.

Associated with buttons on the user interface, these synthetic events are interpreted by the Workflow engine to control workflow navigation by moving the user back or forward, and by suspending or resuming a workflow process.

Comparing Synthetic Events and User Events

A synthetic event differs from a user event. A user event is an event internal to Workflow which is used purely to resume a workflow from the WFProcMgr server process. A user event can be used only in long-running workflows.

A synthetic event is a run-time event and is used to resume a workflow from the application object manager where the synthetic events are generated. Therefore, a synthetic event can only be used for interactive and 7.0 workflows.

For more information about using synthetic events and the interactive workflow process, see "Defining an Interactive Workflow Process" on page 125.

About the Long-Running Workflow Process

The *Long Running Flow* is a type of workflow process that is a persistent workflow that can last for hours, days, or months. One example of a long-running workflow is Send Order to External. In this example, the workflow sends an order to an external system and waits for a response.

You can use the Long Running Flow mode to create a single workflow to handle an entire business process transaction to coordinate between multiple subprocesses. The Quote to Cash business process is an example transaction where Long Running Flow is useful.

You can build a long-running workflow process that is collaborative by assigning subprocesses to users. You do this by employing the Workflow User Event Service business service, which generates user events that can span from one user or session to another user or session. For more information, see "About the Workflow User Event Service Business Service" on page 381.

Note that you cannot build User Interact steps into long-running workflows, but you can build interactive workflows into long-running workflows as subprocesses. Also, you cannot simulate long-running workflows using the Process Simulator in Siebel Tools.

For more information about the long-running workflow, see "Defining a Long-Running Workflow Process" on page 133 and "Assigning a Subprocess to Users to Create a Collaborative Long-Running Workflow" on page 133.

About the 7.0 Workflow Process

The *7.0 Flow* is a type of workflow process that provides backward compatibility for existing workflows that were created in Siebel releases prior to version 7.7. If you have an existing workflow that was defined prior to version 7.7, and you upgrade to Siebel version 7.7, the existing workflow becomes a 7.0 workflow by default. For new workflows that you create you must use a service, interactive, or long-running workflow mode.

Upgrading a workflow that was defined prior to version 7.7 to version 8.0 applies the same rule because a pre-7.7 workflow must first be upgraded to version 7.7 before the upgrade to Siebel version 8.0.

NOTE: It is strongly recommended you not use the 7.0 Flow workflow mode when creating a new workflow process. Prior to Siebel version 8.1, if you do not specify a workflow mode the mode is assumed to be 7.0 Flow. When you create a new workflow process, be sure to specify a Workflow Mode other than 7.0 Flow so that 7.0 Flow is not assumed as the default mode. In Siebel version 8.1, when you create a new workflow process, the Workflow Mode defaults to Service Flow.

About Setting the Workflow Mode Property

This topic describes how the workflow mode determines the types of workflow steps a workflow process can contain.

Setting Workflow Mode With a Wait Step and a User Interact Step

A workflow process that includes a User Interact step must have the workflow's mode property set to Interactive Flow. Such a flow can contain Wait steps as long as they wait for an event, or do not wait. For example, a dummy Wait step is used to assign process property values.

A workflow process with a Wait step is a Long Running Flow only when the Wait step waits for a period of time, after which the workflow must be resumed from the Workflow Process Manager. In other cases, if the Wait step waits for a run-time event in the same user session, the flow is not a long-running workflow. This type of workflow can contain User Interact steps, which make it an interactive workflow.

Setting Workflow Mode With a Sub Process Step

A main process can call multiple subprocesses, but the workflow mode of the main process must be consistent with the workflow mode for a subprocess called by the main process.

Table 24 describes the allowed pairings of main process and subprocess mode types.

Table 24. Description of Workflow Modes Allowed Between Main Process And Subprocess

This Main Process Workflow Mode. . .	. . .Can Have These Subprocess Workflow Modes
Service Flow	Service Flow
Interactive Flow	Service Flow Interactive Flow
Long Running Flow	Service Flow Interactive Flow Long Running Flow
7.0 Flow	7.0 Flow

Defining an Interactive Workflow Process

This topic includes the following topics:

When you define an interactive workflow process, the tasks you perform include:

- Set the Workflow Mode property of the workflow process object definition to Interactive Flow.
- Set the Auto Persist flag for interactive flows that must be persisted.
- Configure business components and applets by adding buttons to applets to make use of the synthetic event that controls user navigation of a workflow process.

About the Synthetic Event

A *synthetic event* is a specialized run-time event that is dedicated to controlling workflow navigation. To control the way a user navigates through the Siebel application, you can create a button on an applet within a view then associate the synthetic event with the button.

For example, in the Account Note view of the Siebel application you are configuring, there is an Account Entry Applet on which you need to include buttons for Back, Next, and SaveWorkflow synthetic events. These buttons allow the user to move forward or backward from the Account Note view, or to suspend an interactive workflow process then return later to resume the workflow.

After you have created the buttons, you associate methods within the interactive workflow process. For example, with Back and Next synthetic events, you associate methods to outgoing connectors on the User Interact step. You set the synthetic event method name in the MethodInvoked property of the button controls.

About Forward and Backward Navigation Between Views

You can use a synthetic event to define an interactive workflow process so that when the user clicks the Next or Back button, the user is taken to the next or previous view in the sequence without losing the context of the process instance.

NOTE: Use a synthetic event to allow the user to navigate backward through views. Run-time events allow forward navigation, but not backward navigation.

Requirements that must be met for a workflow process to navigate back and forth include:

- The workflow being resumed must be an interactive workflow process or a 7.0 workflow process.
- The triggering event must be a workflow navigation event, that is, an event with a name such as InvokeMethod, and a sub-event with a name such as FrameEventMethodWFBFxxxx or EventMethodWFBFxxxx, where “xxxx” is the name of the event, such as Next.

Table 25 describes backward and forward navigation availability for workflow modes.

Table 25. Description of Backward and Forward Navigation with Workflow Modes

Workflow Mode Supported	Workflow Mode Not Supported
Modes that are supported with backward and forward navigation include: ■ Interactive Flow ■ 7.0 Flow	Modes that are not supported with backward and forward navigation include: ■ Service Flow ■ Long Running Flow

Considerations to weigh when deciding whether to allow backward navigation in your workflow process include:

- The backward navigation feature does not undo the effect of the workflow process. Backward navigation only modifies the current step counter to point to a previous step.
- The workflow configuration must make sure the segment of the workflow that can be repeated by the backward navigation feature is idempotent. That is, acts as if used only once, even if used multiple times.

Description of Synthetic Event Methods

Table 26 describes synthetic event methods.

Table 26. Description of Synthetic Event Methods

Synthetic Event Method	Description
FrameEventMethodWFNext	Moves the user forward in the interactive workflow by using an applet. You can also give the method name a prefix of BF, as in FrameEventMethodWFBFxxxx. Use this optional BF prefix to define backward and forward behavior of the synthetic event. A synthetic event with this prefix can be used to resume a workflow process from a step that is different from the current step at which the workflow process is waiting.
EventMethodWFNext	Operates the same as FrameEventMethodWFNext, except navigation is through the business component.
FrameEventMethodWFBack EventMethodWFBack	Move the user backward in the interactive workflow.

Table 26. Description of Synthetic Event Methods

Synthetic Event Method	Description
SaveWorkflow	Suspends and saves the interactive workflow and makes it appear in the user's Inbox.
ResumeLastIntFlow	Resumes the last executed interactive workflow. ResumeLastIntFlow is different from the other events in that it is not tied to a specific workflow process and can be invoked from anywhere in the Siebel application. That is, the button corresponding to this event can be put in an applet, including the task bar where the Site Map icon is located, which is the recommended location for this button.

Procedures for Creating Synthetic Events

This topic describes procedures for creating synthetic events.

To create a synthetic event button for next and back events

1. In Siebel Tools, identify a view to which a User Interact step navigates the user.
2. Configure a Next button or a Back button on an applet where the event is triggered. For more information, see *Using Siebel Tools*.

   The button must be exposed in the applet that is based on the primary business component associated with the business object defined for the workflow process. For example, if the workflow process is based on the Service Request business object, and if the view displays Service Request Activities records, then the button must be exposed in the applet based on the Service Request business component. In this example, the synthetic event button does not work if the button is exposed in the applet based on the Activities business component.
3. Specify the MethodInvoked property of the button control as the name of the associated event. For example, FrameEventMethodWFBack for backward navigation.

4 In the Process Designer, associate the applet type run-time event.

For example, assign FrameEventMethodWFBack to the outgoing connectors of the User Interact step in the workflow process that receives the event. Assign the event using values described in the following table:

Property	Value
Event Type	Applet
Event Obj	AppletName
Event	InvokeMethod
Sub Event	[Method Name] For example, FrameEventMethodWFBack.

TIP: You do not have to manually create buttons for each applet. You can copy a button you have created to other applets by using the Applet Comparison capability in Siebel Tools. Also, if you add the applet button controls to the HTML Model Controls applet, when you create new applets with the New Applet Wizard or the conversion process, you can then choose the Workflow related method buttons.

To create a synthetic event button for the saveworkflow event

1 In Siebel Tools, identify a view to which a User Interact step navigates the user.

2 Configure a Save button on an applet where the event is triggered. For more information, see *Using Siebel Tools*.

3 Specify the MethodInvoked property of the button control as the name of the associated event, SaveWorkflow.

4 Use the script below to invoke the Workflow event handler to handle the button click event. The script also passes to the Workflow event handler the event's contextual information, which is the name of the view where the event occurs.

Note that it is not necessary for you to define the event in the workflow process definition.

```
function WebApplet_InvokeMethod (MethodName)

{

 return (ContinueOperation);

}


function WebApplet_PreCanInvokeMethod (MethodName, &CanInvoke)

{

// Recognize SaveWorkflow event, which is

// used to save Interactive flow
```

```
if (MethodName == "SaveWorkflow")
{
CanInvoke = "TRUE";
return (CancelOperation);
}

return (ContinueOperation);
}

function WebApplet_PreInvokeMethod (MethodName)
{
// Handle SaveWorkflow event.
// Call Workflow Process Manager to save the interactive
// flow(s) that is waiting in the current view.
if (MethodName == "SaveWorkflow")
{
var Inputs= TheApplication().NewPropertySet();
var Outputs = TheApplication().NewPropertySet();

// Event name ("SaveWorkflow"), view name, and the rowId
// of the active row of the underlying buscomp are
// three required parameters for handling the event
Inputs.SetProperty("Event Name", MethodName);
var viewName= TheApplication().ActiveViewName();
Inputs.SetProperty("Sub Event", viewName);
var bc = BusComp ();
var bcId = bc.GetFieldValue ("Id");
Inputs.SetProperty("RowId", bcId);

var workflowSvc= TheApplication().GetService("Workflow Process Manager");
```

```
workflowSvc.InvokeMethod("_HandleSpecialEvent", Inputs, Outputs);


return (CancelOperation);


}


return (ContinueOperation);
}
```

To create a synthetic event button for the resumelastintflow event

1. In Siebel Tools, identify a view to which a User Interact step navigates the user.
2. Configure a Resume button on an applet where the event is triggered. For more information, see *Using Siebel Tools*.
3. Specify the MethodInvoked property of the button control as the name of the associated event, ResumeLastIntFlow.
4. Use the script below to invoke the Workflow event handler to handle the button click event. The script also passes to the Workflow event handler the event's contextual information, which is the name of the view where the event occurs.

   Note that it is not necessary for you to define the event in the workflow process definition.

```
function WebApplet_InvokeMethod (MethodName)
{
return (ContinueOperation);
}


function WebApplet_PreCanInvokeMethod (MethodName, &CanInvoke)
{


if (MethodName == "ResumeLastIntFlow")
{
CanInvoke = "TRUE";
return (CancelOperation);
}
```

```
return (ContinueOperation);
}

function WebApplet_PreInvokeMethod (MethodName)
{
// Call Workflow Process Manager to resume the last-executed interactive flow
if (MethodName == "ResumeLastIntFlow")
{
var Inputs= TheApplication().NewPropertySet();
var Outputs = TheApplication().NewPropertySet();
var workflowSvc= TheApplication().GetService("Workflow Process Manager");
workflowSvc.InvokeMethod("_ResumeLastInteractFlow", Inputs, Outputs);

return (CancelOperation);

}

return (ContinueOperation);
```

About Suspending and Resuming an Interactive Workflow Process

An interactive workflow process that is suspended can be resumed from within the user's Inbox. The user can navigate out of an interactive process, then navigate back into the workflow process and continue where the user suspended the workflow.

For example, assume a transaction involving the user, an insurance agent, cannot be finished because some information is missing, such as a spouse's social security number which is required for an insurance policy quote to be considered finished. In this example, when the insurance agent eventually obtains the social security number after suspending the interactive workflow process, the insurance agent can resume the process from within the Inbox, then enter the social security number to finish the entry of the quote. Once the process is finished, the Workflow engine removes the interactive workflow process from the Inbox.

A suspended interactive workflow is placed in the workflow owner's Inbox for tracking and explicit resumption.

Table 27 describes actions taken when a workflow is suspended.

Table 27. Description of Action Taken When a Workflow is Suspended

How Workflow is Suspended	Description	Result
explicit suspension	The user clicks the Suspend button.	The workflow is saved to the database and placed in the Inbox.
implicit suspension	The user leaves a view that is part of the workflow process.	If, the workflow must be removed from the in-memory cache, then: ■ The workflow is placed in the Inbox ■ The workflow is saved to the database An example of when the workflow must be removed from the in-memory cache is when the cache is full or when the user logs out.

Considerations to weigh include:

- To realize the behavior described in Table 27, the suspended workflow must have its Auto Persist flag checked.
- A suspended interactive workflow in the Inbox is removed from the Inbox when the workflow has run to its end and terminates.

Suspending a Workflow Compared to Persisting a Workflow

There is a difference between suspending a workflow and persisting a workflow. When a workflow is suspended, it can be persisted, but only when the workflow's Auto Persist flag is set to TRUE.

For example, with implicit suspension, if a user leaves a view the workflow took the user to, the workflow instance is saved in the in-memory cache. In this case, no Inbox item is generated. When the user logs out, Inbox items are generated for workflows in the cache that have the Auto Persist flag set to TRUE. So the Inbox items are visible only when the user logs out then logs back in.

In-Memory Cache of a Suspended Interactive Workflow

A user often navigates out of a structured interactive workflow because the workflow is set up to address your specific business needs. When this happens, the interactive workflow remains in memory so it can be resumed later in the same user session. As it is uncommon for a user to have a large number of unfinished tasks at hand, there can be a maximum of eight suspended interactive workflow instances in the memory cache.

This eight instance limit applies to an individual user session. The limit is eight instances total for interactive workflows initiated from within the user session, not eight instances for each interactive workflow. This eight instance limit is not configurable.

The user session is one connection on the Application Object Manager component, regardless of whether this thread is logged as a single user. When the eight instance limit is reached, new interactive workflows are not prevented from running. Instead, if the user initiates another interactive workflow after eight instances are already cached in memory, and this ninth instance must be saved to the cache, then the oldest instance is pushed into the database if the workflow's AutoPersist flag is set, or dropped as the memory cache has reached the cache's limit.

Event Handling With a Suspended Interactive Workflow

Workflow handles events in the following sequence:

1. Checks the in-memory cache to see if there are workflow instances in the cache that can receive an event, using the matching criteria specified by the event.
2. Checks the database to see if there are persisted workflow instances that can receive an event.
3. Resumes instances found in Step 1 and Step 2.

Detection and Handling of the User Logout Event for a Suspended Interactive Workflow Process

Upon receiving the user logout event, the Workflow engine goes through suspended interactive workflows in the in-memory cache. Workflows with the Auto Persist flag checked are saved as Inbox items. Other workflows are deleted.

Defining a Long-Running Workflow Process

This topic includes the following topics:

A long-running workflow process is a persistent workflow that can last for hours, days, or months. An example of a long-running workflow process is an approval process that sends an order to an external system such as SAP, then waits for a response. For more information, see "About the Long-Running Workflow Process" on page 123.

NOTE: When building a long-running workflow process, use *user events* and not *run-time events* to trigger a workflow process and resume a workflow process instance.

Assigning a Subprocess to Users to Create a Collaborative Long-Running Workflow

Using the Sub Process step, you can configure a workflow that assigns an interactive subprocess workflow to users to create a collaborative workflow process. An example of a collaborative workflow is one that includes a requirement for approvals. The route the workflow takes as it moves through workflow activities is a route across multiple users who work in collaboration to finish the job tasks required by the workflow process.

You use the Recipients properties on a Sub Process step to create collaborative workflows. Assignment occurs based on the login name, not on the Position or User ID. This login name can be a literal value, it can be held in a process property or a buscomp field, or it can be the result of an expression.

NOTE: The Process Designer cannot validate the data supplied to make sure it represents a valid login name at design time.

To assign a Sub Process to a user

1 Create a Sub Process step.

2 In the Recipients tab of the MVPW, set the Recipient Name argument field to the login name of the user who is assigned the subprocess.

For more information, see "About Process Properties" on page 74, and "Argument Fields for a Recipient Argument" on page 442.

Configuring a Long-Running Workflow to Invoke a Task

You can use a long-running workflow to assign a task to a user, then create an inbox item for the task and push it to the user's Inbox. The user can then click the inbox item to create a new instance of the task and run it. For example, in an Expense Report *Approval* long-running workflow, after the employee submits an expense report, a new task called *Review Expense Report* is created and assigned to the employee's manager. Because this case is a one-to-one assignment, the assignment can be made simply by looking up the manager's ID from the user's business component. After the manager's ID is retrieved, a new item is created in the manager's Inbox that refers to the *Review Expense Report* task.

To use a long-running workflow to assign a Task to a user

1 In the long-running workflow, at the point where a task needs to be called, determine the ID of the user to whom the new task instance is assigned.

This logic is dependent on your business requirements, and it can be implemented as a Siebel operation in cases where the ID is already present in a business component. The logic can also be implemented as a business service call.

When the assignment logic implies application of assignment rules, a business service interface to Siebel Assignment Manager can be used. The input to the service is variable, depending on the context that is needed for the assignment. The output, however, is always simply the ID of the user to whom the task is assigned.

2 Define a task step that creates a new item in the Inbox of the user that the task is assigned to, specified as the input argument Owner Id.

Adding a Task Step to a Workflow Process

When you add a Task step to a workflow process, the workflow creates a new UI task and assigns it to a user.

To add a Task step within an existing workflow process, open the Process Designer for the workflow. The Status property of the workflow is *In Progress*. If Status is not In Progress, revise the workflow process so that an editable *In Progress* copy is created.

For details on how to configure a Task step, see "Defining a Task Step" on page 106.

Using Workflow Persistence

Workflow persistence is a feature used to store the state of a workflow process instance and the workflow's steps and process properties. The workflow process state and process properties are saved in the S_WFA_INSTANCE and S_WFA_INST_PROP tables.

By using workflow persistence to store the state of a workflow process, you can build an end-to-end workflow which includes Wait steps, Sub Process steps, and other interruptions. Persistence also maintains the active state of a process over short or long periods of time, with activity occurring in various parts of your enterprise.

When persistence is set to TRUE, a user can continue with a workflow process that is suspended. The suspended workflow appears in the user's Inbox.

About Workflow Persistence

Workflow persistence is a property of a workflow process that supports long lived transactions within a single workflow process. Workflow persistence allows process resumption after a pause or a server failure. Workflow persistence saves and restores data when the process is resumed.

For example, with persistence set on an interactive workflow, a user of the workflow can set aside their work and take a break. If the workflow persistence parameter times out, the workflow will be in the user's Inbox ready for resumption when the user returns.

The workflow persistence setting applies to the long-running workflow process, interactive workflow process, and 7.0 workflow process. You cannot use workflow persistence with a service workflow process.

The persistence property is a YES/NO setting. For long-running workflows, persistence is automatically set by the server at execution time. For interactive workflows, you set the persistence property using the Auto Persist flag in the Workflow Processes list editor in Siebel Tools.

Behavior for persistence includes:

- A long-running workflow is automatically persisted.
- An interactive workflow is persisted on suspension, if the Auto Persist flag is set.

  You control workflow persistence by setting the Auto Persist flag. When a session times out or a user logs out of a Web session, a workflow process with the Auto Persist flag set to YES is persisted and can be resumed from the Universal Inbox.
- A 7.0 workflow with persistence set is marked as Auto Persist and is persisted.

NOTE: In some releases prior to Siebel version 8.0, workflow persistence was used for monitoring workflow processes, and persistence was controlled by adjusting two settings that can apply to individual steps of a workflow: frequency and level. With version 8.0 and higher, process monitoring is separate from workflow persistence, and it is not necessary for persistence to be set for a long-running process, because the long-running process is set by default. For 7.0 workflow processes that had persistence set, the Auto Persist flag is automatically set to YES during upgrade and import.

Note that a persisted workflow process is automatically purged once it finishes running.

Modifying Workflow Persistence

For a long-running workflow process, persistence occurs by default. For an interactive workflow process, you set the Auto Persist flag in the Process Designer within Siebel Tools.

CAUTION: Defining Workflow Persistence for a large number of workflow processes can result in the creation of a large number of records in S_WF_PROP_VAL. For more information, see"Disabling Persistence to Avoid Excessive Records in S_WF_PROP_VAL" on page 264.

To configure workflow persistence for an interactive workflow process

1. In Siebel Tools, in the Object Explorer, click the Workflow Process object type.
2. In the Workflow Processes OBLE, click the process you need to work with.
3. In the Auto Persist property, choose YES using the drop down picklist.

Invoking a Workflow Process

This topic describes different ways to invoke a workflow process. It includes the following topics:

Invoking a Workflow Process from a Workflow Policy

To invoke a workflow process from a workflow policy, you define a policy action that uses the Run Workflow Process workflow policy program. Alternatively, you can create a custom workflow policy program by copying Run Workflow Process, then adding program arguments that correspond to workflow process properties. This way you can use the policy program to pass data to the workflow process properties.

For information about defining a workflow policy, see Chapter 9, “About Workflow Policies”.

To invoke a workflow process from a workflow policy

1. In the Siebel client, navigate to Administration-Business Process > Actions.
2. In the Actions applet, click New to define a new action.
3. In the Program field, use the picklist to choose the Run Workflow Process program.
4. In the Arguments applet, click New to create a new Argument.
5. in the Argument field, choose ProcessName from the picklist.
6. In the Value property, enter the name of the workflow process you need to invoke.
7. Navigate to Administration-Business Process > Policy Groups.
8. In the Policy Groups applet, click New to create a new group, then name it.
9. Navigate to Administration-Business Process > Policies.
10. In the Policies List applet, click New to define a new policy.
11. In the Conditions applet, click New to define a condition for the policy that must be met to invoke the workflow process.

12 In the Actions applet, click New to define a new action, then enter the name of the action you defined in Step 2.

13 Run Generate Triggers.

For more information, see "Creating Database Triggers" on page 276.

14 If you are using the Run Workflow Process program, make sure the Workflow Process Manager server component is online.

15 Run Workflow Monitor Agent.

For more information about monitoring workflow policies, see "Executing a Workflow Policy with Workflow Monitor Agent" on page 290.

16 Violate the policy. The action should invoke the workflow process.

Invoking a Workflow Process from a Run-Time Event

This topic provides information about using a run-time event to invoke a workflow process. For more information, see "About the Run-Time Event" on page 150.

How a Run-Time Event Invokes a Workflow Process

This topic describes the simple process the Siebel application uses to invoke a workflow process with a run-time event. For this example, assume the PreWriteRecord event is used to invoke the workflow process.

A run-time event invokes a workflow process as described in the following sequence:

1 The PreWriteRecord run-time event is detected by application object manager components.

2 Since the PreWriteRecord run-time event is defined on the connector emanating out of the start step, the workflow process is invoked.

3 The object manager internally calls the Workflow engine. At the time of the call, the business object that caused the run-time event is passed to the Workflow engine.

4 The Workflow engine uses the active rows on the business object for data operations.

5 The Workflow engine sets up the rows if they are not already active.

This is applicable for an object manager that generates run-time events. Workflow runs with the local thread, and if SWE is present, user interacts are allowed. The parameters used include:

- Context, a CSV string internally registered by Workflow.
- The business component causing the run-time event.

Choosing Between a Run-Time Event and a Workflow Policy

In cases where it is necessary to detect a database event, use a workflow policy, not a run-time event. For example, when using the UI, use a run-time event to trigger a workflow process. When using the Siebel EAI Adapter or the EAI UI Data Adapter, which performs numerous WriteRecord events, use a workflow policy.

For more information, see "Invoking a Workflow Process from a Workflow Policy" on page 137.

Example Requirements for Using a Run-Time Event Compared to a Workflow Policy

Table 28 describes two different requirements for using a run-time event compared to a workflow policy.

Table 28. Examples of Requirements for Using a Run-Time Event Compared To a Workflow Policy

Workflow Requirement	Recommended Invocation Technique
Set the priority to ASAP and send a notification email whenever a new service request is created.	Using a workflow policy is a more appropriate way to detect the database event.
Generate a text file with some information whenever a particular entry applet is viewed.	Using a run-time event is more appropriate. Create a workflow process with the InvokeMethod run-time event defined on the Start step that detects when the Entry Applet is viewed. Use a business service step in this workflow process to generate the text file.

Invoking a Workflow Process With a Button

You can use Siebel Tools to configure a button on an applet that triggers the run-time event that invokes a workflow process.

To create a button that invokes a workflow process

1. From Siebel Tools, configure a button on an applet and specify the MethodInvoked property.

   For more information, see *Using Siebel Tools*.

2. Override the PreCanInvokeMethod property.
3. Edit the server script and compile your changes to the Siebel repository file.

   The following example is specific to Siebel VB:

```
Function WebApplet_PreCanInvokeMethod (MethodName As String, CanInvoke As String)
As Integer

If MethodName = "<Name>" Then

   CanInvoke = "True"

   WebApplet_PreCanInvokeMethod = CancelOperation
```

```
Else

   WebApplet_PreCanInvokeMethod = ContinueOperation

End If

End Function
```

4 Create a workflow process.

5 Choose an invocation method.

To invoke this workflow process from a button click, you must specify a run-time event on a connector. Choose from one of the following invocation methods:

- To start this workflow process, specify the run-time event on the connector emanating from the Start step.
- To resume this workflow process if it is paused as the result of a button click, specify the run-time event in a User Interact step or a Wait step.

6 Specify the run-time event using values described in the following table:

Property	Value
Event	PreInvokeMethod
Event Cancel Flag	True. If this flag is not checked, an error results when running the workflow process.
Event Object	Name of the business component on which the applet that contains the button is based.
Event Object Type	BusComp
Sub Event	Name of the method you set in step Step 1. The Sub Event name must be unique.
Type	(Required) Condition

7 Activate the workflow process.

8 In the Siebel client, navigate to Administration-Runtime Events > Events.

9 From the Menu button on the Events applet, choose Reload Runtime Events.

You must reload personalization after activating a workflow process that registers a run-time event for the workflow process to take effect.

Configuring a Workflow Process to Avoid the Cannot Resume Process Error Message

Requirements that must be met to return an error when a user clicks a button that invokes a workflow include:

- The workflow process must be an interactive flow.
- The workflow process must be started at least once in the current user session.
- An event must be fired and there must be no workflow instance waiting to handle the event.

If these conditions are met, the event is discarded or the user receives an error message.

You can prevent this from happening by requiring the user to run a workflow process to access this view. For example, you can configure a workflow process that includes the User Interact step. You can handle the error by defining an Error Exception branch or error process.

Identifying A Predefined Workflow Process Invoked by a Run-Time Event

In your implementation, it is possible that some predefined workflow processes are invoked by a run-time event. You can use the Action Set Name field of the Run-time Events administration screen in the Siebel client to identify a predefined workflow process that is invoked by a run-time event.

To identify a predefined workflow process that is invoked by a run-time event

1. In the Siebel client, navigate to Administration-Runtime Events > Events.
2. Note the row ID value in the Action Set Name field.

   For example, if Workflow_1-2APXH appears in the Action Set Name field, then 1-2APXH is the row ID of the workflow process being referenced.
3. Navigate to Administration-Business Process > Workflow Processes.
4. In the Name field, query for the row ID of the workflow process you noted in Step 2.

   For example, [Id]='1-2APXH'.

   The query returns the record for the workflow process.

Invoking a Workflow Process from a Configured Business Service

You can invoke a workflow process by defining a configured business service. With a configured business service, such as *Workflow Process Manager (Server Request)*, it is not necessary for you to specify Server Request Broker (SRBroker) parameters.

When you invoke the Server Requests directly, you must specify SRBroker parameters. SRBroker and SRProc are required to run a business service that invokes a server component.

To use *Workflow Process Manager (Server Request)*, SRProc and SRBroker must be running. For more information, see *Siebel System Administration Guide*.

For more information on invoking a business service, see *Siebel Object Interfaces Reference*.

To define a configured Business Service

1. In Siebel Tools, in the Object Explorer (OE) click the Business Service object type.
2. In the Business Services OBLE, add a new business service object definition using values described in the following table:

Property	Value
Name	(This is the name that you can reference in scripting.)
Class	CSSSrmService
Display Name	(This is the name that you see in Workflow views.)

3. In the OE, expand the Business Service object type, then click the Business Service User Prop object type.
4. In the Business Service User Props OBLE, add two new object definitions using values described in the following table:

Property	Value
Component	(Short name of the server component. For example, *WfProcMgr*.)
Mode	(Mode of the server request. For example, *Async*.)

5. (Optional) Enter more user properties pertaining to SRBroker. For more information, see "About the Server Request Broker" on page 37.
6. In the OE, choose the Business Service Method object type.
7. In the Business Service Methods OBLE, add a new object definition using values described in the following table:

Property	Value
Name	(This is the name that you can reference in scripting.)
Display Name	(This is the name that you see in Workflow views.)

8. In the OE, expand the Business Service Method object type, then choose the Business Service Method Arg object type.

9 In the Business Service Method Arguments OBLE, add records specific to the component being invoked.

For example, *ProcessName* for *WfProcMgr*. Note that the name is the short name of the server component parameter.

Invoking a Workflow Process from Another Workflow Process Asynchronously

You can configure a workflow process so that it asynchronously invokes another workflow process directly. To do this, you invoke a workflow process, which calls a custom business service, which in turn invokes the second workflow process using the Asynchronous Server Requests business service.

To invoke a workflow process from another workflow process asynchronously

1 In Siebel Tools, in the Process Designer, add a Business Service step to your workflow process.

2 With the new business service step chosen, use the properties window to set values described in the following table:

Property	Value
Business Service	Asynchronous Server Requests
Method	SubmitRequest

3 In the MVPW, click the Input Arguments tab, then add three new input arguments using values displayed in the following table:

Input Argument	Type	Value	Property Name
Component	Literal	WfProcMgr	(leave empty)
WfProcMgr.ProcessName	Literal	AG Simple Test	(leave empty)
WfProcMgr.RowId	Process Property	(leave empty)	Siebel Operation Object Id

For more information, see “About Process Properties” on page 74.

4 To set component parameters, such as Workflow Process Name, specify the Input Argument Name as [ComponentAlias].[Business Service Method Arg Name].

When the workflow process is executed in the Process Simulator, a request is inserted into S_SRM_REQUEST and a Workflow Process Manager task is started on the server.

Invoking a Workflow Process That Runs in the *Workflow Process* Manager

A workflow process can be run within the Workflow Process Manager server component. The ways in which a workflow process can be invoked on the server include:

- From a workflow policy that executes on the server.
- From a script that specifies the Server Request parameter.
- From a run-time event with the Processing Mode property set to Remote Synchronous or Remote Asynchronous.

If you compiled a custom .srf file using Siebel Tools, this file needs to be added to the Objects directory on the Siebel Application Server. In addition, you must update the siebel.cfg file referenced in the Server parameters to reflect the custom .srf file. Note that the siebel.cfg file is the default configuration file for Workflow Process Manager server components.

Considerations to weigh include:

- Invoking a workflow process from a script is performed in synchronous mode.
- A business service that calls a UI element, including navigation functionality such as the User Interact step, is not supported when a process runs on the server.

For more information, see "About the Workflow Process Manager" on page 34

About Remote Synchronous Processing

If a user invokes a workflow process that runs on the server within the Workflow Process Manager server component, the process executes only if the user is connected to the server. If the user is not connected to the server, the request is queued and executes when the user synchronizes or the server becomes available. For more information, see "About the Server Requests Business Service" on page 374.

About Key-Based Routing with Workflow Process Manager

A server key map defines the rule groups that load and process for each server. You can configure a server to load multiple rule groups. When you define server key maps, you are actually dividing the rules among the different servers. Server key maps are defined in the Server Key Map view in the Assignment Administration screen.

You submit an assignment request by specifying the AsgnKey parameter, where the AsgnKey parameter is the row id of the assignment rule group that is associated with the rules you need to evaluate. When using AsgnSrvr, the AsgnKey parameter must be the row id of one of the rule groups defined for the server in the Server Key Mappings view. The assignment server (AsgnSrvr) first looks for entries in the server key map for a specific server, and then loads rules for only those rule groups associated with that server key map. Assignment Manager uses key-based routing to route the request to a particular instance of Assignment Manager where the rules are loaded for that rule group.

You can specify multiple servers to load the same rule group. Assignment Manager routes requests to one of the servers where that rule group is based on load balancing metrics.

Environments in which the server key mapping feature is supported include:

- Where there is an interactive and dynamic assignment.
- Where a script or workflow process calls a business service.

Reasons for using Key Based Enabled include:

- **Routing.** You can setup the RequestKey parameter on WfProcMgr so that when workflow comes up, it registers this key. This way you can control the routing of WfProcMgr requests.
- **Recovery.** To target the ping messages, each WfProcMgr registers a unique key. The recovery manager uses these to keep track of which WfProcMgr is alive and what WfProcMgr is processing.

In general, if Recovery Manager is activated, then the Key Based Enabled parameter for WfProcMgr should be set to true.

For more information on key-based routing, see *Siebel Assignment Manager Administration Guide*.

About Workflow Process Manager and Automatic Record Insertion

When the Workflow Process Manager inserts a record, by default, the system generated CREATED_BY field is set to 0-1, which is the ROW_ID for employee SADMIN. In some cases, it can be desirable to have the CREATED_BY field set to other than SADMIN's Id.

For example, assume you have a workflow that inserts an activity record when a user performs an action in the Siebel client. Since this workflow is invoked by the Asynchronous Server Requests business service, the workflow is executed by the Workflow Process Manager on the server, and CREATED_BY is set to SADMIN, although the business requirement is to have CREATED_BY set to the user who caused the workflow to invoke.

In this situation, actions you can take include:

- Instead of invoking the workflow on the server using Workflow Process Manager, invoke the process so that it runs locally. As a result, the user's session creates the record and CREATED_BY are set to the user's Id. In this case the workflow process is executed synchronously.
- If the requirement is to record which user caused a record to be inserted, you can pass the user's Login Id or Login Name to the workflow process on the server then write it to an extension field in the business component. You can obtain the user's Login Id or Login Name in a script by using the standard functions LoginId () or LoginName (). For information about how to pass data to a custom workflow process property when submitting a server request, see "About the Server Requests Business Service" on page 374.

Invoking a Workflow Process That Runs in the *Application* Object Manager

Running a workflow process in the application object manager can be useful for enforcing business processes with mobile users or for defining business processes that involve end-user navigation.

Ways in which a workflow process in the application object manager can be invoked include:

- From the Process Simulator.
- From a script specified to run locally in the application object manager.
- From a run-time event with Processing Mode specified as local synchronous.

Invoking a Workflow Process Through Script

A workflow process can be invoked from a script using Siebel VB or Siebel eScript. By using a script, a workflow process can be invoked from anywhere in the Siebel application or from external programs.

When invoking a workflow process from script, you can specify that the process run on the server or in the object manager:

- To run a process on the server, call the service *Workflow Process Manager (Server Request)*.
- To run a process in the application object manager, call the service Workflow Process Manager.

Note that invoking a workflow process from script is performed in Synchronous mode.

Example of Invoking a Workflow Process from Script in the Object Manager

This topic gives one example of invoking a workflow process from script in object manager. You might use this feature differently, depending on your business model.

The following is a sample script that invokes a workflow process called My Account Process. In this example, the process is invoked in the object manager.

```
/ Example:  Invoking a Workflow Process through scripting
function Invoke_Process()
{
var svc = TheApplication().GetService("Workflow Process Manager");
var Input = TheApplication().NewPropertySet();
var Output = TheApplication().NewPropertySet();
var bo = TheApplication().ActiveBusObject();
var bc = bo.GetBusComp("Account");
var rowId = bc.GetFieldValue("Id");

Input.SetProperty("ProcessName", "My Account Process");
Input.SetProperty("Object Id", rowId);

svc.InvokeMethod("RunProcess", Input, Output);
}
```

Example of Invoking a Workflow Process from Script to Pass Field Values to Process Properties

This topic gives one example of invoking a workflow process from script to pass field values to process properties. You might use this feature differently, depending on your business model.

The following script invokes a workflow process called My Opportunity Process. In this example, the workflow process is invoked in the object manager. Field values are passed to process properties defined in the workflow process.

```
//Example:  Passing Field Values to Process Properties
function Invoke_Process()
{
   var svc = TheApplication().GetService("Workflow Process Manager");
   var Input = TheApplication().NewPropertySet();
   var Output = TheApplication().NewPropertySet();
   var bo = TheApplication().ActiveBusObject();
   var bc = bo.GetBusComp("Opportunity");

   var rowId = bc.GetFieldValue("Id");
   var accountId = bc.GetFieldValue("Account Id");

   Input.SetProperty("ProcessName", "My Opportunity Process");
   Input.SetProperty("Object Id", rowId);

   // Pass the value of the Account Id field to the Account Id process property
   Input.SetProperty("Account Id", accountId);

   svc.InvokeMethod("RunProcess", Input, Output);
```

Invoking a Workflow Process from Script Using the Server Requests Business Service

You can invoke a workflow process asynchronously from script by using the Server Requests business service. For more information, see "Example of Using the Server Request Business Service to Invoke a Workflow From Script and Pass Values" on page 378.

Example of Invoking a Workflow Process from a Custom Toolbar

This topic gives one example of invoking a workflow process from a custom toolbar. You might use this feature differently, depending on your business model.

To invoke a workflow process from a custom toolbar button

1 In Siebel Tools, navigate to the Business Services OBLE, then create a new business service object definition using values described in the following table:

Property	Value
Name	TestTBItem
Server Enabled	TRUE

2 Navigate to the Commands OBLE, then create a new command object definition using values described in the following table:

Property	Value
Name	TestTBItem
Business Service	TestTBItem
Method	TestTBItem
Target	Server

If necessary, expose the Command object type in the OE. From the application-level menu choose View > Options, then click the Object Explorer tab. In the Object Explorer Hierarchy window, make sure the Command hierarchy has a check mark. Click OK.

3 In the Object Explorer, choose the Toolbar object type. In the Toolbars OBLE, query for the HIMain toolbar object definition.

If the Toolbars object type is not exposed in the Object Explorer, from the Tools application-level menu choose View > Options, click the Object Explorer tab, then click the checkbox next to Toolbar.

4 In the Object Explorer, expand the Toolbar object type, then choose Toolbar Item.

5 In the Toolbar Items OBLE, add a new record using values described in the following table:

Property	Value
Name	TestTBItem
Command	TestTBItem
DisplayName	TestTB Item
Type	Button
HTML Type	Button
Position	20

6 In the Siebel client, create a workflow process and make sure it can be run successfully from the process simulator.

For more information, see “Preparing and Using the Process Simulator” on page 232.

7 Add the server script below to the business service you created in Step 1.

NOTE: The workflow process name you used in Step 6 must be added in the script.

```
function Service_PreCanInvokeMethod (MethodName, &CanInvoke)

{

if ( MethodName == "TestTBItem" )
```

```
{
CanInvoke = "TRUE";
return( CancelOperation );
}
return (ContinueOperation);
}

function Service_PreInvokeMethod (MethodName, Inputs, Outputs)
{
if ( MethodName == "TestTBItem" )
{
var input = TheApplication().NewPropertySet();
var output = TheApplication().NewPropertySet();
var svc = TheApplication().GetService( "Workflow Process Manager" );
input.SetProperty("ProcessName", "<WF Process Name Created in Step 4>");
svc.InvokeMethod("RunProcess", input, output);
input = null;
output = null;
svc = null;
return (CancelOperation);
}
return (ContinueOperation);
}
```

8 Compile the changes.

9 Launch the Siebel client.

Your new custom button displays in the custom toolbar. When you click it, the corresponding workflow is invoked.

Other Invocation Techniques

Information sources for other methods that can be used to invoke a workflow process include:

- For invoking from a synthetic event, see "About the Synthetic Event" on page 125.
- For invoking from a user event, see "About the User Event" on page 155.
- For invoking from the Process Simulator, see "About the Testing Tools" on page 224.
- For invoking from a server component, see *Overview: Siebel Enterprise Application Integration* and *Siebel eMail Response Administration Guide*.

You can use these techniques to test a workflow process in your test environment before deploying it to the production environment. When testing, being able to invoke from a workflow policy is important because it tests invoking a workflow process on the server. You can also use these techniques to invoke a workflow process in your production environment.

About Business Integration Manager

It is recommended you not use Business Integration Manager and Business Integration Batch Manager. If you were using either one in your business processes you must replace them with Workflow Process Manager or Workflow Process Batch Manager, respectively.

About Events

This topic includes the following topics:

About the Run-Time Event

A workflow process can integrate with the run-time events engine to provide a simplified way to automate a business process. Benefits provided by this technique include:

- Allows real-time monitoring of events.
- Minimizes the need for scripting and workflow policy invocation.

The types of events that are possible include:

- Application.
- Business Component.
- Applet.

A run-time event allows the Siebel application to respond in real time to user actions. A run-time event can be specified in the connectors that emanates from for Start, Wait, or User Interact steps to invoke or resume a workflow process. The properties of the WF Step Branch that are used to define a run-time event include:

- Event Object Type.
- Event Object.
- Event.
- Sub Event.
- Event Cancel Flag.

For more information, see"Invoking a Workflow Process from a Run-Time Event" on page 138.

A run-time event provides one way to invoke a workflow process. For a comparison of invocation methods, see "Invoking a Workflow Process" on page 137.

For more information on run-time events, see *Siebel Personalization Administration Guide*.

Using a Run-Time Event With a Long-Running Workflow

NOTE: It is recommended that a run-time event not be used to trigger a long-running workflow because a run-time event is specifically attached to a single user and a single session.

A run-time event is only for that single user, as it stems from Personalization functionality. Instead, use an interactive workflow or a service workflow to handle the run-time event. Then, after processing and validating the workflow, generate a user event to notify a long-running workflow.

Using a Run-Time Event With the User Interact Step

Events not supported with the User Interact step include:

- The DisplayRecord event.
- The DisplayApplet event.
- The Login event. Use the WebSessionStart event instead.

The event supported with the User Interact step includes:

- The SetFieldValue event for a field that has the Immediate Post Changes property set to TRUE.

  Immediate Post Changes is a property on a business component field that causes an immediate roundtrip to the server if a change is made to the field. This dictates an immediate re-calculation of the field or refresh of the view. When Immediate Post Changes is set to True the browser script PreSetFieldValue event is bypassed. For more information, see *Configuring Siebel Business Applications*.

Using a Run-Time Event Within a Business Object Context

A workflow process that references a run-time event is only invoked when the run-time event is detected from within the same Business Object context in which the workflow process is based.

For example, assume a workflow is invoked by the WriteRecord run-time event and that the workflow's Business Object property is set to Service Request. To update the record, the user clicks the Service Requests List screen tab, updates the Status field, then steps off the record. In this case, the record is written in the context of the Service Request business object, and the run-time event that is defined on the workflow process fires.

However, the run-time event does not fire if the Status field is updated within a non-Service Request business object context. For example, assume the user drills down on a Contact, clicks the Service Requests view tab, updates the Status field, then steps off the record. In this case, the service request record is being written in the context of the Contact Business Object, and the run-time event does not fire.

Using a Run-Time Event Within a M:1 Relationship

When using a runtime-event to trigger a workflow based on a modification made to a record that contains a M:1 relationship with a parent, you must trigger the workflow based on the child's ROW_ID.

For example, assume you need to trigger a workflow when a field in a service request's activity record is updated. Since a given service request can have one or many activities, you must trigger the workflow based on the ROW_ID for the activity, not the service request. If you base the trigger on the service request's ROW_ID, then a change made in the service request's form applet triggers the workflow, but a change made in the activities list applet does not.

To examine this relationship, in the Siebel client navigate to Service Request > Service Request List, create a new service request, drill down by clicking in the SR# field, then use the Activities list applet to create two new activities. In this view, note that the top service request form applet displays fields for the parent service request while the bottom activities list applet displays multiple activities for the parent.

To create a test workflow process triggered within a M:1 relationship

1 In Siebel Tools, in the Workflow Processes OBLE, create a new workflow process object definition with the following values:

Property	Value
Process Name	Update Service Request
Business Object	Service Request
Workflow Mode	Interactive Flow

The workflow process is based on the Service Request business object. The runtime-event used in this workflow occurs on the start step and is based on the Action business component, which is a child of the Service Request. Note that the wait step, used here for testing purposes, requires an Interactive Flow. For more information, see "About the Wait Step" on page 108. To view an example, see "Creating a New Workflow Process Object Definition" on page 316.

2 Open the Process Designer for the workflow process you created in Step 1, then create a workflow that resembles the workflow in the following diagram:

For more information, see "About Workflow Process Steps and Connectors" on page 87, and "Diagramming a Workflow Process" on page 61.

3 Click the Id Triggered connector, then use the Properties window to define values described in the following table:

Property	Value
Event Object Type	BusComp
Event	WriteRecord
Event Object	Action

4 Click anywhere on the canvas background.

5 In the MVPW, right-click then add a new process property using values described in the following table:

Argument Field	Value
Name	ActionBCRowId%
In/Out	In/Out
Business Object	Service Request

For more information, see "About Process Properties" on page 74.

To capture the triggered activity ROW_ID, you define a process property, then use a wait step which can read a field from the child business component into the process property in the wait step's output argument without modifying the underlying data.

6 Click the Get Activity Id step, then click the Output Arguments tab in the MVPW.

7 Right-click, choose New Record, then define a new record using values described in the following table:

Argument Field	Value
Property Name	ActionBCRowId%
Type	Expression
Value	Id
Business Component Name	Action

For testing purposes, this step reads the triggered activity ROW_ID, which is the Id field, into the ActionBCRowId% process property. No input arguments are defined for the wait step.

8 Validate then simulate the workflow process.

For more information, see "Process of Testing a Workflow Process" on page 231.

9 Implement this technique in your production workflow.

This example demonstrates how you can trigger a workflow based on changes made to a child in a M:1 relationship. It includes steps to test the technique. In a production environment, if you no not need to capture the child's ROW_ID, you can simply define the trigger on the connector emanating from the start step, as described in Step 3.

Using a Run-Time Event With the Updated By Field

If a step has a runtime event has its processing mode defined to run locally to either start or resume a Workflow Process, then Updated By is the user who is currently logged into the application.

Run-Time Events That Cannot Be Used to Invoke a Workflow Process

Since a workflow process can be invoked only within the context of a business component record, if there is no business component record context it is not possible to invoke a workflow, and attempting to invoke a workflow using the BusComp Query event will fail. Therefore, any runtime event that has the possibility of returning no results cannot be used to invoke a workflow.

Repeat Usage of a Runtime Event

When defining a run-time event within a workflow process, the run-time event must not be repeated. You cannot use the same event more than once within a given workflow process.

About the User Event

While a run-time event acts on a workflow process from within the application object manager, a user event is internal to Siebel workflow. A user event initiates and resumes a long-running workflow process in the Workflow Process Manager (WFProcMgr) server component.

NOTE: A user event can be used only in a long-running workflow process. A long-running workflow always resumes on WFProcMgr. The behavior cannot be modified to have a long-running workflow resume on a custom Workflow server component.

While a run-time event can be used in a workflow that runs within a single user session, a user event is for use in a long-running workflow that spans multiple users. A user event can be used to trigger a workflow process when the user event is attached to a Start step, or to resume a waiting workflow instance when the user event is attached to a step that can receive an input argument. A user event can also bring data into a workflow instance which can contain user data. This data comes in the form of the user event's payload.

Generating a User Event with the Workflow User Event Service Business Service

A user event requires use of the Workflow User Event Service business service to communicate with the Workflow Process Manager. A User event is a Workflow internal event used to resume a long-running workflow process from the Workflow Process Manager. To create a user event, you invoke the Workflow User Event Service business service, specifying the payload and the correlator.

NOTE: A long-running workflow process must use only user events, not run-time events.

The Workflow User Event Service business service is a standard Siebel business service that can be used everywhere a Siebel business service can be used. You invoke the Workflow User Event Service business service by configuring a business service step that calls it.

The User Event business service can be invoked by supported techniques, such as scripting, COM interfaces, and Java interfaces. This is the recommended way to externally communicate with a Siebel workflow.

A common case is when a 7.0 workflow, an interactive workflow, or a service workflow initiates a user event. A business service step calls the Workflow User Event Service business service to communicate to a long-running workflow running in the background.

NOTE: While any type of workflow process or business service can generate a user event, it is recommended that only a long-running workflow process be configured to receive a user event.

For more information, see "About the Workflow User Event Service Business Service" on page 381.

Invoking the Workflow User Event Service Business Service

This topic describes how to invoke the Workflow User Event Service business service for generating a user event.

To invoke the Workflow User Event Service Business Service for generating a user event

1 Add a Business Service step to a workflow process.

2 Click the new business service then use the Properties window to specify values described in the following table:

Property	Value
Business Service Name	Workflow User Event Service
Business Service Method	GenerateEvent

3 In the MVPW, create a new input argument for the step.

For more information, see "About Process Properties" on page 74.

4 In the Input Argument argument field, choose Payload from the picklist then define the other argument fields, as appropriate.

5 Repeat Step 3 to create an input argument by choosing Correlator Value from the picklist.

6 Repeat Step 3 to create an input argument by choosing User Event Name from the picklist.

Configuring a Long-Running Workflow Process to Wait for a User Event

When using a user event in your workflow process, keep in mind that only a long-running workflow can wait for a user event. Other types of workflows cannot *wait* for a user event, though they can *generate* a user event. A long-running workflow must be configured with *a user* event only, not a run-time event.

To configure a long-running workflow process to wait for a user event

1 Open the design canvas fort the workflow process you must configure to wait for a user event.

2 In the MVPW, establish one of the workflow's process properties as the correlator by setting the property's Correlator Flag to TRUE.

For more information, see "About Process Properties" on page 74.

3 On the branch of the step that handles the event, such as a Start step or a Wait step, use the Properties window to define values described in the following table:

Property	Value
Type	Condition **NOTE:** If Type is set to Default rather than to Condition, the user event is never triggered.
User Event Name	(The name of the workflow you defined for the Value property in Step 6 on page 156.)
User Event Timeout	(The timeout period for the user event.) User Event Timeout works in a similar way as the timeout setting for a run-time event. A workflows is resumed after the timeout period if no user event is received during the timeout period.
User Event Storage	(The name of the process property that holds the payload passed in from the user event.)

NOTE: User events are not queued. If no recipient is waiting to accept the user event with the specified correlator, the event is discarded.

About Handling Errors

An error, which represents a deviation from the normal processing flow, can be a system error or a user-defined error. You can use error handling to notify the user of an error and terminate the workflow process instance.

This topic includes the following topics:

- Using an Error Exception to Handle Errors on page 157
- Using the Stop Step to Handle Errors on page 162
- Using an Error Workflow Process to Handle Errors on page 162
- Defining Recovery for a Failed Workflow Process on page 164

For information about using a process property when handling errors, see "Passing a Process Property to an Error Workflow Process" on page 82.

Using an Error Exception to Handle Errors

An *error exception* is a type of branch designed for handling system and user-defined errors. An example of a system generated error is a failure when sending an email notification. An example of a user-defined error is attempting to submit an order that is not complete.

You can use an exception branch to programmatically handle errors and change the flow depending on when an error is encountered. This technique provides a granular approach to handling exceptions at each step.

In the Process Designer an Error Exception appears as a red connector between two steps. When you click an exception connector, the Properties window displays the connector's WF Step Branch properties.

When an error occurs, the error code and error message are automatically populated in the Error Code and Error Message process properties. An exception allows you to set up a condition using values in these properties.

Similar to other cases where conditional logic is used in a workflow process, an exception on a step is evaluated *after* the step has finished. If you need to evaluate an exception before executing a step, you must attach the exception to the prior step in the workflow process.

Example of Error Exception Handling

This topic gives one example of defining error exception handling. You might use this feature differently, depending on your business model.

In the example displayed in Figure 16, when the Get Organization ID step is unable to get data, the workflow continues to the Lookup Sender by Org step. If Lookup Sender by Org fails, the workflow takes the red Exception branch and sends an email using the Send Lookup Error Email step.

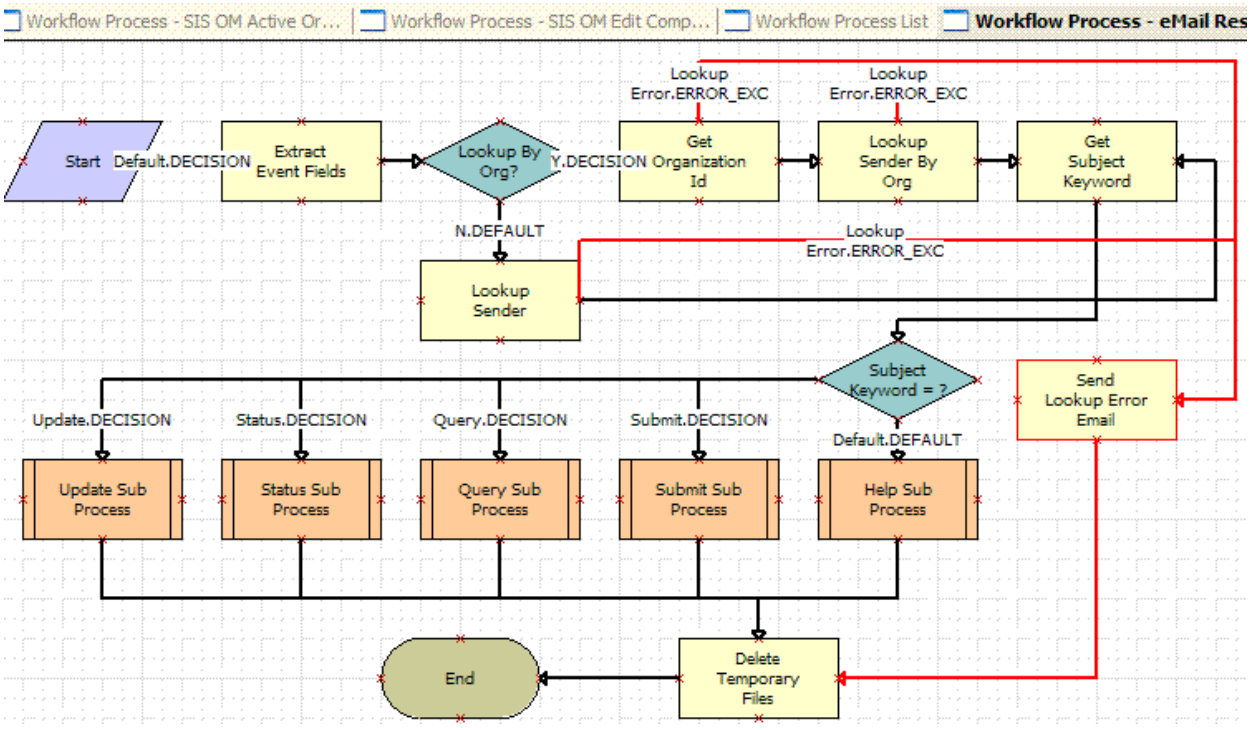


Figure 16. Example of a Workflow That Uses Exception Branches to Programmatically Handle Exceptions

Defining an Error Exception

An Error Exception is defined in the Process Designer.

To define an Error Exception

1. In Siebel Tools, in the Workflow Processes OBLE, right-click the workflow process for which you must define exception handling, then choose Edit Workflow Process.
2. In the Process Designer, drag then drop an Error Exception connector from the palette to the canvas, attaching one end of the connector to an existing step icon for which you need to trap errors.

   Some example step types you might need to trap for error conditions include the Business Service step type and the Siebel Operation step type.
3. Make sure the end of the connector is attached to the step.
4. Drag then drop a Stop step from the palette to the canvas.
5. Attach the unconnected end of the Error Exception connector to the Stop step.
6. With the exception arrow chosen in the canvas, enter a value in the Name property in the Properties window.
7. In the Type property, choose Error Exception or User Defined Exception.
8. In the canvas, double-click the exception icon to access the Compose Condition Criteria dialog box.
9. Define conditions that apply for the exception.

   For more information, see “Defining Conditional Logic on a Branch Connector” on page 114.

Defining an Error Exception to Handle an Update Conflict

You can define an error exception to handle an update conflict that occurs when multiple attempts are made to write to the same record at the same time.

When the Workflow Monitor Agent (WMA) is used to invoke a workflow process, which in turn updates a record, the WMA can fail if a workflow process attempts to update a record that is updated by another user or task since it was initially retrieved by the workflow process. In this case, an error message, such as *The selected record has been modified by another user since it was retrieved*, is displayed. You can prevent the WMA task from failing by defining an error exception to handle an update conflict that occurs while the workflow process is running.

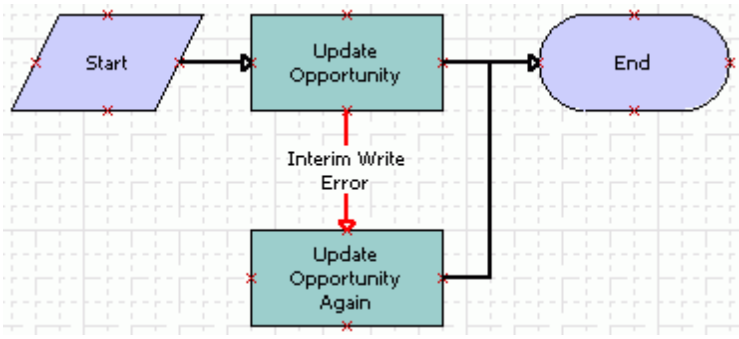
To create a test workflow process that uses an Error Exception to handle an update conflict

1 In Siebel Tools, create a new workflow process object definition using values described in the following table:

Property	Value
Process Name	Error Exception Example
Workflow Mode	Service Flow
Business Object	Opportunity

To view an example, see "Creating a New Workflow Process Object Definition" on page 316.

2 Add steps and connectors until your workflow process resembles the workflow illustrated in the following diagram:



For more information, see "About Workflow Process Steps and Connectors" on page 87, and "Diagramming a Workflow Process" on page 61.

3 Click the Update Opportunity business service step, then use the Properties window to define properties described in the following table:

Property	Value
Business Service Name	ABC Update Opty
Business Service Method	Update Opty

4 With the Update Opportunity business service step still chosen, define an input argument in the MVPW using values described in the following table:

Argument Field	Value
Input Argument	Opportunity Id
Type	Process Property
Property Name	Object Id
Property Data Type	String

For more information, see "About Process Properties" on page 74.

5 Define conditional logic on the error exception named Interim Write Error using values described in the following table:

Property	Value
Compare to	Process Property
Operation	One Must Match (Ignore Case)
Object	Error Code
Values	0x8137 -- 0x6f74

Note that 0x8137 -- 0x6f74 is the error code returned with the *The selected record has been modified by another user since it was retrieved* error message.

For more information, see "Defining Conditional Logic on a Branch Connector" on page 114.

6 Define the Update Opportunity Again business service, using the same values you used in Step 3 and Step 4. For the Name property, use Update Opportunity Again.

7 Validate then simulate the workflow process.

For more information, see "Process of Testing a Workflow Process" on page 231.

8 Implement this technique in your production workflow.

How This Example Works

The Update Opportunity Again step provides a way to write to the opportunity record again in cases where the first attempt to update an opportunity fails due to the update conflict error. Note that this example uses a Business Service step that updates opportunity records. The same technique can be used with other step types that update a record, such as a Siebel Operation or a Sub Process step, and with other types of records, such as accounts or contacts.

Using the Stop Step to Handle Errors

A Stop step is used to show a particular error message while processing. Example usage includes real-time processing for credit card authorization or for order validation. This type of exception handler provides customizable error messages including expressions.

For more information, see "About the Stop Step" on page 109.

Using an Error Workflow Process to Handle Errors

You can use an error workflow process to handle errors. An *error workflow process* is a standard workflow process that becomes an error process when you associate it to another, main workflow process. The association is created by specifying the Error Process Name property in the Workflow Processes OBLE for the main workflow. When an error occurs in the main workflow, the exception branch calls the error workflow process.

The process defined in the Error Process Name property is called when the main process reaches an error state. Execution of the main process stops, the main workflow process passes system defined process properties to the error workflow process, then the error process commences.

Restrictions when using an error workflow process include:

- An error workflow process does not return values back to the originating workflow that failed. If you require that the state of the main workflow be changed by the error handling actions, use an exception branch.
- An error process can be associated with a workflow process that is a subprocess. However, an error process cannot itself contain a subprocess.

Benefits of Using an Error Workflow Process

An error workflow process that is used when you need to handle errors across multiple steps in a workflow process or across multiple workflows is called a *universal exception handler*. This type of exception handling can be used to reduce clutter on the workflow flowchart, and to reduce the need to create a large number of error workflow processes by reusing a single or a small set of error workflow processes.

How Errors Are Handled

This topic describes how error handling varies for a workflow process and how errors are handled for a sub process.

How Errors Are Handled for a Workflow Process

If an error is encountered in a workflow process that does not have an error process defined in the Error Process Name property, the workflow process remains in the In Error state. The original error code is returned to the caller of the process.

If the main workflow process encounters an error and there is an error workflow process defined for the main workflow process in the Error Process Name property, the error workflow process finishes with one of the outcomes described in Table 29.

Table 29. Description of Error Handling When an Error Process is Defined for a Workflow Process

Condition	Error Process State	Error Code	Result
The error process handles the error successfully.	Completed	No error code is returned to the caller of the workflow process.	Error handling is considered successful if the error process arrives at an End step. The error workflow process terminates immediately with a Completed state.
The error process tries to handle the error, but fails with a different error.	Error	A new error code, which you have defined in the Stop step, is returned to the caller of the error workflow process.	Error handling is considered failed if the error process arrives at a Stop step. The workflow process remains in the In Error state.
The error process cannot handle the error.	Error	The original error code is returned to the caller of the workflow process.	If no Start conditions are satisfied, the error process ends immediately. The workflow process remains in the In Error state.

How Errors Are Handled for a Sub Process

If a sub process encounters an error and there is an error workflow process defined for the sub process in the Error Process Name property, the error workflow process finishes with one of the outcomes described in Table 30.

Table 30. Description of Error Handling When an Error Process is Defined for a Sub Process

Condition	Error Process State	Error Code	Result
The error process handles the error successfully.	Completed	(This cell is intentionally empty.)	Error handling is considered successful if the error process arrives at an End step. The subprocess also terminates with a Completed state, then control returns to the main process instance. The main process instance continues to execute from the next step.
The error process tries to handle the error, but fails with a different error.	In Error	The new error code, which you have defined in the Stop step, is returned to the subprocess.	Error handling is considered failed if the error process arrives at a Stop step. Both the error workflow process and the main process terminate with the In Error state.

Defining Recovery for a Failed Workflow Process

For information about how to define recovery for a failed workflow process, see “About the Workflow Recovery Manager” on page 268.

About Batch Processing

A workflow process can be run in batch by running the Workflow Process Batch Manager server component (WfProcBatchMgr). Executing a process in batch allows you to execute the actions in a workflow process for multiple records. When using the Workflow Process Batch Manager, be sure to specify the business object with which the workflow process is associated. You must also use a search specification to locate the business component records that are referenced or manipulated.

NOTE: It is recommended that only a service workflow process or a 7.0 workflow process be run in batch.

Table 31 describes parameters for the Workflow Process Batch Manager.

Table 31. Description of Parameters of the Workflow Process Batch Manager

Display Name	Description
Workflow Process Name	Required. Name of the workflow process definition to execute.
Search Specification	Search specification that identifies the work items to process. The Search Specification parameter is a generic parameter shared by Workflow Management server components. However, this parameter is only used by the WfProcBatchMgr component. A Search Specification parameter specified for a component other than WfProcBatchMgr is not used.

Scheduling a Batch Workflow

You can set up a batch workflow to run a workflow process at specified intervals. For example, 7 A.M. every Monday. You do this by setting up a component request to schedule a batch workflow.

To set up a component request to schedule a batch workflow

1. Navigate to Administration-Server Configuration > Enterprises > Component Definitions.
2. In the Component Request form, click New.
3. In the Component/Job field, click the selection button. The Component/Jobs dialog box appears.
4. Choose Workflow Process Batch Manager.
5. In the Component Request Parameters form applet, click New.
6. In the Name field, click the selection button, then choose Workflow Process Name from the dialog box.
7. In the Value field, type in the name of the workflow process to execute.
8. Click New to add another parameter.
9. In the Name field, click the selection button.
10. Choose Search Specification.

In the Value field, provide a search specification.

For more information on running a workflow process in batch, see *Siebel System Administration Guide*.

About the Repeating Component Request

You can use the Repeating Component Request feature in conjunction with the workflow process to schedule the workflow task to be executed at a predetermined time. To view an example workflow process that benefits from a repeating component request, see "Example Workflow Process That Traverses a Record Set to Close Obsolete Service Requests" on page 324. For more information, see *Siebel Server Administration Guide*.

Using the Search Specification Parameter

When running a workflow process in batch, you can specify a search specification to limit the number of records evaluated. If you do not specify a search specification, Workflow Process Batch Manager executes the workflow process for each record of a particular type in the Siebel application. For example, if there are 100 SRs, the Workflow Process Batch Manager executes the workflow process 100 times, once against each SR.

The Search Specification parameter on the Workflow Process Batch Manager is used to execute the search specification on the primary business component of the business object specified in the Business Object property of the workflow process. For each fetched record, the Workflow Process Batch Manager invokes the workflow process and sets the Object Id process property as the current active row.

Using Workflow Process Batch Manager When Primary and Non-Primary Business Components are Present

When running Workflow Process Batch Manager, if the Link Specification property is set to TRUE on a field on the primary business component, the number of records returned can be more than expected. This condition can affect performance. This is due to the fact that the field's value is passed as a default value to a field in the nonprimary business component through a link.

This condition occurs when the primary business component has a link relationship with one or more nonprimary business components, as established through a Link object definition on the current business object. Therefore, it is important to be careful when the business component is accessed by querying records in a custom Business Service or when running Workflow Process Batch Manager.

Comparing Workflow Process Batch Manager Against Workflow Process Manager

Use the Workflow Process Batch Manager server component when you need to run the workflow process for all records of a particular business component. WfProcBatchMgr considers the primary business component associated with the business object property on the workflow process. WfProcBatchMgr then runs the workflow process on all records within that primary business component.

Considerations When Using a Custom Business Service

A situation can change when the workflow process involves a custom business service. For example, assume the custom business service contains a loop that processes all records and executes the business service code on each record returned from the code itself. Table 32 compares how the two process managers compare in this situation.

Table 32. Comparison of Using WfProcMgr Against WfProcBatchMgr When Using A Custom Business Service That Contains a Loop

If WfProcMgr Is Used	If WfProcBatchMgr Is Used
The business service, and the loop, handles all the records at one time. This is the recommended way to execute this business service and workflow process.	Since the workflow process' business service already contains a loop, then using WfProcBatchMgr causes the business service loop to execute for every record in the primary business component. This leads to undesirable duplicate and repeated execution.

About Global Implementation

This topic describes global implementation of a workflow process. It includes the following topics:

For more information, see *Siebel Global Deployment Guide*.

Configuring a Workflow in a Multilingual Environment

To create a workflow in a language other than the base language, be sure your database is capable of handling multilingual lists of values (MLOV) for the non-base language type. For example, if you are modifying a workflow using language that is FRA and the base language is ENU, be sure List of Values entries exist for the FRA language type.

To make sure list of values entries exist for the appropriate language type

1. In the Siebel client, navigate to Administration-Data > List of Values.
2. Run the following query: `Type = "WF_*"`.
3. Make sure the database is capable of handling multilingual lists of values by running the MLOV upgrade utility. For information on running the MLOV upgrade utility, see *Configuring Siebel Business Applications*.
4. In the List of Values applet, make sure the Active flag is set.

Referencing Literal Values in a Dynamic Picklist

Since literal values are language dependent, a workflow process cannot reference literal values in a dynamic picklist in a global implementation. A workflow referencing literal values become language dependent as well and cannot be run in a global implementation.

Literal values in a dynamic picklist potentially contains content created by the end-user population at run-time. Since these values are content based rather than metadata or LOV based, the makeup of this content is variable and, therefore, cannot be predictably interpreted by the workflow process.

For more information, see "Deploying a Workflow Process in a Multilingual Environment" on page 240 and *Siebel Global Deployment Guide*.

About Defining Expressions for a Workflow Running in a Multilingual Environment

Workflows use the Display value to fetch records from tables. In a multilingual deployment, data in the tables is stored in language independent code (LIC). To run workflows in a multilingual environment, use the LookupValue function to fetch the LIC based on the Display value.

Decision Point Example

You have a Decision Point that compares Account Status to Active. The Account Status field is bounded by the Account Status picklist. You can set the Compare To property to Expression, and set the Expression property to:

```
[Account Status] = LookupValue ("ACCOUNT_STATUS", "Active")
```

Business Service Example

You have a Business Service step that calls the Outbound Communications Manager to send emails to *Expense Approver*. The Recipient Group argument is bounded by the Comm Recipient Group picklist. You can set the Type property to Expression, and set the Value property to:

```
LookupValue ("COMM_RECIP_SRC", "Comm Employee")
```

For more information on globalization, see *Siebel Global Deployment Guide*. For more information on configuring Siebel Workflow to use MLOV capable fields, see *Configuring Siebel Business Applications*.

About the Wait Step and Global Time Calculations

Types of wait steps include:

- An *absolute* wait is a wait period governed solely by the duration specified. For example, an absolute wait set for 30 minutes waits 30 minutes from the time the wait is initiated by a Wait step.
- A *service calendar* wait, on the other hand, is not absolute. For example, a service calendar wait can be set to begin at 6 P.M., but if the service hours for the organization are 9 A.M. to 6 P.M., the wait does not initiate until 9 the next morning. So it runs from 9 to 9:30 instead of 6 to 6:30.

Wait Steps and Time Zone Settings

When a workflow process is executing as a server task, you must shut down then restart the Workflow Process Manager after making changes to the Time Zone user preference for the SADMIN user. The changes only take effect after restarting the Workflow Process Manager. This is important if you are implementing UTC, as it can be necessary for you to set the Time Zone user preference.

Relationships between the type of wait step and time zone settings include:

- An absolute wait is not affected by time zone settings, including server and user time zone preferences. Use UTC with the database server. For more information, see *Siebel Global Deployment Guide*.
- A service calendar wait step requires a time zone for delay computations. In this case, the current user's time zone is used.

About the Local Code Parameter and Data Format

The Workflow Process Manager server component (WfProcMgr), similar to other server components, has a parameter called Local Code which defines formatting of data, such as dates, times, numbers, and currency.

When a workflow process runs within the WfProcMgr, WfProcMgr respects the Local Code setting and formats the data appropriately. If the workflow process communicates with an external application or writes data to file, then date, time, number, and currency data is passed according the format defined in Local Code.

9 About Workflow Policies

This chapter provides information about workflow policies. It includes the following topics:

For more information, see Chapter 12, “Administering Workflow Policies.”, and Chapter 14, “Reference Materials for Siebel Workflow.”

About the Workflow Policies Module

This topic includes the following topics:

The Workflow Policies module allows you to define a policy that can invoke a workflow process. A policy consists of one or more policy conditions. When the policy conditions are met, the policy action is executed.

NOTE: The name Workflow Policies replaces the name Workflow Manager, which was used to refer to the Siebel Business Process automation tool in earlier releases.

Overview of Workflow Policy Objects

Workflow policy objects provide the context in which Workflow Policies operate. The workflow policy object, through the workflow’s policy components, defines the set of tables and columns that can be monitored by a policy and how each table in the workflow policy object relates to the other tables. This collection of columns and the relationships between the tables of the workflow policy object represent the entity within Siebel Tools that you must monitor.

Siebel Tools provides visibility to many predefined workflow policy objects for common business needs, such as Opportunity, Service Request, and Contact. You can modify some predefined workflow policy objects through administrative screens in the Siebel client. You can also use Tools to create custom workflow policy objects to meet your specific business requirements.

Relationships Between Workflow Policy Objects

The relationships between Workflow Policy Objects, Workflow Policy Columns, and Workflow Policy Programs are illustrated in Figure 17.

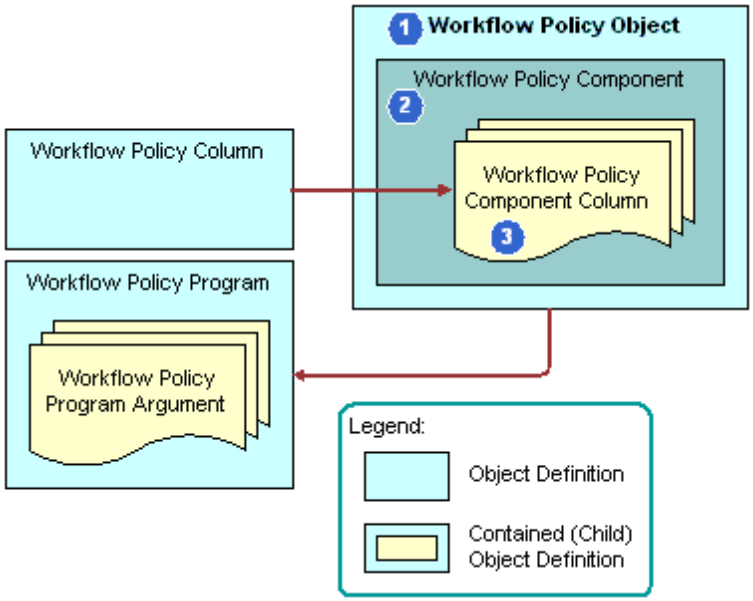


Figure 17. Relationships of Workflow Policy Objects, Programs, and Columns

Relationships between workflow policy objects include:

1 **Workflow Policy Object**. A *workflow policy object* is a collection of workflow policy components. A workflow policy object is defined by the workflow policy object's parent-child relationship to workflow policy components and workflow policy component columns.

2 **Workflow Policy Component**. A *workflow policy component* is a logical mapping of a database table that defines the Siebel database tables that you can monitor. Workflow policy components define the relationships between the primary workflow policy component and other policy components of a workflow policy object. Except for the primary workflow policy component, each workflow policy component defines a relationship to another workflow policy component. This relationship is defined by specifying a source policy column and a target policy column. The source and target columns on a workflow policy component identify foreign key relationships between the tables.

   A *primary workflow policy component* is a workflow policy component that other workflow policy components are directly or indirectly related to. From these workflow policy components, the workflow policy columns that are available for monitoring in the workflow policy can be defined. Each workflow policy object has one and only one primary workflow policy component. The other workflow policy components of a workflow policy object are related to the primary workflow policy component directly or indirectly.

3 **Workflow Policy Component Column**. A *workflow policy component column* defines the columns in the Siebel database table that you can monitor. You expose these columns for monitoring when you define workflow policy conditions for a workflow policy.

To define a workflow policy object and the workflow's components, familiarize yourself with the Siebel Data Model. For more information, see *Siebel Data Model Reference*. For information about tables and how tables are related, see *Siebel Data Model Reference*.

Visualizing the Hierarchy Between Workflow Policy Objects

Each workflow policy component can expose a number of workflow policy component columns. In the Object Explorer, a Workflow Policy Component Column is the child object of a Workflow Policy Component, which is itself a child object of a Workflow Policy Object. To view this hierarchy, see "About the Workflow Policy Object Hierarchy in Siebel Tools" on page 179.

Example Entity Relationship Diagram for a Workflow Policy

Figure 18 displays the entity relationship diagram for four Service Request Workflow Policy components. The diagram illustrates each of the components, their relationship to one another, and which columns are of interest. Service Request is the primary workflow policy component, and the other three components are joined directly or indirectly to it.

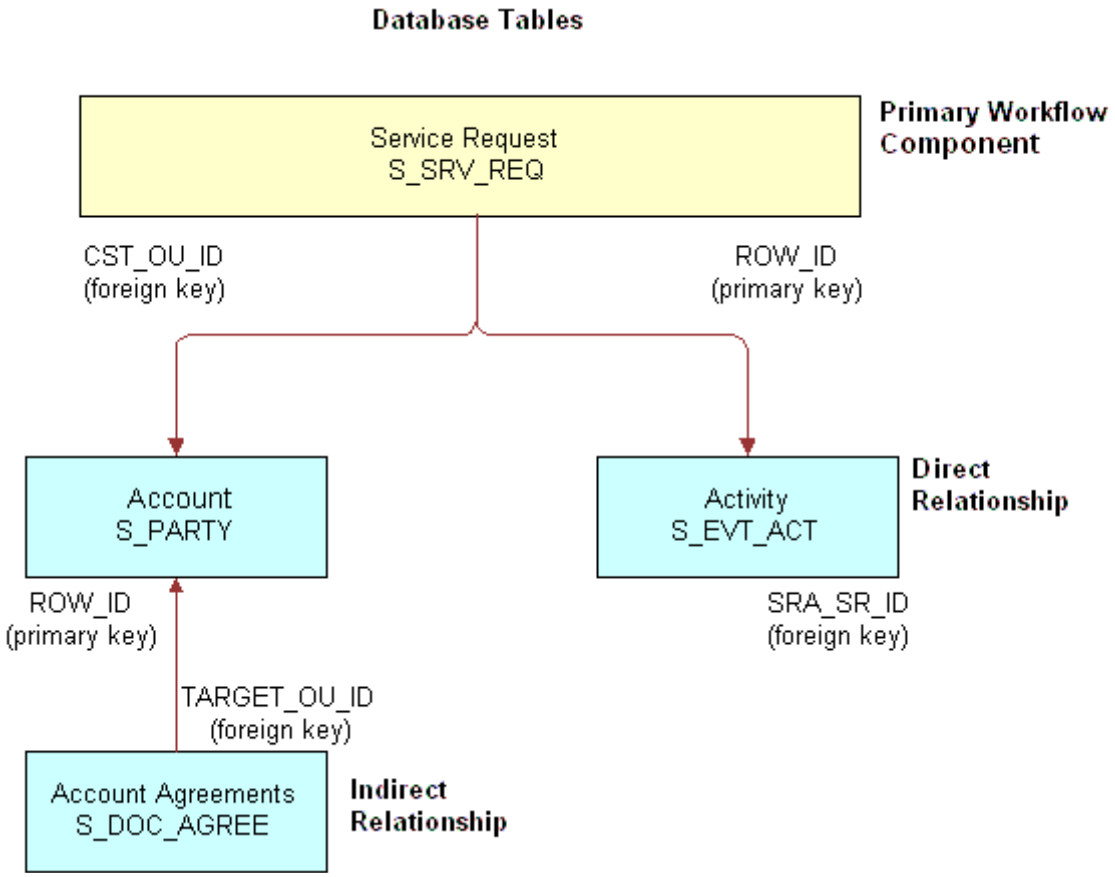


Figure 18. Example Entity Relationship Diagram for a Workflow Policy

About Workflow Policies and Monitoring Tables

Workflow Policies can monitor only Siebel tables. You cannot use a Workflow policy to monitor database tables that are external to Siebel.

CAUTION: Do not monitor the S_DOCK_TXN_LOG table or Enterprise Integration Manager (EIM) table columns. An EIM table is prefixed with EIM_, or ends with _IF. Most tables can be monitored except S_DOCK_TXN_LOG and EIM tables.

Structure of an Example Workflow Policy

The basic underlying construct of a workflow policy is the *rule*. The structure of a rule is: if the conditions are true, then an action occurs. The rule contains a policy condition and a policy action. When the conditions of the workflow policy are met, an action is triggered. Figure 19 illustrates the parts of an example workflow policy.

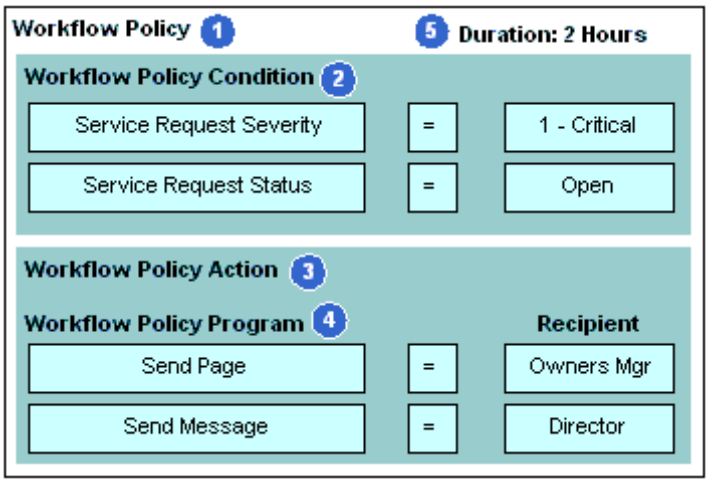


Figure 19. Parts of an Example Workflow Policy

Parts that make up a typical workflow policy include:

1. **Workflow Policy**. A workflow policy is a construct composed of conditions and actions. A workflow policy, based on the Workflow Policies rule structure, represents the rules the database monitors.
2. **Workflow Policy Condition**. A workflow policy condition is a trigger: a circumstance or situation that causes something to happen.
3. **Workflow Policy Action**. A workflow policy action is an action invoked by a workflow policy condition being fulfilled.
4. **Workflow Policy Program**. A workflow policy program is a predefined program that provides arguments for the workflow policy action.
5. **Duration**. A *duration* is the period of time for which workflow policy conditions exist for the conditions of the policy to be met. For more information, see "Planning the Workflow Policy and Conditions" on page 182.

NOTE: The functionality available with a Workflow Policy can be supported by using a Workflow Process. It is recommended that a workflow policy be used to define conditions to invoke a workflow process. Use a Workflow Process to define actions.

Workflow Policy Condition

A *workflow policy condition* expresses an object/attribute relationship to a value. For example, a workflow policy condition can target data, such as Service Request Severity. The workflow policy condition compares that data to a value, such as 1-Critical. The combination of the data element, Service Request Severity, a comparison operation, such as equal (=), and the value, such as 1-Critical, define the policy condition.

The fact that a Service Request Severity is 1-Critical can be an issue only if the policy condition remains valid for some extended period of time, such as two hours. In this example, if the policy condition remains valid, a duration can be set for two hours on the workflow policy. The duration becomes part of the policy condition. The policy actions are not executed until the policy conditions are met for the specified duration.

A workflow policy action can also occur when a time duration is not set. For example, email is automatically sent to a sales manager each time a sales representative quotes a discount rate exceeding 25 percent on revenue less than $100,000.

A policy frequently has more than one condition. The conditions of the policy must be met before an action can occur. A service request with a severity of 1-High and a duration of two hours might be important only if another comparison is also valid, such as the Service Request Status is Open. The policy condition becomes the combination of these two comparison operations:

```
SR Severity = 1-Critical AND SR Status = Open
```

Workflow Policies only support AND linkages between policy conditions, not OR linkages. If you must monitor the SR Severity to be 1-Critical or 2-High and the SR Status is Open, you can use the IN operand to evaluate the OR of the SR Severity Condition.

```
SR Severity IN ('1-Critical', '2-High') AND SR Status = Open
```

Alternatively, OR linkages can be simulated by creating multiple policies for each key policy condition. The combination of workflow policies act similar to an OR linkage. For more information, see "About Logical Operators in the Conditions Applet" on page 207.

Workflow Policy Action

A *workflow policy action* is an event you need to occur when the conditions of your workflow policy are met. The parts of a workflow policy include:

- An *action,* which is a type of request, such as *Send an Urgent Page*.
- The *action parameters,* which are the arguments, such as the name of the recipient of the page and the alphanumeric text transmitted with the page.

You can specify several actions for one workflow policy, such as sending a page to one person and an email to another. You can reuse an action in multiple workflow policies. For a discussion of actions and action parameters, see "About Defining Workflow Policy Objects" on page 213.

NOTE: In most cases, use a workflow policy action to run a workflow process.

You can pass a constant from a Workflow Policy Action into a Workflow Process. For more information, see "Passing a Constant from a Workflow Policy Action into a Workflow Process" on page 417.

Workflow Policy Program

A *workflow policy program* is a predefined program that provides arguments for the workflow policy action. A workflow policy action is based on the underlying program in Siebel Tools and inherits the program's arguments. A workflow policy program defines the particular action that takes place when the conditions of a workflow policy are met. Most functionality included in a workflow policy program can be executed using a workflow process.

You can use programs in multiple action definitions and you can use action definitions in multiple workflow policies.

Types of workflow policy programs include:

- **Send Message.** Sends an email to one or more recipients.
- **Send Page.** Sends a page to one or more recipients.
- **Send Broadcast Message.** Inserts a message broadcast for one or more recipients.
- **DB Operation.** Inserts or updates the data records of a Siebel database table for chosen workflow policy components.
- **External Program.** Allows you to run an executable.
- **Assignment Request.** For internal use only.
- **Generic Request Server.** Submits a server request to a designated server component.

For reference information, see Properties of the Workflow Policy Program on page 449 and Properties of the Workflow Policy Program Argument on page 450.

Predefined Workflow Policy Programs

Workflow policies use workflow policy actions based on workflow policy programs that are predefined in Siebel Tools. Some examples include:

- Send Broadcast Message
- Database Operation

For more information, see "Predefined Workflow Policy Programs" on page 394.

Functional Description of a Typical Workflow Policy

A functional processing of a typical workflow policy is illustrated in Figure 20.

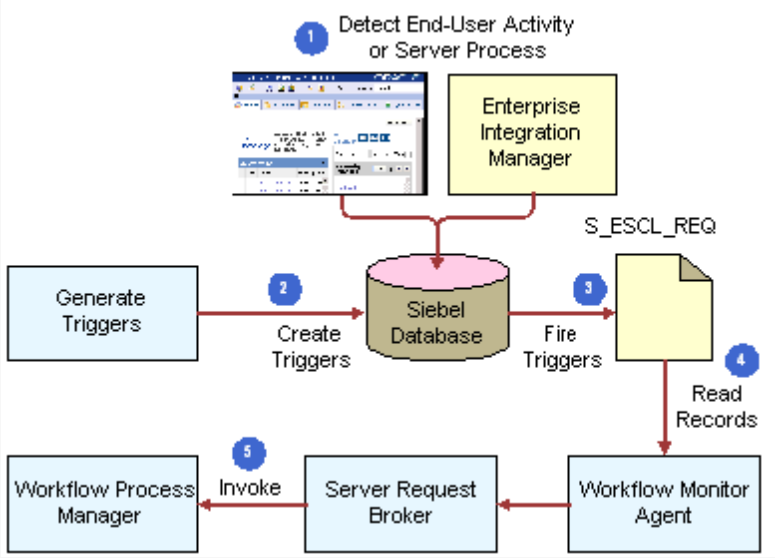


Figure 20. Functional Processing of a Typical Workflow Policy

Processing for a typical workflow policy includes:

1. **Detect End-User Activity or Server Process**. An end-user activity or server process occurs.
2. **Create Triggers**. Workflow Policies enforce policies at the data layer through database triggers. When a particular policy's conditions are met, underlying database triggers capture the database event into a Workflow Policy Manager.
3. **Fire Triggers**. The S_ESCL_REQ table is queued.
4. **Read Records**. The Workflow Monitor Agent, which is a Workflow Policy Manager component, reads records in the S_ESCL_REQ table then processes requests by taking the actions defined.
5. **Invoke Workflow Process Manager**. In some cases, an action defined can invoke the Workflow Process Manager.

Workflow Policy Manager provides scalability by using an additional component called Workflow Action Agent. Workflow Action Agent can be executed on a different application server within the Siebel Enterprise. For more information, see "Executing a Workflow Policy with Workflow Action Agent" on page 288.

For more information about Generate Triggers, Workflow Process Manager and Workflow Monitor Agent, see "Workflow Run-Time Architecture" on page 33.

For more information about the S_ESCL_REQ table, see "About the S_ESCL_REQ Table" on page 281.

About the Workflow Policy Object Hierarchy in Siebel Tools

Using Siebel Tools, you can modify existing workflow policy object definitions and define new workflow policy object definitions to meet your business needs.

Figure 21 highlights the object types you can modify and the hierarchies that exist between them, as they appear in Tools' Object Explorer.

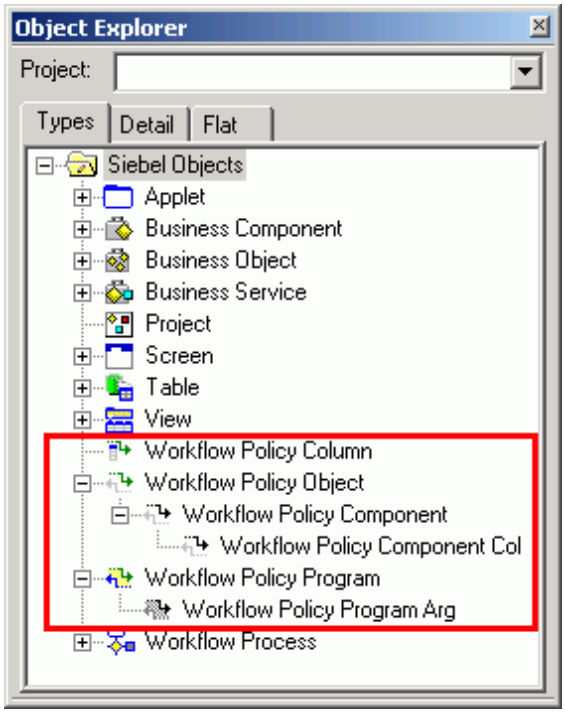


Figure 21. Workflow Policy Object Types in the OE

Considerations to weigh include:

- Workflow policy objects are not displayed in the Object Explorer by default. To display them, from the Tools application-level menu, chose View > Options then click the Object Explorer tab. Scroll to the bottom of the Object Explorer Hierarchy window. Make sure there is a check mark next to each workflow policy object type you need to expose in the OE.
- When you use Siebel Tools to modify or create workflow policy objects on your local system, the changes are not available on the server until they are applied to the server.

For more information about hierarchies in the Object Explorer, and for background information about object types, object definitions, and properties, see "About the Workflow Process Object Hierarchy" on page 65.

Process of Planning a Workflow Policy

This topic provides information about planning a workflow policy. It includes the following topics:

To plan a workflow policy, perform the following tasks:

1. Planning a Workflow Policy Group on page 180
2. Planning a Workflow Policy on page 181
3. Planning Objects to Monitor with a Workflow Policy on page 182
4. Planning the Workflow Policy and Conditions on page 182
5. Planning the Workflow Policy Action on page 183
6. Examining Predefined Workflow Policies and Policy Programs on page 183

Examples in this topic ground the concepts presented. For more information, see "Examples of Planning a Workflow Policy" on page 184.

Before planning a workflow policy, you should determine the automation solution that most closely meets your business automation requirements. For more information, see "Identifying an Automation Solution" on page 43.

Planning a Workflow Policy Group

This task is a step in "Process of Planning a Workflow Policy" on page 180.

Workflow policies are organized into groups. A *workflow policy group* is a collection of workflow policies that provides a means of identifying policies that have similar system requirements, thus facilitating load balancing on the servers and providing scalability. Workflow policy groups allow you to manage and optimize Workflow Agent process performance by grouping similar policies to run under one Workflow Agent process.

Before you create a workflow policy, you must create a workflow policy group. Each Workflow Policy Agent is assigned one workflow policy group. If you are going to run only one Workflow Policy Monitor Agent and one Workflow Policy Action Agent, assign your policies to one policy group.

Reasons to use multiple Workflow Policy Agents include:

- To shorten the lag time between when the policy event is triggered and when Workflow Policies notices the event.
- To spread the workload across multiple application servers.
- To adjust the polling interval so that polling for noncritical policies does not prevent efficient processing of critical policies.

Policies with similar time intervals are generally grouped together. By creating groups of policies with similar time intervals, you can assign the workflow policy group a Workflow Policy Agent with a polling rate that matches the needs of the workflow policies, thereby creating a more efficient use of your resources.

Creating workflow policy groups and using multiple Workflow Policy Agents are part of tuning your system to create the highest performance and can be finished as you monitor your system's performance.

NOTE: The maximum number of policies in a single policy group is MaxInt. The maximum number of policies in policy groups is determined by the maximum number of records in the S_ESCL_RULE table.

Planning a Workflow Policy

This task is a step in "Process of Planning a Workflow Policy" on page 180.

Many of the workflow policy objects and programs you must create for your workflow policies have been predefined by Siebel. However, you can use Siebel Tools to augment programs, create more workflow policy objects, or make more workflow policy columns available for monitoring. For more information, see "About Defining Workflow Policy Objects" on page 213 .

The planning phase is a good time to review your company's business process activities. You need to determine which work can be automated with Workflow Policies then prioritize the implementation sequence.

NOTE: It is recommended you create and implement a small group of policies at a time. After you successfully implement the group, you can proceed to another small group of policies in a systematic manner.

If the business environment necessitates changes to your plan, a workflow policy can be assigned to a different group. For more information, see "Moving a Workflow Policy to a Different Group" on page 302.

For more information on creating a workflow policy, see "Process of Creating a Workflow Policy" on page 187 .

TIP: After planning a new workflow policy, test the policy definition by creating a query based on the policy. Then you can execute the query on your current production environment. The query response can help you determine the frequency of the workflow conditions. You might find that a policy creates excessive notification or insufficient visibility. For more information, see "About Testing, Troubleshooting and Migrating a Workflow Policy" on page 311.

Planning Objects to Monitor with a Workflow Policy

This task is a step in "Process of Planning a Workflow Policy" on page 180.

You should develop a plan for what objects to monitor. For example, assume the service department needs to send an email to the service request contact whenever a service request is opened with a severity level of critical. Table 33 describes the information to record in this example.

Table 33. Example of Objects to Monitor

What to Monitor	Purpose of the Policy
Service request status Service request severity	Send an email to the service request contact when the service request is opened and the service request's severity level is critical.

Planning the Workflow Policy and Conditions

This task is a step in "Process of Planning a Workflow Policy" on page 180.

Develop a plan for the policy properties and conditions, identify the workflow policy object to be monitored in the Siebel database, and determine the monitoring interval period and duration.

Table 34 describes an example plan that details the type of information you must model for the general policy definition in terms of Workflow Policies.

Table 34. Example of Plan for the Workflow Policy

Name	Workflow Policy Object	Workflow Policy Group	Duration
Email Confirmation of SR	Service Request	Medium Frequency This value establishes the monitoring interval period.	0 Duration indicates the time that must elapse before an action is performed. Each workflow policy has one duration. If you need an action to occur after one hour, two hours, and six hours, you must create a different policy for each duration.

Table 35 describes an example plan for the workflow conditions you must plan after you determine your policy's workflow object and other properties.

Table 35. Example of Plan for the Workflow Policy Conditions

Condition Field	Operation	Value
Service Request Severity	=	1-Critical
Service Request Status	=	Open

Conditions are in the form of an expression. The Condition Field is the table column monitored in the database. Workflow policy condition values are read directly from a picklist specified at the table column level. The values called for by the condition operation you specify, such as IN/NOT IN, must be available in the column policy picklist. Otherwise, the policy cannot trigger the action.

Planning the Workflow Policy Action

This task is a step in "Process of Planning a Workflow Policy" on page 180.

A workflow policy action is executed when the conditions of the policy have been met. Workflow Policies come with a set of predefined actions and programs. You can use these or define your own actions or programs to suit your business needs.

Table 36 describes an example plan that details the type of information you must define for a workflow policy action.

Table 36. Example of Plan for the Workflow Policy Actions

Action Name	Workflow Policy Program	Workflow Policy Object	Arguments
Send SR Email to Contact	Send SR Email	Service Request	Send to Contact

Examining Predefined Workflow Policies and Policy Programs

This task is a step in "Process of Planning a Workflow Policy" on page 180.

Workflow Policies comes with a number of predefined workflow policies and policy programs that you can use to meet specific business requirements. Before creating a new workflow policy, you should examine the predefined policies to see if one already exists that meets your business requirements. This way, you can modify existing, predefined objects rather than create new ones.

For more information, see "Predefined Workflow Policy Programs" on page 394 and "Reference of Workflow Policy Object Properties" on page 444.

Planning a Test and Migration Strategy for a Workflow Policy

This task is a step in "Process of Planning a Workflow Policy" on page 180.

Before implementing a new workflow policy, you must confirm that it works property in a test environment consisting of a sample Siebel database and workflow policies test data. Testing new policies, conditions, and actions checks that the policy you release into the production environment properly executes and does not cause conflicts with existing workflow policies.

Suggestions for setting up your test and migration strategy include:

- Make sure your test environment and production environment have identical versions of the software and that you are using realistic data in your database by using a partial or full copy of the production database.
- Create a small group of workflow policies to implement as a first phase of implementation. After you have successfully implemented the first group, you can add more policies in a systematic manner.
- To confirm a new workflow policy, go to your production environment, manually create a query based on the new policy, and check the response. This helps you determine if a policy creates excessive notification or misses the rows you need to monitor.

For more information, see "Migrating Workflow Policies to the Production Environment" on page 313.

Examples of Planning a Workflow Policy

This topic provides examples you can examine to plan workflow policies. It includes the following topics:

Example of Planning a Workflow Policy to Send Discount Notifications

This topic gives one example of planning a workflow policy. You might use this feature differently, depending on your business model.

In this example, the sales department manager needs to be automatically notified whenever sales representatives quote discounts over 30%.

To plan a workflow policy that sends notification for 30% discounts

1 Determine what to monitor.

Configuration information for a workflow policy that monitors quotes with a discount that exceeds 30%, for which the purpose is to notify the sales manager to review and approve the quote is described in the following table:

What to Monitor	Purpose of the Policy
Quotes with a discount exceeding 30% need Sales Manager approval	Notify Sales Manager to review and approve the quote.

2 Plan the workflow policy action.

Values for a workflow policy action that occurs as soon as five new quotes meet the criteria of the workflow policy conditions is described in the following table:

Name	Workflow Policy Object	Workflow Policy Group	Duration	Quantity	Description
Notify Sales Manager on Sales Approval	Quote	Low Frequency	0	5	Notify the manager when a quote with a discount over 30% is created.

3 Plan the workflow policy conditions.

The type of information you need for the policy conditions is described in the following table:

Field (Column Monitored in the Database)	Comparison	Value
Quote Status	=	In Progress
Quote Item Discount Percent	>	30

4 Plan the workflow policy actions that occur when the conditions of the policy are met.

You can also define the action arguments, such as the email subject and the message template, using dynamic values. Definitions for the Send Email to Sales Manager action is described in the following table:

Action Name	Workflow Policy Program	Workflow Policy Object	Arguments and Substitutions
Send Email to Sales Manager	Send Quote Email	Quote	Subject: Please approve quote discount for [Account] Message Template: Please approve the quote discount for quote [Quote Number] and notify [Last User First Name] [Last User Last Name] Repeating Message: The following quotes also need approval [Quote Number]

Example of Planning a Workflow Policy to Send Open Service Request Notifications

This topic gives one example of planning a workflow policy. You might use this feature differently, depending on your business model.

In this example, the service department needs to automate its notification policy when the number of open requests for an agent reaches 20.

To plan a workflow policy that sends notification of a large number of open service requests

1 Determine what to monitor.

A plan for the general policy definition is described in the following table:

What to Monitor?	Purpose of the Policy
Monitor open service requests when they reach a quantity of 20	Use Send Broadcast Message to alert the service representative about the situation.

2 Plan the workflow policy action.

A plan for the policy action is described in the following table:

Name	Workflow Policy Object	Workflow Policy Group	Quantity
Over 20 Open Service Requests	Service Request	High Frequency	20

3 Plan the workflow policy conditions.

After you determine the policy's workflow object and other properties, define the workflow conditions for your workflow policy. The plan for the workflow policy conditions is described in the following table:

Field (Column Monitored in the Database)	Comparison	Value
Service Request Status	=	Open

4 Plan the workflow policy actions that occur when the conditions of the policy are met.

You can also define the action arguments. A plan for the action argument definitions is described in the following table:

Action Name	Workflow Policy Program	Workflow Policy Object	Arguments and Substitutions
Alert Agent of Open SR	Send SR Message Broadcast	Service Request	Abstract: You have over 20 service requests Message Template: You have over 20 service requests. Please review your service request queue.

Process of Creating a Workflow Policy

To create a workflow policy, perform the following tasks:

1 Creating a Workflow Policy Group on page 188

2 Creating a Workflow Policy Action on page 188

3 Creating a Workflow Policy on page 189

Before creating a new workflow policy, check to see if there is a predefined workflow policy that meets your business requirements. For more information, see "Predefined Workflow Policy Programs" on page 394.

For more:

- Examples of creating a workflow policy, see "Example Workflow Policy Configurations" on page 191.
- Background information about objects described in this topic, see "About the Workflow Policies Module" on page 171.
- Information about modifying the compiled objects on which these policies are based, see "About Defining Workflow Policy Objects" on page 213.

Creating a Workflow Policy Group

This task is a step in "Process of Creating a Workflow Policy" on page 187.

You create a group for workflow policies in the Workflow Policies Groups view. Applets on the Workflow Policies Groups view include:

- **Workflow Groups applet.** Allows you to create new policy groups and to view and choose previously existing policy groups. Specifying a workflow policy group determines the monitoring cycle for a workflow policy.

  NOTE: It is recommended that each policy group contain policies that are monitored within similar time intervals. For more information, see "About the Workflow Groups Applet" on page 200.

- **Policies applet.** Lists the workflow policies assigned to the chosen group. For more information, see "About the Workflow Policies Applet" on page 201.

For planning information, see Planning a Workflow Policy Group on page 180.

To create a workflow Policy Group

1. In the Siebel client, navigate to Administration-Business Process > Policy Groups.
2. From the drop-down menu in the Policy Groups applet, choose New Record then enter the name for the group.
3. (Optional) Enter comments in the Comments field.

Creating a Workflow Policy Action

This task is a step in "Process of Creating a Workflow Policy" on page 187.

You must create the appropriate workflow policy action before you create the policy that uses the action. In the Siebel client, you use the Workflow Policies Action view to define the policy action. Applets on the Workflow Policies Action view include:

- **Actions applet.** Used to create a name for your action and choose the appropriate program. For more information, see "About the Actions Applet" on page 212.
- **Arguments applet.** Used to define the arguments for the action. The format of the Arguments applet changes depending on the program type of the action. For more information, see "About the Arguments Applet" on page 202.
- **Recipients applet.** Used to define the contact name, employee name, position, or relative of the workflow policy object that can receive an email, page, or broadcast message. For more information, see "About the Recipients Applet" on page 204.

NOTE: You cannot invoke a DLL or external function through a workflow policy action. Use a workflow process to invoke a DLL or external function.

To create a workflow policy Action

1. Navigate to Administration-Business Process > Actions.
2. In the Actions applet, click New Record from the applet level menu then enter the name of the workflow policy action.

   The name you specify appears in the Actions Applet of the Workflow Policies view.
3. Choose a workflow policy program type from the picklist in the Program field.
4. Choose a workflow policy object, if applicable, from the picklist in the Workflow Object field.

   If you specify a workflow policy object, this action appears only on the Actions Applet of the Workflow Policies view for policies associated with this workflow policy object.
5. Enter a description of the purpose of the action in the Comments field.
6. In the Arguments applet, pick one or more of the arguments and enter the appropriate value.

   The available arguments change according to the workflow policy program type you choose in Step 3.

   For a description of the Argument applet for specific workflow policy program types, see "About the Arguments Applet" on page 202.
7. If the workflow policy program is Send Email, Send Page, or Send Broadcast Message, enter the recipients of the action in the Recipient applet.

NOTE: You cannot invoke a business service from a Workflow Policy Action.

Creating a Workflow Policy

This task is a step in "Process of Creating a Workflow Policy" on page 187.

After creating your workflow policy group and workflow policy action, you are ready to use the Workflow Policies view to finish your workflow policy creation.

Table 37 describes applets in the Workflow Policies view in the Siebel client.

Table 37. Description of Applets Used to in the Workflow Policies View

Applet	Description
Policies List applet	Used to enter and view information about the workflow policy. The entry applet toggles with a list applet so you can move between working on an individual policy and viewing information about several policies or groups of policies.
Conditions applet	Used to define or change the conditions for the workflow policy. You can define as many conditions as necessary. The conditions for the policy must be met to trigger the workflow policy action. If you need the policy to be triggered when one or another condition is true, you must create a separate workflow policy for each condition.

Table 37. Description of Applets Used to in the Workflow Policies View

Applet	Description
Actions applet	Used to enter the name of the previously defined workflow policy action you need to take place when the conditions of the workflow policy are met.
Arguments applet	Used to review the workflow policy action arguments.

For more information, see "Applets Used to Define a Workflow Policy" on page 205. For example configurations, see "Examples of Creating a Workflow Policy" on page 197.

Restrictions With Creating a Workflow Policy

Restrictions when creating a workflow policy include:

- Workflow policies can not be based on the table S_DOCK_TXN_LOG. The unique index for this table is TXN_ID, rather than ROW_ID for other standard Siebel tables. Because Workflow uses ROW_ID to do the comparison and execute actions, it errors out if used against S_DOCK_TXN_LOG.
- You cannot execute a business service from a workflow policy.
- Workflow policies update database fields directly through the Data Layer, and do not go through the Business Object Layer. Therefore, a workflow process that includes business component events related to the updated fields is *not* executed.

Activities for Creating a Workflow Policy

In this task, you create a workflow policy.

To create a workflow Policy

1. Navigate to Administration-Business Process > Policies.
2. In the Policies List applet, choose New Record from the applet level menu, then enter a policy Name, Workflow Object, and Policy Group.
3. In the Conditions applet, choose New Record from the applet level menu, then enter a Condition Field, Operation, and Value.
4. In the Actions applet, choose New Record from the applet level menu.
5. From the picklist in the Action field, choose the name of the action you created in the Workflow Policies Actions view. If necessary, check the Consolidate field.

If at some point in the future you need to reassign the workflow policy to another group, see "Moving a Workflow Policy to a Different Group" on page 302.

Example Workflow Policy Configurations

This topic provides example configurations for workflow policy actions and workflow policies. It includes the following topics:

Examples of Creating a Workflow Policy Action

This topic describes examples that use a workflow policy action to address a specific business need. It includes the following topics:

You can use these examples as the basis for creating your own workflow policy actions.

Example of a Workflow Policy Action That Sends a Page

This topic gives one example of using a workflow policy action to send a page. You might use this feature differently, depending on your business model.

You can have a page sent to the support manager whenever a service request priority becomes very high and the service request is not assigned to anyone. Use the following procedure to define a workflow policy action that addresses this situation.

To configure a workflow policy Action to send a page when a service request is set to the highest value

1. In the Siebel client, navigate to Administration-Business Process > Actions.
2. In the Actions applet, create a new record using values described in the following table:

Field	Value
Name	Page Support Manager when SR request change.
Program	Send SR Page
Workflow Object	Service Request

The Workflow Object field populates automatically only when a workflow policy object is specified in the workflow policy Program field. You choose a workflow policy object from the picklist when it does not automatically populate.

3 In the Send Page Arguments applet, specify the argument using values described in the following table:

Field	Value
Alpha Message Template	The [SR Status] of [SR Number] has changed.

You use the Numeric Message Template for numeric paging and the Alphanumeric Message Template for alphanumeric paging. The type of paging to use is indicated by the pager type in the employee table.

Note that you can copy and paste fields that are available from the Available Substitutions window.

4 In the Recipients applet, create a new record using values described in the following table:

Field	Value
Type	Send to Position
Name	Support Manager

This action is now available to use in a workflow policy.

5 Navigate to Administration-Business Process > Policies.

6 Query the Workflow Object field for Service Request.

This returns a list of policies where the action you just created can most appropriately be used.

7 In the Name column, choose a Workflow policy, such as *Page SR Owner (Gold)*.

8 In the Actions applet, create a new record, then locate `Page Support Manager when SR request` changes, which is the action you just created.

9 Examine the Send Page Arguments applet.

Note that the Send Page Arguments applet populates with values you specified in this procedure.

Example of a Workflow Policy Action That Sends an Email With a Repeating Message

This topic gives one example of using a workflow policy action to send an email with a repeating message. You might use this feature differently, depending on your business model.

In this example, the vice president of sales needs to be notified only after a specific number of deals fail to close. Because this action is used with a workflow policy that uses the Batch feature, you enter relevant information in the Repeating Message field of the Send Message Arguments applet. This is because the recipient receives one email with a consolidated list of the pertinent information on each of the deals. Without a Repeating Message, the email is sent but might not contain meaningful information.

To configure a workflow policy Action to send an email with a repeating message

1 In the Siebel client, navigate to Administration-Business Process > Actions.

2 In the Actions applet, create a new record using values described in the following table:

Field	Value
Name	Excellent Quality Opportunity
Program	Send Opportunity Email
Workflow Object	Opportunity
Comment	Send an email to the VP of Sales when deals aren't closing

3 In the Send Message Arguments applet, specify the argument using values described in the following table:

Field	Value
Subject	Opportunities not Closing
Message Template	Meet with [Last User First Name] [Last User Last Name] about [Opportunity] for [Account]
Repeating Message	Meet with [Last User First Name] [Last User Last Name] about [Opportunity] for [Account]

4 In the Recipients applet, create a new record using values described in the following table:

Field	Value
Type	Send to Position
Name	VP Sales

This action is now available to use in a workflow policy.

5 Navigate to Administration-Business Process > Policies.

6 Query the Workflow Object field for Opportunity.

This returns a list of policies where the action you just created can most appropriately be used.

7 In the Name column, choose a Workflow policy, such as New Opportunity.

8 In the Actions applet, create a new record, then locate `Excellent Quality Opportunity`, which is the action you just created.

9 Examine the Send Page Arguments applet.

Note that the Send Page Arguments applet populates with values you specified in this procedure.

10 In the Policies List applet, make sure the Batch Mode field has a check mark.

Example of a Workflow Policy Action That Sends a Broadcast Message

This topic gives one example of using a workflow policy action to send a broadcast message. You might use this feature differently, depending on your business model.

In this example, a service department needs to automate its notification policy for open service requests when there are at least 20 open requests for one agent.

To configure a workflow policy Action to create a broadcast message for open service requests

1. In the Siebel client, navigate to Administration--Business Process > Actions.
2. In the Actions applet, create a new record using values described in the following table:

Field	Value
Name	Alert Agent of Open SRs
Program	Send SR Message Broadcast
Workflow Object	Service Request

3. In the Send Message Broadcast Arguments applet, specify the argument using values described in the following table:

Field	Value
Abstract	You have over 20 Service Requests.
Message Template	You have over 20 service requests. Please review your service request queue.

4. In the Recipients applet, create a new record using values described in the following table:

Field	Value
Type	Send to Relative
Name	SR Owner

This action is now available for use in a workflow policy. For instructions on how to add an action to a workflow policy, see "Example of a Workflow Policy Action That Sends a Page" on page 191.

Example of a Workflow Policy Action That Executes a Database Operation

This topic gives one example of using a workflow policy action to execute a database operation. You might use this feature differently, depending on your business model.

Insert and update are the two kinds of database operations that are possible in Workflow Policies. Insert allows a record to be inserted into a table in the Siebel database. The update database operation allows one or more columns in an existing record to be changed.

In the following example, a database update occurs when you use Workflow Policies to update the value of the Priority field to Very High if the Severity is Critical.

To configure a workflow policy Action that updates service request Priority

1. In the Siebel client, navigate to Administration-Business Process > Actions.
2. In the Actions applet, create a new record using values described in the following table:

Field	Value
Name	Update SR Priority to Very High
Program	Change SR Priority
Workflow Object	Service Request

3. In the Arguments applet, specify the argument using values described in the following table:

Field	Value
Name	New Priority
Value	1-ASAP

This action is now available for use in a workflow policy. For instructions on how to add an action to a workflow policy, see "Example of a Workflow Policy Action That Sends a Page" on page 191.

Example of a Workflow Policy Action That Runs an External Program

This topic gives one example of using a workflow policy action to run an external program. You might use this feature differently, depending on your business model.

In Siebel Workflow you use the action type Run External Program for defining an action that runs an external program. For example, your company can write a custom executable for calculating the quality of a new lead. You can then call this executable from a workflow process whenever the parameters for calculating the lead change.

In the first of the following procedures, a program named leadcalc.exe is in the C:\bin directory and the action is being defined to call and execute this program. The second procedure provides details for running an external program on UNIX.

To configure a workflow policy Action that runs an external lead calculation program

1 In the Siebel client, navigate to Administration-Business Process > Actions.

2 In the Actions applet, create a new record using values described in the following table:

Field	Value
Name	Run Lead Calculation Program
Program	Run External Program
Workflow Object	(Specify the workflow object involved in this action, such as Account)

Note that after you step off the record, you cannot modify the Workflow Object.

The workflow object field populates automatically only when a workflow policy object is specified in the workflow policy program being chosen, and if the workflow policy program as a workflow object already predefined. You pick a workflow policy object from the picklist when the picklist does not automatically populate.

3 In the Run External Program Arguments applet, set values described in the following table:

Field	Value
Executable Name	leadcalc.exe
Command Line	(Enter command-line parameters. These are the parameters you need to pass to the executable.)
Execute Type	(Choose an execute type.)
Available Substitutions	(Choose the appropriate dynamic fields)

In this applet, you define the executable and information you need to pass to the external program

4 In the Recipients applet, create a new record using values described in the following table:

Field	Value
Type	Send to Position
Name	Support manager

This action is now available for use in a workflow policy. For instructions on how to add an action to a workflow policy, see “Example of a Workflow Policy Action That Sends a Page” on page 191.

To run an external program on a UNIX platform

The Run External Program workflow policy program is not supported on UNIX. However, you can use the following procedure as a workaround.

1. Define a business service that executes an external program:
   a. Navigate to Business Service Administration > Business Service Methods.
   b. Add a new Business Service. For example, Run Program.
   c. Add a new Method. For example, Run.
   d. Add a new Method Argument. For example, Program.
   e. Select Proc: Service_PreInvokeMethod.
   f. Call Clib.system in the function body. For example:

```
var program = Inputs.GetProperty (“Program”)
if (program)
{
Clib.system(program);
}
return (CancelOperation);
```

2. Create a workflow process that calls the business service created in Step 1:
   a. Add and connect a Start step, a Business Service step, and an End step.
   b. For the Business Service step, specify Run Program and Run.
   c. For the input argument for Program, specify the external program you need to run. For example, /bin/mail hkim@pcs.com </home/users/hkim/letter.txt.
3. Run your workflow process.

Examples of Creating a Workflow Policy

This topic provides examples of creating a workflow policy. It includes the following topics:

Example of a Workflow Policy That Sends Paging Notification of Unassigned Service Requests

This topic gives one example of using a workflow policy to send a page. You might use this feature differently, depending on your business model.

In this example, the support manager needs a page sent whenever a service request priority becomes Very High and no one is assigned to the service request. The Send Page action has already been created. Next, you create the workflow policy to implement the workflow policy action.

To create a send page workflow policy

1. Navigate to Administration-Business Process > Policies.
2. In the Policies List applet, choose New Record from the applet level menu. Define a new policy record using values described in the following table:

Field	Value
Name	Page Support Manager
Workflow Object	Service Request
Policy Group	High Frequency
Duration	2 hours

3. Define the conditions under which the Page Support Manager policy applies. In the Conditions List applet, add two new condition records using values described in the following table:

Condition Field	Operation	Value
Service Request Priority	=	High
Service Request Owner	IS NULL	(leave empty)

4. Scroll down to the Actions applet, then add an Action.
5. From the picklist in the Action field, choose the appropriate notification action.

   For example, *Page for Critical SR Gold Support*.

Example of a Workflow Policy That Sends Email Notification of Open Opportunities

This topic gives one example of using a workflow policy to send an email. You might use this feature differently, depending on your business model.

In this situation, the vice president of sales needs to be notified when the number of deals that are not closed reaches a designated level. In this case, you have already created a workflow policy action that batches information on the deals and sends an email message containing information on the number of deals you designated.

To create a send email workflow policy

1 Navigate to Administration-Business Process > Policies.

2 In the Policies List applet, chose New Record from the applet level menu. Define a new policy record using values described in the following table:

Field	Value
Name	Very High Value Opportunity
Workflow Object	Opportunity
Policy Group	Medium Frequency
Quantity	5
Batch	checked

NOTE: It is not necessary for you to define the Quantity field to have a repeating message.

3 Define the conditions under which the Very High Value Opportunity policy applies. In the Conditions List applet, add two new condition records, using values described in the following table:

Condition Field	Operation	Value
Forecast Probability	< =	50
Forecast Revenue	>	400000

4 Define the Action to take when the conditions are met. In the Actions applet, add a new action record using values described in the following table:

Action	Consolidate
Excellent Quality Opportunity Email	Checked

5 Scroll to the bottom of the screen and examine the Send Message Arguments applet. Note that values in the applet automatically populated with information from the Action you defined in Step 4.

6 In the Actions applet, click the Excellent Quality Opportunity E-mail hyperlink in the Action applet. Examine values in the Actions applet and Send Message Arguments. Note that these values can be modified to meet the specific needs of your implementation.

Manipulating the Workflow Policy Monitor to Align Timing of Email Creation

To make sure the email action works properly, you can restart the Workflow Policy Monitor and Workflow Policy Action agents and set the Workflow Policy Action parameter Sleep to at least 5 minutes.

These actions make sure your email lists the opportunities that meet the conditions of the workflow policy. If the Workflow Policy Action agent runs too fast, then there will be an individual email each time the conditions of the workflow policy are met. Starting server processes is discussed in "About Workflow Policy Administration" on page 275.

About Views and Applets Used to Define a Workflow Policy

This topic provides information about the views and applets used to define a workflow policy. It includes the following topics:

Applets Used to Define a Workflow Policy Group

This topic provides information about the views and applets used to define a workflow policy group.

About the Workflow Groups Applet

Table 38 describes fields in the Workflow Groups applet.

Table 38. Description of Fields in the Workflow Groups Applet

Field	Usage
Name	Define the name of the workflow policy group. Workflow policies belong to one and only one policy group. This name can be no more than 30 characters long.
Comments	Define comments describing the purpose or use of the policy group.

About the Workflow Policies Applet

Table 39 describes fields in the Policies applet of the Workflow Policies Groups view.

Table 39. Description of Fields in the Policies Applet

Field	Usage
Name	Define the name of the policy included in the workflow policy group.
Workflow object	Define the workflow policy object for which the policy was created. For example, Service Request, Opportunity, or Quote.
Activation Date/Time	Define the date and time that the policy was or will be activated. If this field is null, the policy is activated immediately.
Expiration Date/Time	Define the date and time that the policy expires. If this field is null, the policy is active indefinitely.
Comments	Define comments describing the purpose or use of the policy.

Applets Used to Define a Workflow Policy Action

This topic provides information about the views and applets used to define a workflow policy group.

About the Actions Applet

Table 40 describes fields of the Workflow Policies Actions applet.

Table 40. Description of Fields in the Actions Applet

Field	Usage	Possible Value
Name	Define the name of the workflow policy action.	A descriptive name that is: ■ Consistent with your overall naming strategy ■ Meaningful to the policy maker
Program	Define the workflow policy program associated with the action.	This program is chosen from a picklist. For more information, see "Using the Run External Programs Type" on page 202.

Table 40. Description of Fields in the Actions Applet

Field	Usage	Possible Value
Workflow Object	Define the workflow policy object that this action is associated with. This action is now available only to policies that are based on this workflow policy object.	Chosen from a picklist of workflow policy objects.
Comments	Define comments describing the purpose or use of this action.	Enter comments text.

About the Arguments Applet

This topic describes how the arguments applet works.

For a description of each workflow policy program type, the available workflow policy program arguments and valid values, see "About Predefined Workflow Policy Program Types" on page 394.

Applets Refresh Automatically

The applets displayed in the Workflow Policies view change automatically depending on the program type you choose for the workflow policy action. This behavior is different from most views you use in the Siebel client.

For example, when navigating to Administration-Business Process > Policies, the following applets are displayed: Policies List, Conditions, Actions, Arguments. If you choose Send Page to Opportunity Sales Rep in the Action field of the Actions applet, the Arguments applet is automatically replaced with the Send Page Arguments applet.

Other Considerations to Weigh

Other considerations to weigh include:

- Program arguments are case sensitive. You must enter the correct case. Use the argument picklists when possible instead of entering the arguments yourself.
- Before using email or paging, you must perform the setup procedures described in "About Workflow Policy Administration" on page 275.

Using the Run External Programs Type

The External Programs Arguments applet appears if a Run External workflow policy program type is chosen in the Program field of the Actions applet.

For more information, see "Example of a Workflow Policy Action That Runs an External Program" on page 195.

Table 41 describes arguments for the Run External workflow policy program.

Table 41. Description of the Run External Workflow Policy Program Type

Argument	Description
Executable Name	Path and name of executable to run. For example, the executable launches from the Siebel Server. The executable can be a batch program.
Command Line	The command line to use. Include the parameters that you need to pass to the executable. If passing multiple substitution parameters, you must include a space between each parameter. For example: "[FirstName]"^"[LastName]", where ^ is a space. A substitution parameter that contains spaces must be encased in quotes. Note that the quotes are also passed as part of the parameter. For more information, see *Siebel Object Types Reference*.
Execute Type	Settings for Execute Type include: ■ **Wait.** Workflow Policies waits for the external program to finish and examines the return code of the external program. If the return code is not 0, an error condition occurs. ■ **No Wait.** Workflow Policies executes the external program in the background then continues processing. The return code is not checked. Note that for Visual Basic programs which create files, set Execute Type to Wait to avoid possible corruption of files. When set to No Wait, Visual Basic attempts to write files twice, thus corrupting the data.
Available Substitutions	Dynamic fields that can be used as command line parameters. When the action executes, the substitution value is populated with the value from the record in violation.

If no path is supplied for the Executable Name argument, the executable is assumed to be in the current path of Workflow Policies running on the Siebel Server. For example, if your Siebel Server is installed on C:\siebsrvr, then the default path for the executable name is C:\siebsrvr\bin.

NOTE: The external program cannot be one that is interactive, requires a user interface, or accesses the Windows desktop.

About the Recipients Applet

The Action Recipients applet is only available when a Send Email, Send Page, or Send Message Broadcast program is chosen in the Actions applet. Table 42 describes fields in the Recipients applet.

Table 42. Description of Fields in the Recipients Applet

Field	Usage	Possible Value
Type	Define the possible values that are dependent on the workflow policy program defined for the action. Recipients apply to workflow policy programs of the following types: Send Email, Send Page, and Send Broadcast Message.	Possible values include: ■ **Send to Employee.** Picklist of employees. ■ **Send to Position.** Picklist of positions. ■ **Send to Contact.** Picklist of contacts. ■ **Send to Relative.** Send to an individual or group of individuals, such as a service request owner or opportunity team member, related to the Workflow object, such as an Opportunity or Service Request. ■ **Send to Address**. Represents a direct email address for programs that send email.
Name	Define the Name of the recipient based on the recipient type.	Contact name, employee name, position, or relative of workflow policy object to send an email, page, or broadcast message. Note that messages are sent based on position. When choosing an employee as the recipient, if multiple employees occupy the same position, the message is sent to every employee, even though you are only choosing one employee as the recipient.

About the Send to Relative Recipient Type

The Send to Relative recipient type sends an email or page to an individual or position associated with the current record. For example, you can send an email to the primary sales representative of an opportunity when the conditions of the policy are met.

Send to Relative is also used when you need to configure a custom Send to [xxxx] recipient. Because you must use one of the Recipient Type choices in the picklist, you use Send to Relative to define a custom recipient type. The Recipient Type choices are Send to Employee, Position, Contact, or Relative.

Email Manager does not send email to the same recipient twice for the same action. If it detects that an email has already been sent to a specific email address, another one is not sent. If the Send to Relative type returns more than one recipient, each recipient is sent an email as long as each email address is unique.

Sending an Email to Multiple Relative Recipients

A workflow policy can be configured so that the same email message is sent to a number of different relative recipients.

to send an email to multiple relative recipients

1. Create an action for the policy program.
2. In the Recipients applet, add a new record and set the Type field to Send to Relative and the Name field to the name of the first recipient.
3. Repeat step Step 2 for each additional recipient, updating the Name field in the Recipients applet for each recipient.
4. Associate the action with the workflow policy.

Applets Used to Define a Workflow Policy

This topic provides information about the views and applets used to define a workflow policy group. It includes the following topics:

About the Policies List Applet

Table 43 describes fields that appear in the Policies List applet.

Table 43. Description of Fields in the Policies List Applet

Field	Usage	Explanation
Policy Name	Define the name of the workflow policy.	(This cell is intentionally empty.)
Workflow Object	Define the workflow policy object for which the policy was created.	For example, Service Request, Activity, or Accounts. This field is required.
Group	Define the workflow policy group to which the policy belongs.	Each policy must be assigned to a workflow policy group. For more information, see "Moving a Workflow Policy to a Different Group" on page 302.
Comments	Define comments about the purpose or use of the policy.	(This cell is intentionally empty.)

Table 43. Description of Fields in the Policies List Applet

Field	Usage	Explanation
Activation Date/Time	Define the date and time that the policy was or will be activated.	If this field is null, the policy is activated immediately.
Expiration Date/Time	Define the date and time that the policy expires.	If this field is null, the policy is active indefinitely.
Duration	Define the duration fields in days, hours, or minutes that the conditions must be true for the workflow policy to be executed.	This field is ignored if the policy is run in batch.
Created By	Define the login name of the person who created the workflow policy.	The information in this field is automatically filled. This field is read only.
Created On	Define the date and time the workflow policy was created.	The information in this field is supplied for you. This field is read only.
Quantity	Define the number of records that meet the policy conditions before the policy action executes.	If you do not specify a quantity, Siebel Workflow assumes a quantity of 1. Quantity allows policy administrators to create conditions that are based on a number of records that meet the policy conditions. For example, an administrator can create a workflow policy that sends a broadcast message when 20 or more critical service requests are open.
Batch	Specify, by checking Batch, that this policy evaluates records that potentially meet the conditions of the policy.	The Workflow Monitor Agent scans records using the conditions of the policy to find the matches. When this field is checked, run Workflow Monitor Agent with the Batch Mode flag set to TRUE. The default is unchecked.

About the Conditions Applet

Table 44 describes fields in the Conditions applet of the Workflow Policies view.

Table 44. Description of Fields in the Workflow Policy Conditions Applet

Field	Required	Usage	Example
Field	Yes	Define the workflow policy component column in the workflow policy object on which the workflow policy condition is based.	To specify the service request priority or service request open date, choose the workflow policy column instance from the picklist for the field.
Comparison	Yes	Define the comparison to make between a workflow policy agent's column value and the value you specify.	For example, equals (=) or greater than (>). Choose the comparison from the picklist for the field. For more information, see "About Logical Operators in the Conditions Applet" on page 207.
Value	Yes, except when the Comparison field has a value of Is Null, Is Not Null, Is Updated, Is Deleted, or Is Added.	Define the value to compare to the workflow policy column value instance.	For example, not started or very high. For more information, see "About Logical Operators in the Conditions Applet" on page 207, and "Entering Date Calculations in the Conditions Applet" on page 211.

NOTE: For a Workflow Policy to be triggered which has a Condition Field <> [Value *n*], there must be a value specified in this field other than that specified in the Workflow Policy condition. If there is no value, the workflow policy is not triggered. The Operator IS NULL is used in this case.

About Logical Operators in the Conditions Applet

You use comparison values in the Operation field. The field exposes the Workflow policy component column for monitoring.

Standard Comparisons in the Conditions Applet

The Comparison field supports greater than (<), less than (>), not equal to (<>), greater than or equal to (>=), less than or equal to (<=), equal (=), LIKE, IN, NOT IN, BETWEEN, IS NULL, and IS NOT NULL operators.

Table 45 describes comparison values for a typical database.

Table 45. Description of Comparisons for a Typical Database

Comparison	Value
<	5
>	5
<>	5
>=	5
<=	5
=	A
LIKE	Abc%
IN	(1, 2, 3)
NOT IN	('A', 'B', 'C')
BETWEEN	1 and 2
BETWEEN	'A' and 'B'

Your specific database syntax requirements might vary. Syntax rules include:

- When using LIKE, IN, NOT IN, or BETWEEN with character fields, you use single quotes around the value.
- When using IN or NOT IN, you must place the value within parentheses.
- An AND is implied between multiple conditions defined using these comparison values. AND means that conditions must be met before the action occurs.
- LIKE and NOT LIKE allow you to use wildcards. For example, LIKE Smith% or NOT LIKE Sm%th%.

Requirements of the underlying database include:

- When you specify values for the comparison operands LIKE, IN, NOT IN, and BETWEEN in the Value field of the Conditions applet of the Workflow Policies view, it must be in a form that the underlying database expects.
- IN, NOT IN, and BETWEEN require you to enter the database specific format for the field being examined. For example, IN ('a', 'b', 'c') or IN (1, 2, 3, 4) and BETWEEN 'A' and 'B' or BETWEEN 1 and 10.

- On an MS SQL Server database, when you create a workflow policy condition on a LONG column, the available comparisons are IS NULL, IS NOT NULL, LIKE, and NOT LIKE.

  NOTE: It is up to the policy creator to make sure the syntax is correct. Siebel Business Process Designer only passes the BETWEEN clause to the database. It does not confirm syntax, except for date and time. For date and time fields, Siebel Business Process Designer converts the date and time columns to the format of month/day/year, hour:minute:second.

Specialized Comparisons That Must Be Used With Batch Workflows

The Comparison field supports the specialized operators IS ADDED, IS UPDATED, and IS DELETED. The IS operators serve as a starting point for the examination of the workflow policy.

When creating a batch type workflow policy, the comparison operators IS ADDED, IS UPDATED, or IS DELETED *must* be used in conjunction with regular conditions. These comparison operators are considered special conditions intended for Dynamic mode when triggering rows to look up regular conditions.

Comparisons that work at the workflow policy component level, but do not operate at the field level, include:

- **IS ADDED.** If a new row is added for this workflow policy component, then trigger this workflow policy to be examined.

  If used in conjunction with standard comparisons, IS ADDED can be triggered when a record is updated.

- **IS DELETED.** If a row is deleted from this workflow policy component, then trigger this workflow policy to be examined.

Comparisons that operate at the field level include:

- **IS UPDATED.** If the field's value has changed by adding a new record with the specific field or by modifying the field in an existing record, then trigger this policy to be examined. To monitor if a field for a particular table was updated, use the workflow policy component column that represents the LAST_UPD column for that table.

  To monitor if a field within the workflow policy component was modified, use the field that is named after the workflow policy component.

Table 46 describes specialized comparisons for database platforms that can be used in creating a workflow condition.

Table 46. Description of Specialized Comparisons

Comparison	Value	Explanation
IS ADDED	Use IS ADDED with a workflow policy component column specified in the Condition field and nothing specified in the Condition value.	The condition is met when an instance of the workflow policy component is added. For example, if the Service Request policy component column is defined in the Condition field and IS ADDED is defined in the comparison, the condition is met when you create a new service request.
IS UPDATED	Use IS UPDATED with a field specified in the Condition field and nothing specified in the Condition value. The condition is met when the field changes.	For example, if service request status is defined in the Condition field and IS UPDATED is defined in the comparison, the condition is met when the Service Request status changes. For more information, see "Using IS UPDATED in the Conditions Applet" on page 211.
IS DELETED	Use IS DELETED when you specify a child workflow policy component in the Condition field, and nothing is specified in the Condition value.	A child workflow policy component is a workflow policy component that is associated with a parent workflow policy component in Siebel. For example, a parent workflow policy component might be Service Request. A child workflow policy component might be Service Request Activity. If IS DELETED is used in conjunction with other conditions, the other conditions must be based on the parent workflow policy component. For more information, see "Using IS DELETED in the Conditions Applet" on page 211.

Note that an OR is implied between specialized comparisons, where one or more of the conditions must be met before the action occurs. For example, you can have a service representative receive an email when an open service request has an activity added to it. Conditions in the policy that you then create include:

- Service Request Status = 'Open'
- Service Request Activity Component IS ADDED.

If a workflow policy condition is IS ADDED or IS UPDATED, then the triggers generated do not represent all the conditions specified in the policy. The triggers that are not represented are ignored. This condition can be viewed by examining the trigger.sql file that is produced as a result of the comparison. This is expected behavior. Note that if the workflow policy condition is modified then Gentrig must be run to implement the updated workflow conditions.

Note that when using a condition with a standard comparison, the condition is encompassed in the triggers that are created. However, when using a specialized comparison, since Workflow Monitor Agent evaluates the condition at runtime, triggers on the tables do not include the standard comparison conditions.

Using IS UPDATED in the Conditions Applet

Although Workflow policy conditions are joined when an IS UPDATED statement is present, the format of the trigger.sql statement that is generated for the workflow policy condition will not have AND within the SQL syntax.

When IS UPDATED is not present, the Workflow Monitor Agent only invokes the policy if conditions defined in the policy are violated. However, if an IS UPDATED comparison is included as criteria on a field, then other fields in the Policy conditions are not checked.

Using IS DELETED in the Conditions Applet

As an example, assume you must notify a service request owner if an activity is deleted from a service request that has a sub-status of In Process. Table 47 describes the configuration you use in this case.

Table 47. Example of Using IS DELETED

Policy	First Condition	Second Condition	Action
Based on the Service Request Workflow policy object	Condition is based on each of the following being true: ■ Field is Activity Component ■ Comparison is IS DELETED ■ Value is blank	Condition is based on each of the following being true: ■ Field is Service Request Sub-Status ■ Comparison is equal (=) ■ Value is In Process	Send an email to the SR owner

NOTE: When using IS DELETED, the ROW_ID of the deleted record from the child workflow policy component is not tracked or logged into the S_ESCL_REQ table, nor can the Workflow Monitor Agent component determine which exact row being deleted caused the policy to trigger. If you must have Workflow capture the row being deleted, you must use a workflow process driven by a run-time event. The run-time event is the PreDeleteRecord event on the BusComp event object type.

Entering Date Calculations in the Conditions Applet

Workflow Monitor considers both date and time when evaluating Workflow Policy conditions that perform a date comparison. CURRENT can be used when entering a value for a date comparison. The format for using CURRENT is CURRENT [+/-] d:h:m where *d* is day, *h* is hours, and *m* is minutes. For example, CURRENT + 01:02:03 is the value in CURRENT plus one day, two hours, and three minutes.

You can use CURRENT in the comparison value for date fields. You can also use CURRENT when you specify the activation and expiration dates for a broadcast message action.

For more information about time calculations, see "Date Formats with Workflow Policy Programs" on page 451.

About the Actions Applet

Table 48 describes fields in the actions applet of the Workflow Policies view.

Table 48. Description of Fields in the Actions Applet

Field	Usage
Action	Define the name of the action.
Sequence	Define the sequence of the action relative to other actions. This field is required.
Contact Last Name	Define the last name of a contact when the recipient of the action is a contact in the database.
Contact First Name	Define the first name of a contact when the recipient of the action is a contact in the database.
Employee Login	Define the login name of an employee when the recipient of the action is an employee.
Position	Define the position of an employee when the recipient of the action is a position.
Relative	Define the relative type when the recipient of the action is determined by the workflow object. For example, service request owner.
Consolidate Flag	Use this flag to consolidate the action to one instance if more than one record meets the conditions of the workflow policy during the same action interval. Default is FALSE. The consolidate flag is unavailable with actions that send pages.

Many of the properties in the Workflow Policies view are predefined in other Siebel Business Process Designer views by using the Siebel client or Siebel Tools. You can modify the predefined properties or define new values for these properties. The property values appear in picklists in the applets that constitute the Workflow Policies view.

Using an Insert Operation With a Workflow Policy Action

An insert operation with a Workflow Policy Action cannot populate the Primary Owner.

For example, you cannot modify the Workflow Policy Program, such as Create SR Activity, to populate Primary Activity Owner, OWNER_PER_ID, since the intersection table, S_ACT_EMP, must also be populated. Since the same Workflow Policy Program cannot update two tables within one database operation, Workflow Process must be used to populate OWNER_PER_ID. Earlier versions, such as Siebel version 6 can support this technique because Activity can have only one employee assigned. However, in later versions, such as Siebel version 7.x, Activity can be assigned to multiple employees.

About the Send to Position Recipient Type

The Send to Position recipient type allows you to send to the primary employee of this position without having to know the name of the person. The employee must be ACTIVE. The Send to Contact recipient type allows you to pick an available contact in the Siebel system.

About Defining Workflow Policy Objects

Values assigned to predefined object definitions for the various objects associated with configuring a workflow policy are modified in the Siebel client. You can use Siebel Tools to modify these predefined objects or to create new object definitions. After compiling your modifications, the modified definitions are available for use in the Siebel client.

This topic includes the following topics:

This topic provides information about defining the underlying workflow policy objects that, once compiled, are configured through the Siebel client. For more:

- Procedures for using the Siebel client to create a workflow policy that references these compiled objects, see “Process of Creating a Workflow Policy” on page 187.
- Predefined policies you can use instead of defining new objects, see “Predefined Workflow Policy Programs” on page 394.
- Reference information, see “Reference of Workflow Policy Object Properties” on page 444.

About the Display of Workflow Policy Objects

This topic describes how workflow policy objects are displayed in the Siebel client. For information about how workflow policy objects are displayed in Siebel Tools, see “About the Workflow Policy Object Hierarchy in Siebel Tools” on page 179.

About Workflow Policy Object Definitions in the Siebel Client

You use Siebel Tools to modify predefined or to create custom workflow policy object definitions and workflow policy programs. Once deployed, these customizations are visible in the Workflow Policy view and the Workflow Action view in the Siebel client.

Object Definitions Displayed in the Workflow Policy View

After object definitions for a Workflow Policy Object have been deployed from Siebel Tools, they appear in the Workflow Policy view in the Siebel client. Table 49 describes these objects.

Table 49. Description of Workflow Policy Objects Displayed in the Workflow Policy View

This Workflow Object Defined in Siebel Tools. . .	. . .appears in this picklist in the Workflow Policy View in the Siebel Client
Workflow Policy Object	Workflow Object picklist in the Policies List applet.
Workflow Policy Component Column	Condition Field picklist of the Conditions applet. Values displayed in the picklist for the Condition field are context-sensitive, depending on the value in the Workflow Object field in the Policies List applet.

Object definitions for the Workflow Policy Component do not appear in the Workflow Policy view. However, the Workflow Policy Component object type acts as an aggregator for values displayed in the Condition Field. Since the Workflow Policy Component Column object type is a child of the Workflow Policy Component, values for the Workflow Policy Component Column object definitions that are part of a given Workflow Policy Component are displayed in the Condition Field.

To view the deployed definitions, in the Siebel client you navigate to Administration-Business Process > Policies.

Object Definitions Displayed in the Actions View

After object definitions for a Workflow Policy Program have been deployed from Siebel Tools, they appear in the Workflow Action view in the Siebel client. Table 50 describes these objects.

Table 50. Description of Workflow Policy Program Objects Displayed in the Workflow Action View

This Workflow Object Defined in Siebel Tools. . .	. . .Appears Here in The Actions View in The Siebel Client
Workflow Policy Program	Program field picklist in the Actions applet
Workflow Policy Program Argument	Arguments field picklist in the Arguments applet Values displayed in the picklist for the Arguments Field are context-sensitive, depending on the value in the Program field in the Policies List applet.

To view the deployed definitions, in the Siebel client you navigate to Administration-Business Process > Actions.

Information specific to Workflow Policies is described in this document. For more information about general Tools usage, see *Using Siebel Tools*.

Defining Workflow Policy Objects

This topic describes how to use Siebel Tools to define object definitions that are used when configuring a workflow policy. It includes the following topics:

For more information, see "Examples of Predefined Workflow Policy Programs" on page 399.

Defining a Workflow Policy Column

This topic describes how to define a workflow policy column. Before you can add a workflow policy column to a workflow policy component, you must define the workflow policy column in the Workflow Policy Columns Object List Editor (OBLE).

The procedure has two parts. First, you identify the business object, business component, and applet that use the new workflow policy column. Second, you create the new workflow policy column.

CAUTION: Deleting a workflow policy column requires removal of references to the columns in workflow policy objects. Even if a workflow policy column is not currently being used in an active policy, the Workflow Monitor Agent is referencing the columns in the repository. This facilitates fast activation of new policies in the future, but care must be taken to remove old references if design requirements change so as to avoid potential link conflicts.

To define a new Workflow Policy Column

1. In the Siebel client, navigate to the view that will use the new policy column. For example, from the Accounts List, drill down on an account record, then choose the Activities view tab.
2. From the application-level menu, choose Help > About View.
3. Note the Business Object, Business Components, and Applets that this view uses.
4. In Siebel Tools, in the Business Components OBLE, choose one of the business components identified in Step 3, then note the value in the Table property.

   The Table property displays the table name for the Siebel database table that this business component references.

5 Choose Workflow Policy Column in the Object Explorer.

6 From the application-level menu, choose Edit > New Record.

7 In the Workflow Policy Columns OBLE, use values you found in previous steps of this procedure to define properties for the new object definition.

Note that the table name/column name combination must be unique. You are not allowed to save the object definition if the table name/column name combination has already been defined in another object definition.

Configuring a Workflow Condition Based on a Foreign Key

You can configure a workflow condition that is based on a foreign key that exists in the primary table of the workflow object. For example, SOPTY.CURR_STG_ID, where S_OPTY is the primary table of the Opportunity workflow object, and CURR_STG_ID is a foreign key from S_STG.NAME.

To configure a workflow condition based on a new Workflow Policy Column

1 In the Workflow Policy Columns OBLE, create a new workflow policy column object definition with S_STG.NAME in the Name property.

2 Make sure CURR_STG_ID is added under the Opportunity workflow component.

3 Create a new workflow component in the Opportunity workflow object based on the S_STG table, using values displayed is the following table:

Property	Value
Name	(your choice)
Source Table Name	S_STG
Source Column Name	ROW_ID
Target Component Name	Opportunity
Target Column Name	CURR_STG_ID

4 Add the new workflow column, S_STG.NAME, you created in Step 1 to the new workflow component you created in Step 3.

You can now create a workflow condition that is based on the new workflow column.

Defining a Workflow Policy Object

This topic describes how to define a workflow policy object.

CAUTION: In the Workflow Policy Components OBLE, you should define one, and only one record as the primary workflow policy component. Only one record should have the Primary property checked.

To define a new Workflow Policy object

1. In Siebel Tools, navigate to the Workflow Policy Objects OBLE.
2. From the application-level menu, choose Edit > New Record.
3. Define properties for the new record.
4. In the OE, choose the Workflow Policy Components object type.
5. From the application-level menu, choose Edit > New Record.
6. Make sure the Primary property has a check mark.

   Note the caution provided at the beginning of this topic.
7. Add more Workflow Policy Component object definitions, defining relationships to the primary workflow policy component, as necessary.
8. In the OE, choose the Workflow Policy Component Col object type.
9. In the Workflow Policy Component Columns OBLE, add object definitions for each of the workflow policy components you defined in Step 7.

Defining a Workflow Policy Component

This topic describes how to define a workflow policy component.

To define a Workflow Policy Component

1. In Siebel Tools, choose Workflow Policy Object in the OE.
2. In the OBLE, choose the object definition with Account in the Name property.
3. In the OE, expand Workflow Policy Object then choose the Workflow Policy Component object type.
4. In the OBLE, create a new record and enter a value in the Name property for the new Workflow Policy Component.
5. Enter a value in the Source Table Name property.
6. Enter a value in the Source Column Name property.

   The Source Column Name property establishes a relationship between the workflow policy component you are currently defining and the primary policy component.
7. Identify the set of columns for the workflow policy component you just defined that you must monitor:
   a. In Siebel Tools, navigate to the Workflow Policy Column OBLE.
   b. Identify the column in the predefined workflow policy columns that has an activity assigned to it but is not currently exposed in the Workflow Policy Component Column OBLE.

Defining a Workflow Policy Component Column

This topic describes how to associate a column with a workflow policy component.

Your business can use specialized terminology that more clearly defines the environment within your company. You can use the Alias property in the Workflow Component Columns OBLE to create a custom label. Then, you can use this label when referring to the workflow component column definition.

Values that appear in the Condition Field picklist on the Conditions applet of the Workflow Policies view in the Siebel client are populated from the Alias property.

Adding or Modifying a Workflow Policy Component Column

To create a new workflow policy object or if you need to add new columns to an existing workflow policy object, you add or modify a Workflow Policy Component Column object definition to the workflow policy object.

To add a Workflow Policy Component Column to a Workflow Policy Object

1. In Siebel Tools, navigate to the Workflow Policy Columns OBLE.
2. Query the Name property for the name of the column you need to modify.

   If the query returns no results, create a new Workflow Policy Column object definition that references the required data table and data table column.
3. Navigate to the Workflow Policy Objects OBLE, then locate the required object definition.
4. In the OE, choose the Workflow Policy Component object type, then locate the required object definition in the Workflow Policy Components OBLE.
5. In the OE, choose the Workflow Policy Component Col object type.
6. Define your customization:

   - If you are creating a new definition, from the application-level menu, choose Edit > New Record, then add values to the Workflow Column Name and Alias properties, as necessary.
   - If you are modifying an existing definition, locate the required Workflow Policy Component Column object definition, then modify the Alias property.

   When defining the Workflow Column Name property, use the picklist to choose a column from the current set of columns available for this Workflow Policy Component.

   Note that if the workflow policy component column that you must monitor is not in the list, you must define a new object definition for it by using the Workflow Policy Columns OBLE.
7. In the OE, choose the Workflow Policy Object object type.

   By default, the Workflow Policy Object with the name of the object you just modified should be the active record in the OBLE.
8. From the application-level menu, choose Tools > Compile Selected Objects, then click Compile.
9. In the Siebel client, navigate to Administration-Business Process > Policies.
10. In the Policies List applet, click the Query button.

11 In the Workflow Object field query for the workflow object you just modified, then click Go.

Note that your custom alias appears in the Condition Field picklist in the Conditions applet.

Example of Modifying a Workflow Policy Component Column to Display a Custom Organization Label

This topic gives one example of modifying a workflow policy column. You might use this feature differently, depending on your business model.

In this example, the sales department in your company needs to refer to the person who is handling the opportunity as the Opportunity Primary Sales Associate.

To modify a Workflow Policy Component Column to change the opportunity organization label

1 In Siebel Tools, locate the Workflow Policy Object named Opportunity in the Workflow Policy Objects OBLE.

2 In the OE, choose the Workflow Policy Component object type, then choose the Workflow Policy Component named Opportunity in the OBLE.

3 In the OE, choose the Workflow Policy Component Col object type.

4 Locate the Workflow Policy Component Column named Opportunity Primary Sales Rep Position, then change the Alias property to Opportunity Primary Sales Associate.

5 In the OE, choose the Workflow Policy Object object type.

By default, the Workflow Policy Object named Opportunity should be the active record in the OBLE.

6 From the application-level menu, choose Tools > Compile Selected Objects, then click Compile.

7 In the Siebel client, navigate to Administration-Business Process > Policies.

8 In the Policies List applet, click the Query button, enter New Opportunity in the Name field, then click Go.

9 In the Conditions applet, create a new record, setting the Condition Field to Opportunity Primary Sales Rep Position, and the Operation to IS ADDED.

Note that your custom alias appears in the Condition Field picklist.

Defining a Workflow Policy Program

This topic describes how to define a workflow policy program. A workflow policy program is a generic event that actions are based on. You define a program by defining the workflow policy event.

CAUTION: Do not rename or change the name of an existing workflow policy program. Doing so can result in the loss of actions created for the program.

When defining a Workflow Policy Program that inserts new records, you must determine and provide the minimum field values that constitute a valid record for the inserted record, as defined in the repository for the table. Values you should provide include:

- Provide values for required columns that constitute the inserted record. If a default value is defined for a column, that default value is used on the insert if the program specifies none. For example, S_EVT_ACT has two required columns: NAME and ROW_STATUS. ROW_STATUS defaults to Y so you do not have to set a value in the program.
- It is not necessary for you to provide a value for system generated columns such as CREATED, CREATED BY, LAST_UPD, LAST_UPD_BY, ROW_Id, MODIFICTION_NUM, CONFLICT_Id.

For more information, see *Siebel Data Model Reference*.

CAUTION: Thoroughly test SQL queries you plan to use with custom policy programs. Be aware that if the SQL statement fails to find rows, the workflow policies action is unable to process a token.

Copying an Existing Workflow Policy Program Object Definition

The advantage of copying an existing workflow policy program object definition is that if something goes wrong with your customized workflow policy program, you can always start over with the original, unmodified workflow policy program.

NOTE: It is recommended that when you define a new Workflow Policy Program, you first copy an existing workflow policy program object definition that is similar to what you need, then modify the copy to suit your specific business requirements.

Also, when you modify a copy of an existing workflow policy program, you can often reuse a significant portion of the preexisting configuration, which leads to fewer errors than if you create an entirely new workflow policy program.

To create a Workflow Policy Program

1. In Siebel Tools, in the OE choose the Workflow Policy Program object type.
2. In the OBLE, choose an existing Workflow Policy Program object definition that is similar to what you need for the new workflow policy program.
3. Right-click then choose Copy Record. This copies the entire program, including the program arguments.
4. In the new Workflow Policy Program object definition, modify the appropriate properties to meet the needs of the new program, such as Workflow Object.
5. Define the program arguments.

   Enter the arguments carefully to make sure capitalization, punctuation, and spelling are correct:

- Type the entries in the Name column *exactly* as indicated in Table 126 on page 452. Primary ID, Primary Table, Operation Type, SQL Statement, and SQL Statement Outputs must have one space between each word and each word must be properly capitalized. For example, Primary Id must have one space between the two words, capital P, and lowercase d.

  NOTE: In program arguments, the carriage return character that exists in SQL Statement and SQL Statement Outputs can cause unexpected behavior for a workflow policy program. In most cases, the substitution value is not substituted with the intended value but is instead substituted with the [Label] literally. Avoid using the carriage return character.

- When using SQL statements in program arguments, make sure the statements are specific to the particular RDBMS you are using.
- Type the names of the column pairs exactly: One space between each word, identically capitalized, one space in front of the left parenthesis and no spaces in the column.
- The order of the rows is not important.

NOTE: Before using a program and the program's related program arguments in a workflow policy, you must delete inactive or incomplete program argument definitions. These can cause Workflow Monitor Agent errors.

Defining a Workflow Policy Program Argument

This topic describes how to define a Workflow Policy Program Argument.

Example of Creating a Workflow Policy Program Argument to Send Opportunity Email

This topic gives one example of using a workflow policy program argument to send an email. You might use this feature differently, depending on your business model.

The example described in this topic adds a new a workflow policy program argument, in this case, Send Opportunity Email. The current recipients of type relative are limited to the Primary Sales representative. You need to add a relative for Primary Contact. This allows policy makers to create an action that sends an email to the Primary Contact for an opportunity.

To add an alternative Send to Relative to the send opportunity email program

1. In Siebel Tools, navigate to the Workflow Policy Programs OBLE.
2. In the OE, choose the Workflow Policy Program Arg object type.
3. Make sure an object definition named Send to Relative exists in the Workflow Policy Program Arguments OBLE.
4. Create a new record, setting the Name property to `Primary Contact`.

   Note that anew workflow policy program argument cannot have the same name as an SQL Statement Output. If an attempt is made to add a new program argument that does have the same name as an SQL Statement Output, the Monitor Agent server task suspends and display the message *`Examining request for policy`*.

5. Click in the Default Value property.

6 In the compose box, create your SQL statement. For example:

```
select O.PR_CON_ID, 'Send to Contact'
from &Table_Owner.S_OPTY O
where O.ROW_ID=?
```

Workflow passes the ROW_ID of the violating row, so be sure your SQL queries use the same ROW_ID. In this example, the WHERE clause is written to use the ROW_ID of the opportunity row that violates the policy.

Certain requirements must be considered when creating an SQL statement. For more information, see "Creating an SQL Statement for a Workflow Policy Program Argument" on page 222

TIP: SQL statements are database vendor–specific. Use an external SQL tool to build and test your statements. When the test works, copy the statement into the field.

7 Set the PickList property to Workflow Relative Type Picklist.

This picklist identifies this argument as a relative.

Note that the Visible field is checked for your new Workflow Policy Program Argument object definition. The changed field becomes checked when you create a new program argument.

Creating an SQL Statement for a Workflow Policy Program Argument

An SQL statement written for a Workflow Policy Program Argument must include:

- The table name and column name you reference must be uppercase.
- The case-sensitive table name must be prefixed with `&Table_Owner.`
- The SQL statement must be valid for the specific database vendor being used.

Considerations When Using SQL Statements for a Workflow Policy Program Argument

Considerations when using SQL statements for a workflow policy program argument include:

- It is recommended that a workflow program SQL statement return only one record. To be sure, initiate statements that use outer joins rather than inner joins.
- Only one SQL statement policy program argument can be used for a given Workflow Policy Program. Do not use two or more SQL statement program arguments for a Workflow Policy Program.

CAUTION: It is recommended you thoroughly test your SQL queries prior to implementing them as Workflow program SQL statements.

10 Testing a Workflow Process

This chapter describes how to test your workflow process before you deploy it. It includes the following topics:

In Siebel Tools you can use the Validate Tool to check a workflow process for semantic consistency errors and the Process Simulator to test the process for logic errors. You can also test a workflow process by using the business service simulator, and in real time by using the run-time environment. The tool you use to test your workflow process depends on the type of process you are testing.

For more information about developing high-level planning strategies for testing the Siebel application, see *Testing Siebel Business Applications*.

About the Validate Tool

The Validate tool in Siebel Tools is an error-correction tool you can use before simulation and deployment. You can launch the validate tool for a workflow process by using the context-sensitive right-click menu in either the process designer or the Workflow Processes Object List Editor (OBLE).

Validation enforces the semantic consistency of a workflow process that otherwise cannot be readily enforced by structural constraints. For example, using validation, you can make sure an error process does not itself contain an error process. When you validate a workflow process, you are given warnings about errors the workflow process contains. You can then correct the errors before running the Process Simulator.

Table 51 describes errors the Validate tool can detect.

Table 51. Description of Errors the Validate Tool Can Detect

General Description of Error	Result of Using Validation
Connectors not attached correctly.	Makes sure the workflow process diagram's branches are connected correctly.
Outgoing branches not specified for Decision points.	Makes sure you specify outgoing branches for each Decision point in the workflow process.
Business services and business service methods not specified for Business Service steps.	Makes sure each Business Service step in the workflow process is not missing a business service or a business service method.

Table 51. Description of Errors the Validate Tool Can Detect

General Description of Error	Result of Using Validation
Business components missing from Siebel Operation steps.	Makes sure you specify the business component upon which each Siebel Operation step acts.
Subprocess names not specified for Sub Process steps.	Makes sure each Sub Process step specifies the appropriate subprocess the workflow calls.

For information about how to use the validate tool, see "Validating the Workflow Process" on page 231

About the Testing Tools

This topic describes tools you can use to test a workflow. It includes the following topics:

About the Process Simulator

This topic provides information about the Process Simulator. It includes the following topics:

The Workflow Designer includes the Process Simulator, a simulation tool that allows you to step through a workflow process while viewing the results of each step. Simulating your workflow process before deploying it to your production environment verifies that resulting actions are accurate and useful, and that the results are as expected.

For more:

- Basic information about how the Process Simulator works, see "Workflow Simulation Architecture" on page 30.
- Procedural information about how to use the simulator, see "Preparing and Using the Process Simulator" on page 232.
- Detailed process simulator and Watch window usage instructions, see "Using the Process Simulator and the Watch Window to Test the Workflow" on page 334.
- Information about testing your workflow process within an overall planning framework, see "Roadmap to Developing a Workflow Process" on page 39.
- Using the Business Service Simulator in cases where the process simulator cannot be used, see "About the Business Service Simulator" on page 229.

Considerations for Using the Process Simulator

This topic provides information you should consider when using the Process Simulator.

Siebel Client Usage With the Process Simulator

Most workflow processes can be tested and debugged using the Process Simulator, which is hosted in Siebel Tools. To use the simulator, you must have the Siebel Mobile Web Client installed. The Siebel Mobile Web Client can connect to a development or local database that has the test data required to debug a workflow.

When you click the Start Simulation button on the Simulate toolbar, the client launches according to the debug settings you entered in "Preparing the Process Simulator" on page 232. You use this Siebel client instance as the run-time environment for the simulation. There are no actions you must take in this Siebel client instance unless the workflow being simulated is an interactive flow.

After the Siebel client is initialized successfully, the Simulation In Progress dialog box disappears, control passes back to Siebel Tools, the Start step executes, and the workflow is paused at the next step in the workflow. At this point, if the first step executes as expected, the next step of the workflow is highlighted in the Process Simulator view.

Workflow Mode and the Process Simulator

You can use the Process Simulator to test a workflow that runs in the Siebel client. This includes a service workflow, 7.0 workflow, interactive workflow, and a workflow based on a run-time event. You cannot use the Process Simulator to test a long-running workflow or a workflow that involves a server component, such as Workflow Process Manager, Server Request Broker, Assignment Manager, and Communications Server. To test a workflow that involves a server component, the workflow must be deployed to the run-time environment and tested using the application server.

Using the Simulator With an Interactive Workflow

When using the simulator with an interactive workflow you must perform some actions in the Siebel client while the simulator is running.

While running the simulator, if the highlighted step is a User Interact step, when you click the Simulate Next button the corresponding view defined for the step is displayed in the Siebel client.

When this occurs, switch to the Siebel client and make sure the run-time event is executed as expected by the User Interact step to resume workflow execution.

After the User Interact step is executed successfully in the Siebel client, control is passed back to Siebel Tools, and the highlight moves to the next step in the workflow.

Testing a Workflow That Involves Server Components

You cannot use the Process Simulator to test a workflow that involves server components, such as a long-running workflow process. If a workflow process involving a server component is run in the Process Simulator, incorrect behavior results. To test a workflow process that involves a server component, you test the workflow in the run-time environment.

For example, if you need to test a workflow process that invokes Siebel Assignment Manager, you deploy the workflow to the run-time environment. You export the workflow from Siebel Tools and import it to the Siebel client. Then you test the workflow in real time with the working server components.

Invoking a Workflow Process With the Process Simulator

Of the various ways to invoke a workflow process, invocation from the Process Simulator is an easy way to test and debug a workflow process. You can debug process steps as you define them in Siebel Tools, where the Process Designer and the Process Simulator both reside.

When the workflow process is run from the Process Simulator, it runs in the application object manager. Actual invocation of the process can be run in the application object manager or in the Workflow Process Manager server session, depending on specific parameters. Because some workflow processes that can run in the Workflow Process Manager server session might not be able to run in the application object manager, it is possible that every workflow process cannot be simulated.

Other ways of invoking a workflow process involve performing the invocation outside of Siebel Workflow. For more information, see "Invoking a Workflow Process" on page 137. For information about invoking a workflow process from a server component, see *Overview: Siebel Enterprise Application Integration* and *Siebel eMail Response Administration Guide*.

Using the Simulator With Scripts

You can use the Process Simulator with a workflow that references a business service or business component that contains script. However, if that script contains a breakpoint, and if the Arguments field in the Siebel Tools debug configuration contains /h, the Simulator might not perform as expected.

A breakpoint is a marker on a line of Basic code that tells Basic to suspend execution at that line so that the state of the program can be examined using the Siebel Debugger. The /h argument directs the Siebel debugger to open the Watch window.

Ways to avoid this problem when using the Process Simulator include:

- Remove all breakpoints from scripts on business services or business components the workflow process references.
- Remove the /h argument from the Siebel Tools debug configuration.

To remove the/h argument:

1. From the Tools application-level menu, choose View > Options, then click the Debug tab.
2. Remove the /h argument from the Argument field.

For more information about breakpoints, the /h argument, and Siebel Debugger usage, see *Using Siebel Tools*.

Using the Simulator With a Workflow Subprocess

To avoid a validation error, you must publish and activate subprocesses called by the workflow process you are simulating prior to invoking the simulator. For more information, see "Publishing a Workflow Process" on page 242.

Process Simulator Usage Considerations

Considerations when using the Process Simulator include:

- In the Siebel client, if the workflow process does not contain a user interact step, do not navigate or click UI elements while using the Process Simulator. If you must navigate in the Siebel client, it might be necessary to close the Siebel client and Siebel Tools, then open Siebel Tools to rerun the Simulator.
- In Siebel Tools, do not navigate anywhere outside of the Process Designer while the Process Simulator is running.
- After the Siebel client launches, it might be necessary to ALT+TAB back to Siebel Tools to proceed with the simulation.
- You can use the Process Designer to make changes to step properties, then return to the Process Simulator to test the process. To make changes to step properties, first stop the Process Simulator, then restart the simulation after changes are made.
- You might need to hide the Object Explorer and the Properties window to better view your work in Siebel Tools. You can also resize the Siebel Tools window so that it covers only a part of the visible display area.
- To use the Watch window, right-click the canvas, and then choose Watch Window. For more information, see "About the Process Simulator Watch Window" on page 228.
- The Workflow Process Simulator simulates a wait period if a Wait step is specified in seconds. However, if the unit of time is specified in minutes or greater, the Simulator simply moves on to the next step.
- A workflow process does not have to be active to run it in the Process Simulator. The simulator ignores activation date, expiration date, and status.

CAUTION: Your test environment and production environment must have identical software versions.

About the Simulate Toolbar

The Simulate toolbar includes several buttons that can be used to control execution of the simulation. Table 52 describes buttons in the Simulate toolbar.

Table 52. Description of Simulate Toolbar Buttons

Button	Description
Start Simulation	Activates the Start step in the process. Resumes the workflow process if it is paused.
Simulate Next	Activates the step immediately after the step that just executed.

Table 52. Description of Simulate Toolbar Buttons

Button	Description
Complete Simulation	Pauses the simulation.
Stop Simulation	Stops the workflow process.

About the Process Simulator Watch Window

The Process Simulator includes a Watch window that dynamically displays business component record values and process property values for the workflow process. These values are associated with and are manipulated by the workflow process being simulated.

As each step is finished during the simulation, values being manipulated by the simulation are dynamically updated and displayed in the Watch window. Table 53 describes information available in the Watch window.

Table 53. Description of the PropertySet of the Watch Window

PropertySet	Description
Simulator Status	Displays real-time status information for the simulation. For example, *Step Completed,* or *Simulation Ended Successfully.*
Process Properties	Displays current value for each process property defined for the workflow process. For example, the value 7-4HWSV is displayed for the Object Id process property.
BusComp	Displays business component user data for fields in the business component record currently being processed by the simulator. For example, the value 40000 is displayed for the Revenue field.

To view an example that includes Watch window usage instructions, see "Using the Process Simulator and the Watch Window to Test the Workflow" on page 334.

The Watch window is often used in conjunction with the Workflow Utilities business service to monitor process property values and business component data. For more information, see "About the Workflow Utilities Business Service" on page 382.

To use the Watch window

1. From within the Process Designer for the workflow process you must simulate, right-click the canvas then choose Simulate.
2. Click the Start Simulation button, then wait for the Siebel client to launch and return control to Tools.

3 Right-click anywhere in the Process Simulator canvas, and then choose Watch Window.

In the Watch window, the left column displays the name of the property set type. The right column displays the current value for the property.

The Watch window is not available until after you perform Step 2. You must start the simulation and wait for control to return to Tools prior to opening the Watch window.

Watch Window Usage Considerations

Considerations when using the Watch window include:

- You can hide the Watch window by right-clicking then choosing Hide Watch Window.
- If the Watch window appears empty, right-click in the simulation canvas, then choose Watch Window. This refreshes the display.
- In addition to opening the Watch window from within the Process Simulator canvas, you can also open the Watch window by choosing View > Debug Windows > Watch from the Siebel Tools application-level menu.

About the Business Service Simulator

A workflow process can be run as a business service from the Business Service Simulator in the Siebel client.

Use the Business Service simulator when you must debug script in conjunction with a workflow. You can set breakpoints in the script then execute the workflow in the Siebel Mobile Web Client. When the workflow executes a service for which a breakpoint is set, control is returned to the Script Debugger in Siebel Tools.

The workflow must be published and activated before the workflow can be tested with the Business Service simulator. To use the simulator, conditions that must be met include:

- The Siebel Mobile Web Client must be installed.
- The workflow process must be exported from Siebel Tools.
- The workflow process must be imported through the Siebel client.

TIP: Alternatively, you can publish and activate the workflow to the Siebel Mobile Web Client directly from Siebel Tools.

For more information about Business Service Simulator usage, see “Diagnosing a Failed Workflow Process in a Production Environment” on page 262.

About the Event Logs

To view more detailed information on the execution of a workflow, set event logs so that you can view the log files. This technique is useful if you cannot perform real-time debugging or do not have the Process Simulator readily available. However, using this technique can result in large log files that must be analyzed. For more information, see "Setting Tracing and Event Log Levels" on page 259. For information on using the Log File Analyzer, see *Siebel System Monitoring and Diagnostics Guide*.

Process of Testing a Workflow Process

This process is a step in "Roadmap to Developing a Workflow Process" on page 39.

To test a workflow process, perform the following tasks:

1. Validating the Workflow Process on page 231
2. Preparing and Using the Process Simulator on page 232
3. Verifying Functionality on page 234

Before deploying a new workflow process, you must test it in a test environment. Testing a new process verifies that the process you release into the production environment executes properly and does not cause conflicts with another existing workflow process.

When testing a workflow process, suggestions for setting up your test strategy include:

- Make sure your test environment and production environment have identical versions of the software.
- Use realistic data by using a partial or full copy of the production database.

This topic provides procedural information on how to test a workflow process. For background information, see "About the Validate Tool" on page 223, and "About the Process Simulator" on page 224. To view an example that includes detailed process simulator and Watch window usage instructions, see "Using the Process Simulator and the Watch Window to Test the Workflow" on page 334.

Validating the Workflow Process

This task is a step in "Process of Testing a Workflow Process" on page 231.

To validate a workflow process

1. In Siebel Tools, in the Workflow Processes OBLE, choose the process you need to validate.
2. Right-click then choose Validate.

   The Validate dialog box appears.
3. Click Start.

   Starting validation appears in the bottom left corner of the Validate dialog box.
4. If the validation is successful, there are no errors to report and in the bottom left corner of the dialog box, a message displays *Total tests failed: 0*.
5. If the validation is not successful, a list of errors are displayed along with the rules each error violates. Perform one of the following actions:

   a. Click the error to view the message in the Details window of the dialog box.

   b. Save the errors to a log file by clicking the Save As button at the bottom of the validation panel.

   Note that all errors are not fatal. Some errors are only warnings.

Preparing and Using the Process Simulator

This task is a step in "Process of Testing a Workflow Process" on page 231.

Preparing the Process Simulator

In this task you prepare the process simulator for use, which includes exposing the Simulate Toolbar, and preparing debug settings.

Table 54 describes debug run-time information you must configure.

Table 54. Description of Fields Used With the Siebel Tools Debugger

Field	Value
Executable	$SiebelClient\BIN\siebel.exe Make sure you enter the full path for the siebel.exe executable file.
CFG file	$SiebelClient\BIN\ENU\uagent.cfg Make sure you enter the full path for the uagent.cfg configuration file.
Browser	$\Program Files\Internet Explorer\iexplore.exe
Working directory	$SiebelClient\BIN
Arguments	/h
User name	$username
Password	$password
Data source	$datasource

To prepare debug settings for the Siebel client

1. In Siebel Tools, from the application-level menu, choose View > Toolbars > Simulate.

   This exposes the Simulate toolbar.

2. From the application-level menu, choose View > Options, then click the Debug tab.
3. Define the fields in the Debug tab according to the guidelines listed in Table 54, where $ represents settings specific to your setup.
4. Click OK.

Next, you use the process simulator.

Using the Process Simulator

In this task, you invoke then run the Process Simulator on a workflow process.

For more information, see "Considerations for Using the Process Simulator" on page 225. For examples that use the simulator, see "Using the Process Simulator and the Watch Window to Test the Workflow" on page 334, and "Testing the Workflow Process" on page 342.

Before you run the Process Simulator or deploy your workflow process, you should Validate the workflow. For more information, see "About the Validate Tool" on page 223.

CAUTION: When testing a workflow process using the Process Simulator, it is important to note that the workflow runs just as if it were called in a production environment. For instance, if the process includes a Siebel operation such as update or add, records in the database are modified when you run the Process Simulator.

To use the Process Simulator

1. Close all Siebel client sessions.

   To avoid encountering a multiple client session error, you must close all client sessions prior to launching the Process Simulator. You cannot start the Process Simulator if a Siebel client session is running.

2. Confirm there are no Siebel client icons in the Task Bar.

3. In the OBLE, locate then right-click the Workflow Process object definition you need to simulate.

   As an alternative, you can launch the simulator from within the Process Designer. To do so, launch the Process Designer for the workflow you need to simulate, right-click anywhere in the Process Designer background, then choose Simulate.

4. Choose Simulate Workflow Process.

   The Process Simulator opens. A read-only version of the workflow process diagram displays where the canvas appeared previously. The canvas is replaced with a pale yellow background, the start step is highlighted to mark the simulation's starting point, and the Start Simulation button on the Simulate toolbar becomes active.

5. In the Simulate toolbar, click the Start Simulation button to begin the simulation.

6. The Simulation In Progress dialog box appears momentarily.

7. Observe the Siebel client launch, then wait for control to return to Tools.

   A new instance of the Siebel client launches, displaying the Administration-Business Process Process Simulator screen. For more information, see "Siebel Client Usage With the Process Simulator" on page 225.

8. In the Simulate toolbar, click the Simulate Next button to execute the highlighted step.

9. Continue clicking the Simulate Next button until the last step has executed.

   As you step through the workflow process, examine results of each step in the Watch window until the workflow process finishes. For more information, see "About the Process Simulator Watch Window" on page 228.

10 If the workflow process does not execute as expected, take corrective action, then restart the simulation at Step 5.

   For more information, see "Troubleshooting Workflow Process Simulation" on page 235.

11 (Optional). Examine test results in the Siebel client.

   If your workflow process manipulates record data, you might be able to examine test results in the Siebel client. To view an example that manipulates record data through the process simulator, see "Testing the Workflow Process" on page 321.

12 Close the Simulator window.

After the last step has executed the simulation automatically ends. You can also use the Stop Simulation button to halt the simulation before the last step executes.

Verifying Functionality

This task is a step in "Process of Testing a Workflow Process" on page 231.

To finish testing for a workflow process, you should verify the workflow actually implements the required functionality that meets your business requirements. This is often performed by manipulating data in the Siebel client, then verifying workflow process execution matches the functionality that was specified during the planning phase. To view an example, see "Verifying Workflow Process Functionality" on page 360.

Troubleshooting Workflow Process Simulation

This topic provides guidelines for resolving workflow process simulation problems. Table 55 describes symptoms to look for to resolve the problem.

Table 55. Descriptions for Resolving Workflow Process Simulation Problems

Symptom or Error Message	Diagnostic Steps or Cause	Solution
When trying to simulate a workflow, upon starting the workflow, the simulator does not run.	If a Start step has an outgoing branch with a run-time event, the simulator waits on the event and never reaches the next step.	Choose one of the following approaches: ■ Remove the run-time event from the simulation. ■ Add a Default branch. **NOTE:** It is recommended you have two branches: one with Type set to Condition, to wait for the run-time event, and the other with Type set to Default.
When attempting to rerun the simulator, an error dialog displays *Still waiting on Workflow Process Simulator*, or *The run-time client is not responding*.	Navigating in the Siebel client during or after running the simulator can cause a failure during subsequent attempts at running the simulator.	Do not navigate in the Siebel client while the simulator is running or after the simulation ends. For more information, see "Process Simulator Usage Considerations" on page 227.
After completing the user interact step, the simulator fails to proceed down the workflow.	The branches out of the user interact step can be missing associated run-time events.	Make sure branches out of the user interact step have the respective run-time events associated with actions performed on the user interact step.

Table 55. Descriptions for Resolving Workflow Process Simulation Problems

Symptom or Error Message	Diagnostic Steps or Cause	Solution
After the Siebel client launches, control does not return to Tools, or a dialog in Tools indicates the client is still launching, even after a long wait.	(This cell is intentionally empty.)	Log out of the Siebel client session and close Siebel tools. Open Windows task manager. Click the Processes tab, right-click the dbeng9.exe Image Name, choose End Process, then click Yes. Right-click the siebel.exe Image Name, choose End Process, then click Yes. Restart Tools and run the Simulator again.
An error message appears during simulation execution, similar to *Error loading siebel operation step definition [step name]. The specified step definition is not valid.*	Errors in the workflow process configuration are revealed during the simulation, which can include errors in how objects are defined or how objects are used by the workflow process.	Close the simulator, correct the configuration errors, then run the simulation again.

11 Administering Workflow Processes

This chapter describes concepts and procedures for administering Workflow Processes. It includes the following topics:

Process of Deploying a Workflow Process

This process is a step in "Roadmap to Developing a Workflow Process" on page 39.

To deploy a workflow process, perform the following tasks:

1. Preparing the Run-time Environment for a Workflow Process on page 238
2. Publishing a Workflow Process on page 242
3. Activating a Workflow Process on page 242

Workflow definition and workflow deployment occur separately. After you have defined a workflow process in the Process Designer in Siebel Tools, you deploy the workflow process by preparing the run-time environment, publishing the workflow in Siebel Tools, then activating the workflow in the Siebel client.

When deploying, the workflow process definition is read from the repository then Siebel Workflow writes the definition to the run-time environment as XML. You deploy the workflow process with the workflow's deployment parameters, Replication, Activation Date, Expiration Date, and Monitoring Level. For more information, see "Setting Workflow Process Monitoring Levels" on page 258.

For a conceptual overview of this topic, see "Workflow Deployment Architecture" on page 31.

Preparing the Run-time Environment for a Workflow Process

This task is a step in "Process of Deploying a Workflow Process" on page 237.

This topic provides information about work you must perform to make sure the run-time environment is capable of running the workflow process. It includes the following topics:

Although you do not necessarily need to perform these topics in the order presented, you should consider each topic to determine if it applies to the workflow process you are deploying.

Making Sure Objects the Workflow Process References are Current

If the workflow process you are deploying contains Sub Process steps or references new repository objects such as business components, business services, and views, you must first make sure these subprocesses or repository objects are available to the workflow you are deploying. In the case of Sub Process steps, deploy the subprocess before deploying the parent workflow, so the subprocess is accessible to the parent workflow process. In the case of new repository objects, first compile the new repository objects so they are accessible to the workflow process you are deploying.

Activating Fields Used by a Workflow Process

Fields that are not activated by the Object Manager must be activated for Workflow to be able to reference and use them. When fields are exposed on the user interface, they are activated by the Object Manager, so a workflow process running on the Object Manager that references these fields is able to run properly. But when fields are not exposed on the user interface, they are not activated by the Object Manager. In this case, a workflow process running on the Object Manager cannot use these fields, and an error is generated.

To activate the fields necessary for a workflow process to run, perform one of the following tasks:

- In Siebel Tools, in the Fields Object List Editor (OBLE), explicitly activate the field by setting the field's Force Active property to TRUE.
- Expose the fields on the user interface.
- Activate the fields through scripting, such as a business service that activates the fields to be used before the workflow process uses them.

Deploying a Workflow Process to the Siebel Mobile Web Client

The Replication field in the Workflow Deployment view allows you to choose whether to route a workflow process definition to your Siebel Mobile Web Client. Routing only the workflow process definitions that your Siebel Mobile Web Client needs allows you to reduce the amount of data in the local database.

To set the Replication parameter, navigate to Administration-Business Process > Workflow Deployment > Active Workflow Processes. Table 56 describes possible values for the Replication field.

Table 56. Description of Parameter Settings for the Replication Field in the Workflow Deployment View

Field	Description
Replication	Values you can choose from include: ■ **All.** The workflow process definition is routed to Siebel Mobile Web Clients and regional nodes. ■ **Regional.** The workflow process definition is routed to the regional nodes only. ■ **None.** (Default) The workflow process definition is not routed to the Siebel Mobile Web Client or the regional nodes.

For more information, see "Deploying a Workflow Process on a Regional Node" on page 240.

Restricting Siebel Mobile Web Client Routing

When modifying the Replication field to choose whether to route a workflow process definition to your Siebel Mobile Web Client (MWC), keep in mind that changing the Replication field value from None to All adds the workflow process definition and related records to the Siebel Mobile Web Client or regional node when it synchronizes with the server.

Behavior with Full Copy Nodes

Note that if you extract a regional node with the routing group set to FULL COPY then workflow process definitions with Replication set to None are routed to the MWC. You can perform the following to confirm this behavior:

1. Modify two existing or create two new workflow processes, one with Replication set to All and the other with Replication set to None.
2. Extract a regional node with the routing group set to FULL COPY.
3. Examine the dx file in the regional node outbox to confirm that both workflow processes were routed.

Run-Time Event Behavior for Mobile Clients

There is some variation in the way run-time events behave in mobile clients compared with other Siebel client types.

NOTE: It is recommended that the processing mode be Local synchronous or remote asynchronous. If remote asynchronous is used for mobile clients, the workflow process is triggered after you synchronize with the server. For more information, see "About Remote Synchronous Processing" on page 144.

Periodically Notifying Mobile Users Who Have Not Synchronized

You can configure a workflow process to send a notification email to mobile users who have not synchronized over a given period of time. For more information, see 476188.1 (Doc ID) or 476275.1 (Doc ID) on Oracle*MetaLink* 3.

Deploying a Workflow Process on a Regional Node

You can execute a workflow process on regional nodes. The workflow can be called from script or run-time events.

When executing a workflow process on a regional node, the workflow must reside on the regional node. The settings and environment must be replicated entirely on the nodes. The objects the workflow is referencing must be available on the regional node.

Deploying a Workflow Process in a Multilingual Environment

A workflow process deployed from one language's object manager are not available on another language's object manager until after a restart.

For example, you create a workflow process that is called in a business component write-record event routine by scripting. You publish and activate this workflow, then restart the servers. You see that the workflow is called, both for callcenter_enu and callcenter_esn. Then you revise and publish this workflow from Siebel Tools. From within callcenter_enu, you activate this workflow. You see that callcenter_enu uses the revised workflow, but callcenter_esn does not. If you activate this workflow in callcenter_esn, an error results. You must restart the callcenter_esn object manager to get the new workflow.

When deploying a workflow that is used in multi-language deployments, the object managers for each language must be restarted.

Deploying a Workflow Process as a Web Service

A workflow process can be deployed as a Web service.

To deploy a workflow process as a web service

1. In Siebel Tools, in the Workflow Designer, examine definitions in the MVPW to determine if a process property has the Data Type argument field set to Integration Object. If so, make sure that each of those process properties has a value specified in the Integration Object argument field.

   If you have hierarchical data in a business service used by the workflow process, you must set Data Type for the process property to Integration Object, then specify an integration object in the Integration Object argument field.

   If no integration object is specified, the following error occurs: *The selected workflow process contains hierarchy type process properties without having the integration object name specified*.

   For more information, see "About Process Properties" on page 74.

2. Make sure the workflow process you need to deploy is published and activated.

   For more information, see "Publishing a Workflow Process" on page 242.

3. In the Workflow Processes OBLE, right-click the workflow process you need to deploy, then choose Deploy as Web Service.

   The Expose Workflow Process as Web Service dialog appears.

4. In the top window of the dialog, specify the operation name for the new Web service.

   Typically, you specify an underlying business service method name without spaces, such as *CreateAccount*. Use a method for a business service defined in the workflow process.

5. In the bottom window of the dialog, specify the URL that identifies the address for the Web service. Replace [webserver] with a valid host name and [lang] with a valid language code, such as enu.

6. (Optional) To generate a Web Services Description Language (WSDL) file, click the Generate WSDL checkbox, then specify a location to save the WSDL file.

7. Click Finish.

   The Web service is created.

8. To view the Web service you just created:

   a. In the Siebel client navigate to Administration-Web Services > Inbound Web Services.

   b. Query the Name column in the Inbound Web Services applet for the workflow process name you clicked in Step 3.

   To display Web services created from workflow processes and business services, query the Namespace column for `http://siebel.com/CustomUI`.

For more information, see *Integration Platform Technologies: Siebel Enterprise Application Integration*.

Publishing a Workflow Process

This task is a step in "Process of Deploying a Workflow Process" on page 237.

To publish a workflow process

1. In the Object List Editor, choose the workflow process you need to deploy.
2. In the WF/Task Editor toolbar, click the Publish button.

   For more information, see "About the WF/Task Editor Toolbar" on page 72.

Next, activate a workflow process.

Activating a Workflow Process

This task is a step in "Process of Deploying a Workflow Process" on page 237. Note that the first procedure describes how to activate a single workflow process. The second procedure describes how to activate workflow processes in batch.

To activate a workflow process

1. Log in to the Siebel client with administrator privileges, connected to the appropriate database.
2. Navigate to Administration-Business Process > Workflow Deployment.
3. In the Repository Workflow Processes applet, query for the workflow you published in Step 2 on page 242.
4. With the workflow process chosen, click the Activate button.

   Activate adds a new record which is visible in the Active Workflow Processes applet. Activate checks the syntax for validity, registers run-time events if used, and changes the status of the process to Active. If the workflow process has previously been activated, Activate also changes the status of the previous active version to Outdated.
5. In the Active Workflow Processes applet, set the deployment parameters for the workflow process:
   a. If necessary, query for the workflow you just activated.
   b. Set the activation date in the Activation Date/Time field.
   c. Set the expiration date in the Expiration Date/Time field.
   d. Make sure Replication is set to None, unless you are deploying the workflow process to the Siebel Mobile Web Client.

      If you are deploying the workflow to the Siebel Mobile Web Client, see "Deploying a Workflow Process to the Siebel Mobile Web Client" on page 239.
   e. Set the monitoring level in the Monitoring Level field.

      For more information, see "Setting Workflow Process Monitoring Levels" on page 258.

6 If a run-time event is defined in the workflow process, you must reload the run-time events.

   a Navigate to Administration-Runtime Events.

   b Click the applet menu then choose Reload Runtime Events.

   This reloads the run-time events in the current object manager session. To view an example, see "Deploying a Workflow Process Invoked By a Runtime Event" on page 345. For more information, see *Siebel Personalization Administration Guide*.

Now you can invoke the workflow process through the Process Simulator, a script, or a Workflow Policy. For more information, see "Invoking a Workflow Process" on page 137.

To activate workflow processes in batch

1 In the Siebel client, navigate to Administration-Business Process > Workflow Deployment.

2 Query for the workflows you need to activate.

3 Choose the workflows you need to activate, then click the Activate button.

   NOTE: It is recommended that you do *not* choose every workflow in the list.

Process of Migrating a Workflow Process

This process is a step in "Roadmap to Developing a Workflow Process" on page 39.

To migrate a workflow process, perform the following tasks:

1 Developing a Migration Strategy on page 244

2 Migrating a Workflow Process on page 248

Migrating is the act of moving workflow process definitions from a lower version repository to a higher version repository or from a development environment to a production environment. Once you have tested and published a workflow process in a development environment, you can migrate it to the production environment.

NOTE: Before migrating data, be sure data required for the workflow process is also present in the target environment. For example, if your workflow process requires custom entries in the List of values (LOV) table, make sure these are present and active.

Developing a Migration Strategy

This task is a step in "Process of Migrating a Workflow Process" on page 243. This topic includes the following topics:

When planning a migration strategy for a workflow process development effort, you should perform the following tasks:

1. Choose the migration tool you use to migrate workflow process objects.
2. Consider the requirement to redeploy your workflow process after it is migrated.

Once you have deployed the workflow process, it is ready to be migrated. *Migrating* is the act of making a workflow process available in other environments, such as migrating a workflow process from a development environment to a production environment. One of three utilities can be used: ADM, REPIMEXP, and Import/Export.

Considerations to weigh include:

- If you choose not to connect Siebel Tools to your production repository, you must use REPIMEXP for migrating your workflow process definitions.
- When you migrate workflow processes using ADM, the workflows are activated automatically. If you use REPIMEXP or the Siebel Workflow import/export option, you must redeploy the workflows manually. For more information, see "Redeploying a Workflow After it is Migrated" on page 248.

Comparison of Migration Options

Table 57 provides a comparison summary of options for migrating workflow processes.

Table 57. Comparison of Options for Migrating a Workflow Process

Use This Migration Option. . .	. . . When These Requirements Exist
REPIMEXP	You are rolling out your release and most or all repository objects must be migrated. REPIMEXP is the manual process for repository migration. Use this repository-migration option if you do not need to use ADM, or cannot use ADM.
ADM	You are migrating customization data for your entire enterprise, including workflow process data. You must migrate more than 10 workflow processes at one time.

Table 57. Comparison of Options for Migrating a Workflow Process

Use This Migration Option. . .	. . . When These Requirements Exist
Workflow Admin Service Business Service	You can perform bulk or batch migration of workflow processes. As an alternative to using the Workflow Admin Service Business Service, you can perform client-side batch activation and expiration by way of the File menu in the application.
Import/export	You can migrate workflow processes in increments of approximately 10 or less.

Migrating With REPIMEXP

The REPIMEXP utility allows for exporting importing of repository objects. Since this utility migrates repository objects, including your workflows, it is most useful when your organization is ready to roll out an entire release.

The Repository Import/Export utility is found in the siebel/bin directory. Use REPIMEXP for bulk migration of repository objects, including workflow definitions. In the command-line interface, type repimexp/help to view your usage options.

Using REPIMEXP, you cannot pick and choose which workflows to migrate. To choose a single workflow or only certain workflows for migration, use the Import/Export migration option.

Migrating With Application Deployment Manager

Application Deployment Manager (ADM) is a feature that automates the process of migrating enterprise customization data from one Siebel application environment to another, including from a development environment to a production environment. This customization data can include views, responsibilities, assignment rules, workflow processes, workflow policies, and so forth.

ADM is designed to provide a single deployment tool that covers various areas within the Siebel application. The objective is to reduce the potential manual setup and deployment work and provide as much automation as possible to decrease the error rate.

A workflow process deployment package in ADM includes the SRF, the workflow processes, their subprocesses, and their run-time settings, such as activation/expiration times, monitoring levels, and so forth. For information, see *Siebel Application Deployment Manager Guide*.

Migrating With the Workflow Admin Service Business Service

The Workflow Admin Service business service allows you to perform import/export, deployment, and activation on multiple workflow processes in bulk. Workflows are identified through a search specification. For more information see, "About the Workflow Admin Service Business Service" on page 384.

Migrating With Import/Export

You can use Import/Export for incremental migration of workflow process objects. You use Siebel Tools to export workflows from one environment and to import workflows to another environment.

The Workflow import/export feature is designed only to migrate an individual workflow process or a small set of workflow processes. For example, Workflow import/export cannot migrate 150 workflow processes at one time. To migrate large numbers of processes, break them into sets of 10 workflow processes or less.

Using Siebel Tools to Migrate a Workflow Process

Using Siebel Tools, you import the workflow process definition into the repository of the target environment, then you mark the workflows for migration by clicking the Publish button. After this, the definitions are ready to be activated.

This approach makes sure the versions of the workflow definitions that exist in the repository tables and the run-time tables are the same.

To use Siebel Tools to import a workflow process, you:

1. Export the workflow process to XML using the Import/Export Utility.
2. Import the workflow process definition into the repository of the target environment.
3. Activate the workflow process.

Figure 22 illustrates an incremental migration using the Import/Export utility and Siebel Tools.

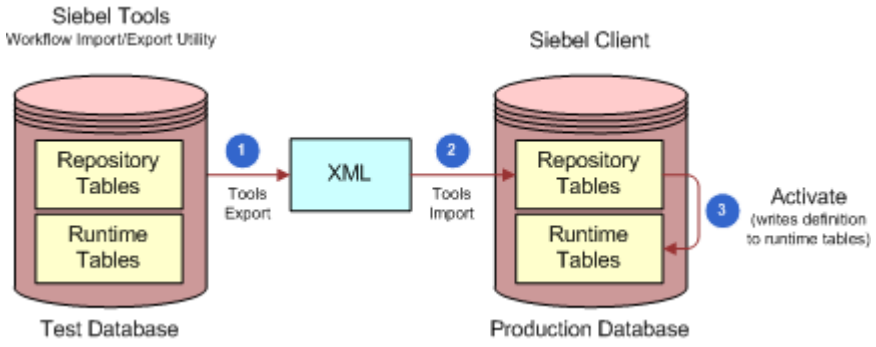


Figure 22. Illustration of Incremental Migration Using Import/Export From Within Siebel Tools

Using Siebel Tools In Conjunction With the Siebel Client to Migrate a Workflow Process

You can import a workflow process definition directly into the run-time tables. This approach bypasses the requirement for you to write the definitions into the repository tables of the target environment and activation from the Siebel client, although these steps are still performed behind the scenes by the Workflow engine. This approach causes the latest version of the workflow definition in the run-time tables, used by the Workflow engine, to be different from the version that resides in the repository tables.

NOTE: This is an effective technique for testing a workflow in a different environment. However, it is recommended that this technique not be used for general migration of workflows across environments.

To use the Siebel Client to migrate a workflow process, you:

1. Export the workflow process to XML using the Export Utility.
2. Import the workflow process definition into the repository of the target environment.

Figure 23 illustrates an incremental migration using Import/Export to export from within Siebel Tools and import from within the Siebel client.

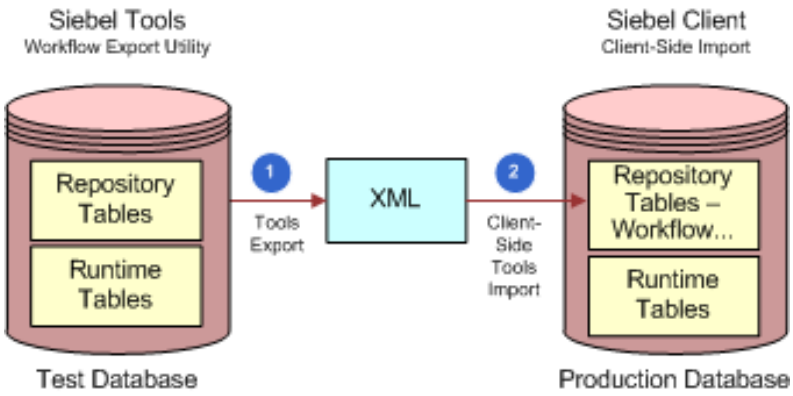


Figure 23. Illustration of Incremental Migration Using Import/Export from Within Siebel Tools and Import from Within the Siebel Client

Importing and Exporting a Workflow Subprocess

When importing a workflow process that contains subprocesses, you must first import the subprocesses and then the parent workflow process. Import the parent process only after the subprocesses are successfully imported. This rule also applies for importing workflow processes in batches.

It is not necessary to export subprocesses first when exporting a workflow process.

Name length must also be considered for subprocesses when importing a workflow process. A subprocess can have a name of up to 100 characters in length. A Subprocess name that exceeds this limit can trigger errors during import.

Importing and Exporting Carriage Returns With the Outbound Communications Manager Business Service

If a workflow process calls the SendMessage method of the Outbound Communications Manager business service to send email, and the message body uses carriage returns to format text display, the workflow process can be exported. However, when importing the workflow process back into the database, the imported workflow process will not contain the carriage returns. Instead, the carriage returns appear as square characters, and the message body is treated as a single paragraph.

To remedy this situation, you must edit the message body, replacing the square characters with carriage returns. A carriage return is entered by pressing the enter key.

Redeploying a Workflow After it is Migrated

When planning a migration strategy, it is important to consider potential redeployment actions that need to be taken after the workflow is migrated.

After migrating your workflow process to a production environment, it might be necessary for you to redeploy the workflow before you can run it. Table 58 describes the ways redeployment requirements depend on how the workflow is migrated.

Table 58. Comparison of Migrating With ADM Against REPIMEXP or Import/Export

Redeployment with ADM	Redeployment with REPIMEXP or Import/Export
If you used ADM to migrate workflows, you do not need to manually redeploy them in the target environment.	If you used REPIMEXP or Import/Export to migrate workflows, you must manually redeploy them in the target environment.

Migrating a Workflow Process

This task is a step in "Process of Migrating a Workflow Process" on page 243.

After you have developed a migration strategy, use the tool identified in the migration strategy to perform the migration. This topic provides procedural information for migrating workflow policies, and using the import/export option or the Workflow Admin Service business service options. For procedures for other strategies, see the relevant documentation.

Note that if you are migrating a large number of workflow processes, you can create a small group of processes to migrate as a first phase of implementation. After you have successfully migrated the first group, you can add more processes in a systematic manner.

Migrating Workflow Policies

For more information, see "Migrating Workflow Policies to the Production Environment" on page 313.

Importing and Exporting a Workflow Process

You can use the import/export feature both as a migration tool and as a way to back up individual an workflow process. Use a meaningful naming convention when choosing a filename for an exported workflow process to make it easy to understand the purpose of the process.

Using Workflow Import and Export to Back Up a Workflow Process

Siebel Tools features that are not applicable to Workflow objects include:

- SIF export and import.
- Object Compare.
- Three way merge during upgrades.

Because Siebel Tools excludes Workflow objects from these features, it is important to use the Workflow import and export feature to back up and restore a workflow object definition. For example, if you archive a project in Siebel Tools, the Workflow objects within that project are not archived.

CAUTION: If you delete the objects from a project expecting that they can be restored from the SIF, it is important to keep in mind that Workflow objects are an exception, and cannot be restored from the SIF. Use the Workflow import and export feature to back up and restore workflow definitions.

To export a workflow process

1. In Siebel Tools, in the Workflow Processes OBLE, choose the workflow process or processes you need to export.

   To choose more than one workflow process, press and hold the CTRL key while choosing the processes.

2. Right-click then choose Export Workflow Process.

   The Save As dialog box appears.

3. Enter the file path, filename, and the .xml filename extension, then click Save.

   The workflow process or processes are exported. If you chose more than one process to export, the processes you chose are saved to one XML file.

Note that when exporting a workflow process that contains subprocesses, you must also export the subprocesses. Subprocesses are not exported automatically.

To import a workflow process

1. In Siebel Tools, display the Workflow Processes OBLE.
2. Right-click then choose Import Workflow Process.

   The Open dialog box appears.

3. Choose a path and filename of the workflow process to import, then click Open.
4. Choose the project to which you need to add the workflow process.

   The workflow process is imported with a status of In Progress.

Note that if a process definition of the same name exists in the target environment, the newly imported process definition's version number is incremented by one.

Invoking the Workflow Admin Service Business Service

This topic provides procedural information on invoking the Workflow Admin Service business service. For more information, see "About the Workflow Admin Service Business Service" on page 384.

To invoke the Workflow Admin Service through the Business Service Simulator

1. In the Siebel client, navigate to Administration-Business Service > Simulator.
2. In the Simulator applet, click New, then define fields according to values described in the following table:

Field	Value
Service Name	Workflow Admin Service
Method Name	Activate

3. In the Input Arguments applet, click New, then define fields according to values described in the following table:

Field	Value
Test Case #	1
Property Name	FlowSearchSpec
Property Value	[Process Name] like '*Pricing Procedure*' and [Status] = 'Completed'

4. In the Simulator applet, click Run.

   Upon completion, the simulator displays the results of the simulation in the Output Arguments applet. In this example, the Property Value field in the Output Arguments applet displays the number of workflow processes whose name starts with Pricing Procedure and whose status is completed. You should modify the Property Value to meet your specific business requirements.

5 To examine simulation output, navigate to Administration-Business Process > Workflow Deployment, then issue a compound query using values described in the following table:

Field	Value
Name	*Pricing Procedure*
Deployment Status	Active

The query returns a list of workflow processes that were also represented in the Output Arguments applet of the business service simulator. The total number of records returned in the query should match the number of records displayed in the Property Value field described in Step 4.

Administering Workflow Processes in the Siebel Client

This topic includes the following topics:

About the Administration-Business Process Views

This topic provides descriptions for views found in the Administration-Business Process view.

Workflow Deployment View

The Workflow Deployment view allows you to deploy and activate a workflow process. This view displays the parent/child relationship between a workflow process and its run-time records. For detailed information on a chosen workflow process, drill down on the process name in the top applet to navigate to a form applet. To use this view, navigate to Administration-Business Process > Workflow Deployment.

Table 59 describes child views for the Workflow Deployment view.

Table 59. Description of Child Views of the Workflow Deployment View

View Tab Label	Description
Active Workflow Processes	This view lists workflow process definitions that have been deployed.
Child Items	This view filters records to display only child items of the process definition chosen in the parent list applet.

Workflow Instance Admin View

The Workflow Instance Admin view displays processes that are in running, waiting, error states, and that have persistence set. The view allows you to view, monitor and control workflow processes. For a chosen instance, you can change the value of the process properties before resuming the instance. To use this view, navigate to Administration-Business Process > Workflow Instance Admin.

Note that persistence for a version 7.0 workflow is set if the workflow process has a Wait step or the workflow's Auto Persist flag is checked.

Table 60 describes applets in the Workflow Admin View.

Table 60. Description of Applets in the Workflow Admin View

Applet Name	Description
All Workflow Process Instances	The top applet in this view lists workflow process instances that are running, in an error state, or in a waiting state.
Related Instances	This applet displays the associated processes for the chosen parent workflow process-subprocesses and error processes.
Process Properties	This applet displays the process properties for the process instance chosen in the Related Instances applet. You can change the value of these process properties before resuming an instance that is waiting or in an error state.

Workflow Instance Monitor View

The Workflow Instance Monitor view provides a log of workflow instances. The Workflow Instance Monitor allows you to monitor workflow process instances in most states, as well as step instances and aggregate data. To access this view, in the Siebel client navigate to Administration-Business Process > Workflow Instance Monitor.

Table 61 describes views available in the Workflow Instance Monitor.

Table 61. Description of Views in the Workflow Instance Monitor

View / Applet Name	Description
Workflow Process	The Workflow Process applet displays workflow definitions for workflow processes that have monitoring turned on, that is, the Monitoring Level is not NONE.
Process Instances	Displays the related log instances for the chosen workflow process.
Step Instances	Displays the steps and process properties for the chosen process instance.
Aggregate Data	Displays aggregate data as a chart view for the chosen workflow process.

About the Monitoring Level

When the Monitoring level is set for a deployed workflow process definition, the workflow process instance remains in the Workflow Instance Monitor view after it has finished and is no longer visible in the Workflow Instance Admin view.

Depending on the monitoring level set for the chosen workflow process, you might see no records in the Step Instances and Aggregate Data applets. For more information, see "Setting Workflow Process Monitoring Levels" on page 258.

Workflow Processes View

The Workflow Processes view allows you to review workflow processes defined prior to Siebel version 7.7. To use this view, navigate to Administration-Business Process > Workflow Processes.

Note that the Workflow Processes view and its child views are read-only views used specifically for reviewing workflow processes defined prior to version 7.7. You view new workflows that you create in the Workflow Processes OBLE in Siebel Tools.

Table 62 describes child views in the Workflow Processes view.

Table 62. Description of Child Views of the Workflow Processes View

View Tab Label	Description
Process Definition	This view allows you to view definitions for workflow processes defined prior to version 7.7.
Process Designer	This view allows you to view the process flow diagrams for workflow processes defined prior to version 7.7..
Process Properties	This view allows you to view the properties for workflow processes defined prior to version 7.7.

Administering Workflow Process Instances

This topic provides procedural information on administering workflow process instances. It includes the following topics:

Viewing Run-Time Instances of a Workflow Process

You can view a workflow's run-time instances in the Workflow Instance Admin view.

To view run-time instances for a workflow process

1. In the Siebel client, navigate to Administration-Business Process > Workflow Instance Admin.
2. In the All Workflow Process Instances applet, choose the workflow process you need to administer from the Process Name field.
3. In the Related Instances child applet, you can view the workflow's run-time instances and their parameters.

Stopping a Workflow Process Instance

From within the Workflow Instance Admin view, a Workflow administrator can stop a workflow process instance if persistence is defined. After a workflow process instance is stopped, it is removed. A process instance with a status of Running, Waiting, or Error can be stopped.

If you stop a running workflow process instance, the execution of the process instance is terminated. This is different from purging a workflow process instance from the monitoring log. For more information, see "Purging a Workflow Process Instance from the Log" on page 256.

CAUTION: A stopped workflow process instance cannot be resumed.

To stop a workflow process instance

1. In the Siebel client, navigate to Administration-Business Process > Workflow Instance Admin.
2. In the All Workflow Process Instances applet, choose the workflow process you need to stop from the Process Name field.
3. In the Related Instances applet, choose the process instance you need to stop.
4. In the applet menu, choose Stop Instance.

   For caution information, see "Stopping a Workflow Process Instance" on page 254.

Stopping Workflow Process Instances for a Workflow Process

You can stop workflow process instances of a specific workflow process definition in the Workflow Deployment view.

If you know the specific instance you need to stop, stop the workflow process from the Workflow Instance Admin view. For more information, see "Stopping a Workflow Process Instance" on page 254.

If you need to stop process instances of a specific version of a workflow process definition, then you can delete the process definition from within the Workflow Deployment view. Note that deleting a process definition from the Workflow Deployment view does not only remove a process definition from the run time. It also stops process instances of the deleted process definition.

To stop process instances of a workflow process definition

1. In the Siebel client, navigate to Administration-Business Process > Workflow Deployment > Active Workflow Processes.
2. Choose the process instance you need to delete.
3. From the applet menu, choose Delete Process.

TIP: Using Multi-select, you can choose multiple workflow process definitions. Because the Delete Process functionality deletes a process definition and stops the process instances of that definition,. Process instances related to the chosen definitions are also stopped as a result of the deletion. So, if you need to perform a batch delete of multiple workflow processes, follow the steps outlined in the above procedure, using Multi-select to choose multiple workflow processes rather than just one. To use Multi-select, keep the CTRL key depressed as you click each record in the applet.

Stopping a Workflow Process Instance from Script

Using the _StopInstance method, you can stop a workflow process instance using script. An example usage of this technique occurs when you have an interactive workflow which needs to be cancelled but which is suspended in a Wait step. In this scenario, the Process Instance Id is already known.

To stop a workflow process instance from script

Invoke the _StopInstance method on the Workflow Process Manager business service, as in the following example, which uses a hard-coded Process Instance Id:

```
var bs_WF     = TheApplication().GetService("Workflow Process Manager");

var ps_inputs    = TheApplication().NewPropertySet();

var ps_outputs    = TheApplication().NewPropertySet();

ps_inputs.SetProperty("ProcessInstanceId", "1-IIT");

bs_WF.InvokeMethod("_StopInstance" , ps_inputs, ps_outputs);
```

Deactivating a Workflow Process Instance

You can deactivate a single instance of a workflow process, or multiple process instances in one batch.

To deactivate a workflow process instance

1. In the Siebel client, navigate to Administration-Business Process > Workflow Deployment > Active Workflow Processes.
2. Choose the process instance you need to deactivate.
3. From the applet menu, choose Deactivate Process.

   The status of the chosen process instance for the workflow process is changed to Inactive.

To deactivate workflow process instances in batch

1. In the Siebel client, navigate to Administration-Business Process > Workflow Deployment > Active Workflow Processes.
2. Choose the process instances you need to deactivate.

   Use Multi-click, holding down the CTRL key while clicking each record in the applet.

   TIP: Instead of using Multi-click, formulate a query to display only the workflow processes you need to deactivate, then choose Select all from the applet menu.
3. From the applet menu, choose Deactivate Process.

   The status of chosen process instances for the workflow process is changed to Inactive.

Expiring a Workflow Process Instance

This administrative procedure is performed in Siebel Tools, not in the Siebel client.

To expire a workflow process instance

1. In the Workflow Processes OLBE, choose the workflow process you need to expire.
2. From the application-level menu, choose Tools > Lock Project.

   This action locks the project defined in the Project property for the currently chosen object definition in the OBLE.
3. In the WF/Task Editor toolbar, click the Expire button.

   For more information, see "About the WF/Task Editor Toolbar" on page 72.

Purging a Workflow Process Instance from the Log

You can use the purge feature to delete process instances that have a status of Stopped or Completed before the user-specified date. If you need to delete a paused instance, stop the instance first.

Note that if you delete a running workflow process instance from the log, the execution of the process instance is unaffected and the workflow process continues to run.

To purge process instances from the log

1. In the Siebel client, navigate to Administration-Business Process > Workflow Instance Monitor > Process Instances.
2. In the Process Instances applet, choose the workflow process instance you need to purge.
3. Click the Purge button.
4. In the Workflow Instance Monitor Purge dialog box, choose a date.
5. Click Purge.

Description of Where Workflow Process Instance Information is Stored

Tables in which workflow process instances are stored include:

- S_WFA_INSTANCE
- S_WFA_INST_PROP
- S_WFA_INST_WAIT

Tables in which workflow instance monitoring information is stored include:

- S_WFA_INST_LOG
- S_WFA_INSTP_LOG
- S_WFA_STPRP_LOG

Monitoring Workflow Processes in a Production Environment

This topic describes how you can monitor and troubleshoot a workflow process in the production environment. It includes the following topics:

Overview of Monitoring and Troubleshooting Tools

Tools that can be used to monitor and troubleshoot include:

- **Progress and status information.** Use the Workflow Instance Monitor view and the Workflow Instance Admin view to monitor the progress and status of each process instance.
- **Operation details.** Use the event logging facility to trace operations performed by each processing step of a Workflow application.
- **Performance-measurement data.** Use Siebel ARM to capture timing data for measuring performance of Workflow applications.
- **Failure-analysis records.** In case of a system failure, Siebel FDR (Flight Data Recorder) automatically captures the events that might have led to a system failure.

You monitor and control workflow process execution in the Administration-Business Process views of the Siebel client. You can monitor and troubleshoot Siebel Workflow in the production environment, where you can view progress and status information, operation details, performance measurement data, and failure analysis records.

You can diagnose a production workflow process with one or a combination of the tools described in this topic. For more information, see "Diagnosing a Failed Workflow Process in a Production Environment" on page 262.

Setting Workflow Process Monitoring Levels

The Monitoring Level you set determines the frequency with which log writing occurs and, hence, the data available in the workflow instance views.

There are two views you can use to monitor workflow processes. For more information, see "Workflow Instance Admin View" on page 252 and "Workflow Instance Monitor View" on page 252.

Setting Monitoring Level Parameters

When the workflow instance is created, the monitoring level is read from the workflow process definition and remains throughout the lifetime of the instance unless the instance is paused. In the case of an instance being paused, when the instance is resumed, the monitor level is reread from the definition.

Table 63 describes monitoring level parameters and their corresponding log writing frequencies you can set for a workflow process.

Table 63. Descriptions of Monitoring Level Parameters

Monitoring Level	Record Process Instance	Record Step Instances	Record Process Properties
0 None	No	None	None
1 Status	Yes	None	None
2 Progress	Yes	All Steps	None
3 Detail	Yes	All Steps	All Steps
4 Debug	Yes	All Steps	All Steps

To set the monitoring level parameter

1. In the Siebel client, navigate to Administration-Business Process > Workflow Deployment.
2. In the Active Workflow Processes applet, set the Monitoring Level field.

Considering the Impact of Monitoring Levels on Performance

The monitoring level determines the frequency at which data is written to the disk. The frequency is optimized internally by the Workflow run-time environment, based on the workflow type and the monitoring level you choose.

Configuring monitoring at any level incurs performance overhead to your workflow processes. It is best to set the monitoring level to 0 (None) or 1 (Status) on workflow processes running in production.

NOTE: It is recommended that monitoring levels of 2 (Progress) and higher only be used for debugging workflow processes.

At the Debug monitoring level, the log is written to the disk after every step.

Monitoring Levels and Version 7.0 Compatibility

Version 7.0 workflows with persistence frequency and persistence level set are mapped to a monitoring level based on the following logic:

- If the persistence frequency is set to NONE, then the monitoring level is set to NONE.
- If the persistence frequency is ON_PAUSE or EVERY_STEP, persistence is explicitly turned on and the monitoring level is set:
  - If the persistence level is ALL_STEPS, then the monitoring level is set to PROGRESS.
  - If the persistence level is CURRENT_STATE, then the monitoring level is set to STATUS.

In this release, persistence and monitoring are separate features that serve different purposes. Persistence is a quality of service and is controlled at definition time. Monitoring is an administrative tool and is controlled at deployment time. Monitoring typically does not impact workflow process functionality.

Setting Tracing and Event Log Levels

You can use tracing levels to troubleshoot a workflow process. Note that setting a trace level above the default level affects performance. Reset the trace level to the default value after troubleshooting is finished.

Table 64 describes events Siebel workflow uses for logging.

Table 64. Description of Workflow Process Logging Events

Event	Level	Description
Workflow Definition Loading (DfnLoad)	3	Traces process and step definitions loaded into memory.
Workflow Engine Invoked (EngInv)	4	Trace methods invoked and arguments passed to the Workflow engine.
Workflow Process Execution (PrcExec)	4	Traces process instance creation and completion. Traces the process property get/set.
Workflow Step Execution (StpExec)	4	Traces step creation and completion, branch condition evaluation, business service invocation, and business component insert/update.

Table 64. Description of Workflow Process Logging Events

Event	Level	Description
Workflow Performance (WfPerf)	4	Measure overall process execution time.
Workflow Performance (WfPerf)	5	Measure process and step execution time.
Workflow Recovery (WfRecv)	3	Trace instance recovery status and progress. Applicable only to the Workflow Recovery Manager server component.
Workflow Recovery (WfRecv)	4	Trace instance recovery details. Applicable only to the Workflow Recovery Manager server component.

Increasing Tracing Levels for Workflow Management Server Components

You can generate a more detailed trace file to assist in troubleshooting Workflow Process Manager, Workflow Process Batch Manager, and Workflow Recovery Manager.

Increase tracking levels prior to executing the server process.

To increase tracing levels

1. In the Siebel client, navigate to Administration-Server Configuration > Servers > Components > Events.
2. In the Components applet, choose the component for which you need to generate tracing.

   These components include the Workflow Process Manager, the Workflow Process Batch Manager, or the Workflow Recovery Manager.
3. Click the Events tab to view the configurable event types for the chosen component.

   The log level is set to a default value of 1.
4. Change the log level value to 3, 4, or 5.

   For more information, see Table 64 on page 259.
5. (Optional) For more troubleshooting, you can repeat Step 4 to increase the tracing level for the Object Manager SQL log event.

More tracing information is generated as the Component Event Configuration Log Level value increases. For more information on the different Log Level values available for Component Event Configuration, see *Siebel System Monitoring and Diagnostics Guide*.

Capturing Data with Siebel Application Response Management

The Workflow Process Manager server component can use Siebel Application Response Measurement (Siebel ARM). Siebel ARM captures timing data useful for monitoring the performance of the Siebel application, and records this information to binary files.

Table 65 describes Siebel ARM levels.

Table 65. General Description for Siebel ARM Levels Used With Siebel Workflow

Siebel ARM Level	Description
1	The Workflow Process Manager business service records the time it takes to execute a service method. Examples of service methods in Workflow Process Manager are RunProcess and ResumeInstance.
2	The Workflow Process Manager business service records the time it takes to: ■ Execute a service method ■ Execute a workflow step ■ Write monitoring data to the disk Siebel ARM level 2 can help you determine the logging overhead when you increase the monitoring level of your workflow process.

Table 66 describes Siebel ARM levels and their Siebel ARM areas and subareas that are defined for Siebel workflow.

Table 66. Detailed Description for Siebel ARM Levels Used With Siebel Workflow

Level	Area	Subarea	Description
1	WORKFLOW	CORDR_RESUME	Resume a suspended process.
1	WORKFLOW	CORDR_EXECUTE	Execute a process.
1	WORKFLOW	ENGNE_INVOKE	Invoke a Workflow Process Manager service method.
2	WORKFLOW	STEPS_EXSTEP	Execute a step.
2	WORKFLOW	MONTR_WRTE	Write process monitoring data to disk.

For more information on configuring Siebel ARM and the Siebel ARM analyzer, see *Siebel Performance Tuning Guide*.

Recording Behavior with Siebel Flight Data Recorder

Siebel flight data recorder log files, identified with the .fdr extension, are records of system and server component behavior at run time. In the event of a system or server component failure, the settings and events leading up to the failure are captured and logged. The Siebel flight data recorder log file can then be used to troubleshoot and analyze the specific settings and events that occurred prior to the failure. The Siebel flight data recorder log files are stored in the Binary subdirectory of the Siebel Server root directory.

FDR instrumentation points have been embedded in the Workflow Process Manager business service and the Workflow Recovery Manager business service to provide capture-processing details in case of a system failure or server component failure.

For information about getting help with the log file, see "Getting Help With A Workflow Error" on page 264. For more information on Siebel flight data recorder files, see *Siebel System Monitoring and Diagnostics Guide*.

Diagnosing a Failed Workflow Process in a Production Environment

This topic includes following topics:

Diagnosing a Failed Workflow Process

This topic includes procedural information to diagnose problems in the production environment:

- Diagnose a workflow process using tracing.
- Diagnose a workflow process using instance monitoring.
- Diagnose a workflow process using the Business Service Simulator.

In addition, the following topics are included:

For reference information on some of the tools described in this topic, see "Monitoring Workflow Processes in a Production Environment" on page 257

To diagnose a workflow process using tracing

1 Turn on tracing for the appropriate component running the workflow: Workflow Process Manager, Workflow Process Batch Manager, or the application Object Manager.

2 View the event log files.

For details on how to turn on tracing, see "Setting Tracing and Event Log Levels" on page 259.

To diagnose a workflow process using instance monitoring

1 In the Siebel client, navigate to Administration-Business Process > Workflow Deployment.

2 In the Active Workflow Processes applet, choose the workflow process you need to monitor.

3 Set the monitoring level to 3-Detail or 4-Debug.

At this level, each execution of a process, from whichever component it is run, records state and process properties values for each step.

NOTE: The 3-Detail and 4-Debug monitoring levels affect performance of the Workflow engine. For this reason, use these levels for troubleshooting purposes only.

4 Navigate to Administration-Business Process > Workflow Instance Monitor > Process Instances.

5 View the monitoring information and take corrective action.

To diagnose a workflow process using the Business Service Simulator

1 Run the workflow process from the Business Service Simulator using the Workflow Process Manager business service.

This step executes the workflow process in the application object manager.

2 In the Siebel client, navigate to Administration-Business Service > Simulator.

3 In the Simulator applet, create a new record then set the fields using values described in the following table:

Field	Value
Service Name	Workflow Process Manager
Method Name	RunProcess
Iterations	1

4 In the Input Arguments applet, create a new record, and perform the following procedure:

a Set the Test Case # field to 1.

b Choose and open the Property Name field.

The Property Name field opens a multi-value applet.

5 In the multi-value applet, click New then set the fields using values described in the following table:

Field	Value
Property Name	ProcessName
Value	(Enter the name of the workflow process)

6 Click Save.

7 Repeat Step 5 and Step 6 for other parameters passed to the process, especially RowId, if necessary.

8 In the multi-value applet, click OK.

9 In the Simulator applet of the Simulator view, click Run.

TIP: To increase data available to you for troubleshooting, set the monitoring level to 4-Debug, as described in Step 3 on page 263, launch the workflow with the Business Service Simulator, then view process execution information in the Workflow Instance Monitor views.

For more information, see "About the Business Service Simulator" on page 229.

Disabling Persistence to Avoid Excessive Records in S_WF_PROP_VAL

S_WF_PROP_VAL stores process property values for workflow processes. Records are created in S_WF_PROP_VAL along with a new S_WF_STEP_INST record when a workflow process is executed.

There is the potential for S_WF_PROP_VAL to become very large over time, since a workflow process definition typically contains five or more process properties. Therefore, five records can be added to S_WF_PROP_VAL for a given process instance.

Having Persistence defined on a large number of workflows can cause an increase in the size of S_WF_PROP_VAL. To avoid this, review workflow process definitions and disable persistence unless it is absolutely necessary. Persistence can be disabled by setting the Auto Persist property to NO on the workflow process object definition in the OBLE. For more information, see "About Events" on page 150.

Getting Help With A Workflow Error

For help with a workflow error, create a service request (SR) on OracleMetaLink 3. Alternatively, you can phone Global Customer Support directly to create a service request or get a status update on your current SR. Support phone numbers remain the same and are listed on OracleMetaLink 3.

The ways you can communicate workflow errors include:

- Send the log files.

  Generated by turning on tracing, as described in Step 1 on page 263.

- Communicate the error code and error message.

  Except for service flows, when a workflow process encounters an error, the process state is persisted with a status of In-Error. If a workflow process encounters an error in one of the workflow's subprocesses, the subprocess state is also persisted. Both the error code and the error message are saved in the process properties. An administrator can examine the error codes and error messages in the Process Properties applet of the Workflow Process Instance Admin view. When a record is chosen from the All Workflow Process Instances list, the related instance applet lists its parent and child processes. You can communicate the error code and the error message for further assistance.

Troubleshooting Workflow Process Execution Problems

This topic describes guidelines for resolving workflow process execution problems.

Table 67 lists Symptoms/Error messages you can review to resolve problems with workflow process execution.

Table 67. Description of Troubleshooting Workflow Process Execution Problems

Symptom	Diagnostic Steps/Cause	Solution
After activating a workflow, it is not executing when the corresponding run-time event is triggered.	(This cell is intentionally empty.)	Make sure *Reload Runtime Events* has executed. To tell if a workflow process is triggered, turn on workflow logging for EngInv, StpExec, and PrcExec. For more information, see "Increasing Tracing Levels for Workflow Management Server Components" on page 260.
After revising a workflow and reactivating it, the revised workflow information has not been read.	(This cell is intentionally empty.)	For workflows running in the Workflow Process Manager server component, reset the parameter *Workflow Version Checking Interval*. By default the interval is 60 minutes.

Table 67. Description of Troubleshooting Workflow Process Execution Problems

Symptom	Diagnostic Steps/Cause	Solution
When a workflow is triggered by the run-time event DisplayApplet, the workflow is triggered the first time but not subsequently.	Since the DisplayApplet event is a UI event, and the default Web UI framework design uses a cache, the event only fired when the first time non-cached view is accessed. The workflow was triggered only when the event fired and worked correctly.	To make the field still work in this scenario, you can explicitly set EnableViewCache to FALSE in the .cfg file.
After triggering a workflow from a run-time event, the row ID of the record on which the event occurred is not retrieved.	The run-time event passes the row ID of the primary business component and not the row ID of the business component on which the run-time event is acting.	Retrieve the row ID of the active business component using search specifications. For example: the Active_row-id process property is equal to [Id], defined as Type equals Expression and the business component equals the business component name.
The following error message is displayed: *Cannot resume Process [x-xxxxx] for Object-id [x-xxxxx]. Please verify that the process exists and has a waiting status.*	This error typically occurs in the following scenario: 1 A workflow instance is started and paused, waiting for a run-time event. 2 The run-time event fires. The workflow instance is resumed and run to completion. 3 The run-time event fires for a second time. The Workflow engine tries to resume the workflow instance and fails, since the workflow instance is no longer in a Waiting state.	Deleting existing instances does not help. Ignore the error message and proceed. Step 1 through Step 3 must occur, in that order, in the same user session for the error message to be reported. As a result, the error message disappears when the application is restarted. The Purge feature only works on stopped and finished instances. To delete persisted or incomplete instances, you first must manually stop the instances.
Unable to access a different business object from a workflow process.	The Workflow architecture restricts the use of one business object to a workflow.	Use a Sub Process step to access a different business object.

Table 67. Description of Troubleshooting Workflow Process Execution Problems

Symptom	Diagnostic Steps/Cause	Solution
The following error message is displayed: *Unable to initiate the process definition [process name].*	(This cell is intentionally empty.)	Make sure the following conditions are true: ■ The workflow process exists. ■ The process status is set to Active. ■ The process has not expired.
One of the following error messages is displayed: ■ OMS-00107: (: 0) error code = 4300107, system error = 27869, msg1 = Could not find 'Class' named 'Test Order Part A' ■ OMS-00107: (: 0) error code = 4300107, system error = 28078, msg1 = Unable to create the Business Service 'Test Order Part A'	(This cell is intentionally empty.)	Make sure at least one .SRF is copied to the Siebel Server\objects\[lang] directory.
The following error message is displayed: ■ *The selected record has been modified by another user since it was retrieved.*	A workflow process attempted to update a record that was updated by another user or task since the record was initially retrieved by the workflow process.	Define an error exception to handle the update conflict. For more information, see "Defining an Error Exception to Handle an Update Conflict" on page 159.

About the Workflow Recovery Manager

This topic describes Workflow Recovery Manager. It includes the following topics:

Workflow Recovery Manager Overview

Functionality provided by Workflow Recovery Manager includes:

- Recovers interrupted long running workflow instances upon Siebel Server failure. As the workflow instance is recovered, the workflow engine attempts to continue forward execution. If forward progress is not possible, the workflow is marked as IN_ERROR and manual intervention is required to resume the workflow.
- A workflow instance is resumed at the last checkpoint to maintain the integrity of the execution. The checkpoint is automatically determined by the workflow engine based on hints provided by the developer during design time. The workflow engine automatically persists relevant execution data to resume execution upon workflow process failure.
- Recovery of workflow process instances is load-balanced. One server thread can be responsible for determining the recovered instances. This server thread delegates the actual resumption of the recovered instance to other server threads in a load-balanced manner.
- Workflow recovery can be started without interfering normal execution of the workflow engine. One server thread can start the recovery of workflow instances, while other threads in the workflow engine still handle new requests to execute workflows. Thus, the workflow engine is blocked by the recovery thread.

Workflow Recovery Manager Architecture

Figure 24 illustrates the architecture used for recovering a workflow process.

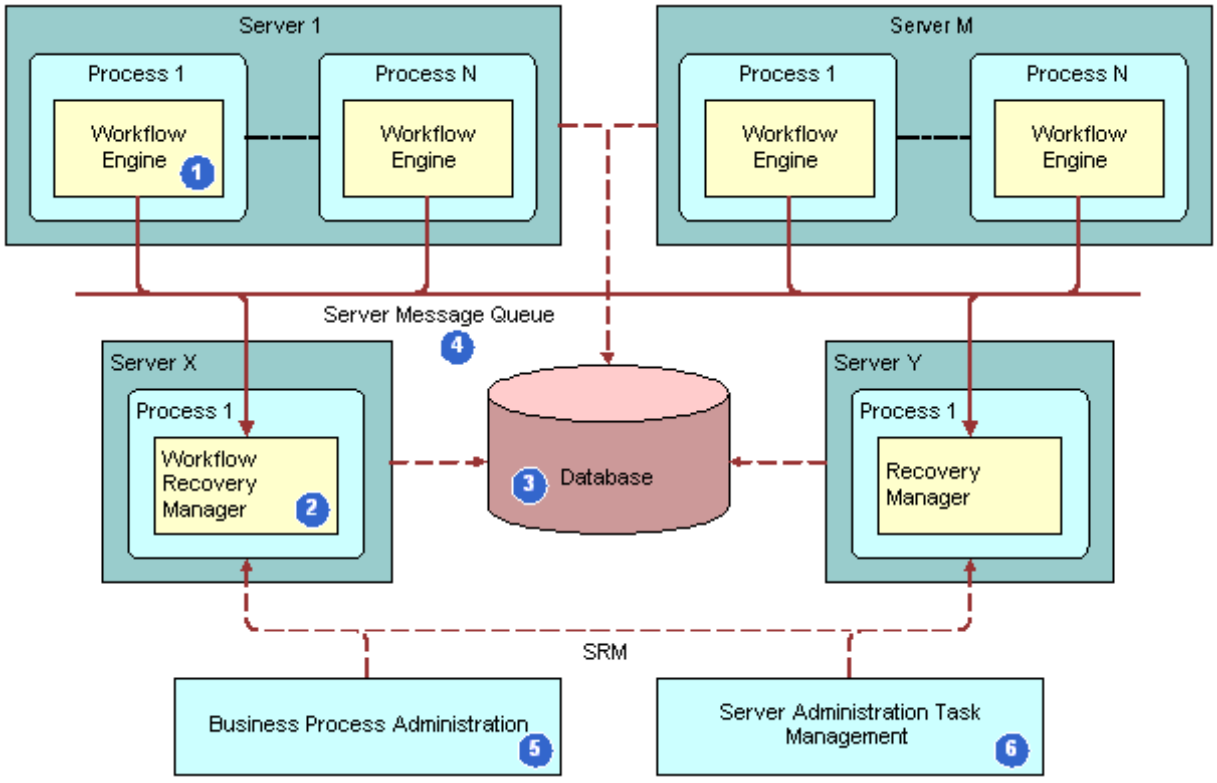


Figure 24. Workflow Recovery Manager Architecture

The Workflow Recovery Manager architecture consists of the following components:

1. **Workflow Engine.** A batch component, WfProcMgr, responsible for executing long running workflows. As the workflow is executed, the workflow engine persists the execution state into the database at the appropriate time.
2. **Workflow Recovery Manager.** A batch component responsible for identifying interrupted workflows due to server failure. When Workflow Recovery Manager discovers an interrupted workflow instance, the recovery manager forwards the workflow instance to a workflow engine to resume its execution. The recovery manager itself does not execute the workflow instance.
3. **Database.** A database used for storing the execution state of the workflows. The persisted record is used to restore execution state when the workflow process resumes execution from failure.
4. **Server Message Queue.** The workflow engines and the recovery managers multi-cast messages through the Server Message Queue.
5. **Business Process Administration View.** Allows the administrator to manually request the Workflow Recovery Manager to recover an interrupted workflow.

6 **Server Administration Task Management.** Allows the system administrator to create a recurrence task for the Workflow Recovery Manager. The recurrence task requests the server manager to periodically scan for interrupted workflows.

Administering Workflow Recovery Manager

The status of the WfRecvMgr component is Online once it is started, functioning similar to other components. However no tasks are visible unless a recovered workflow instance is recovered after a server failure.

If you run *list tasks for comp WfRecvMgr* and no executed tasks are returned, it indicates there are no failed workflow process instances. This is expected behavior.

WfRecvMgr in an Active status indicates the WfRecvMgr component is up and running.

You can start the Workflow Recovery Manager in the Siebel client.

To start the Workflow Recovery Manager

1 In the Server Manager command-line interface, issue the `enable compgrp workflow` command.

This command makes sure the Workflow component group is activated on the server. For more information about Server Manager usage, see *Siebel System Administration Guide*.

2 In the Siebel client, navigate to Administration-Server Management > Components.

3 Query for Workflow Recovery Manager.

4 If Workflow Recovery Manager is not Online, click the Start Up button.

This action starts the Workflow Recovery manager. It is not necessary to set other parameters.

Testing Workflow Recovery Manager Execution

You can test Workflow Recovery Manager execution.

To prepare for testing of Workflow Recovery Manager execution

1 Connect to Server Manager from the Siebel Server.

You should see the `smgr>` prompt. For more information about Server Manager usage, see *Siebel System Administration Guide*.

2 Issue the following command and check whether the Component Group named Workflow is activated:

```
Smgr>list compgrp
```

3 Perform the following steps if the Workflow component group is not activated:

a Enter the following command to activate it:

```
Smgr> enable compgrp workflow
```

b Stop then restart the server.

4 In the Siebel client, navigate to Administration-Workflow Process > Workflow Deployment.

5 Make sure there is a workflow running that fails if the server is reset.

For Workflow Recovery Manager to attempt recovery, there must be an instance of an active, running workflow process that fails if it is interrupted. Peruse the Active Workflow Processes applet to make sure such a workflow is present.

To test Workflow Recovery Manager execution

1 Open a Telnet session connected to the Siebel Server, issue the command `siebps`, then make a note of the Process Id of the Siebel Server.

If you are using Windows, use Windows Task Manager instead of Telnet.

2 Open a second Telnet session then connect to the Server Manager prompt.

3 From the `smgr>` prompt, issue the following command to start the execution of the workflow:

```
Start task for comp wfprocmgr with processname = 'AutoRec_NN'
```

A new task for the component `wfprocmgr` with the task id is started.

4 From the first telnet session you opened, check whether the file `Inloop.txt` is created.

Note that numeric values are added to the file over time.

5 Issue the following command to reset the Siebel Server:

```
Reset_server -e <enterprise> -M <server name>
```

This causes the workflow you are monitoring to fail.

6 Restart the Siebel Server then enter the `siebps` command as you did in Step 1.

The Siebel Server process is started again.

7 Connect back to the `smgr>` prompt then issue the following command to resume execution of the workflow from the point of failure:

```
smgr>start task for comp wfrecvmgr
```

A new task for the component wfrecvmgr is started with a new task id.

8 From the Telnet session enter the following command to check the contents of the file:

```
tail -f inloop.txt
```

Note that numeric values are being added to the file until 2999 is reached.

9 Issue the following command to check the last value entered on to the file:

```
outfile.txt
```

The last and only value in this file should be 3000.

10 Run `list tasks for comp WfRecvMgr`.

Observe that there are now one or more tasks in the list. These are tasks Workflow Recovery Manager has taken to recover workflow processes that were interrupted when you reset the server in Step 5.

Recommended Workflow Recovery Techniques

It is recommended you follow certain techniques to make workflow recovery successful. These techniques include:

- Modify the Allow Retry Flag property on Siebel Operation steps and Business Service steps to reduce the number of checkpoints, and thereby minimizing run-time overhead. If the Recovery Manager cannot determine from which step the workflow instance must recover, then those instances are marked for manual recovery. Recovery is performed by the Recovery Manager based on the process instance's state information that is saved by the Workflow engine. The state information is saved at recovery checkpoints. For performance optimization, the recovery checkpoints are determined by the Workflow engine based on the nature of the step and the Allow Retry flag property on a workflow step.
- Break down a complex business service into a number of simpler business services. This makes it easier to individually recover each smaller business service.
- Use multiple recovery managers. Having multiple recovery managers provides a means to safe-guard against transient or permanent system problems. For example, such a condition can exist in an environment where the subnet in which the recovery manager resides is frequently down. However, it is typical to have only one recovery manager as long as the recovery manager itself can be automatically restarted upon server failure.

Recovering a Workflow Process

If the Workflow Process Manager server component fails, the workflow process automatically resumes the interrupted workflow instances when the server restarts. Recovery is performed by the Recovery Manager based on the process instance's state information that is saved by the Workflow engine.

You can recover an interrupted workflow process either automatically or manually.

Automatic Recovery of a Workflow Process Instance

If the Workflow Process Manager server component fails due to an event that occurs outside of the Workflow Process Manager server component, such as a server failure, Siebel workflow automatically resumes the interrupted workflow instances when the server restarts.

For a workflow process instance that cannot be automatically recovered, you can manually recover the process. For example, if the server fails in the middle of a Siebel operation to update a record, then the workflow is unable to determine if the Siebel Operation has finished. You might need to manually make sure the update was finished before resuming the workflow execution. In another case, if the Siebel operation queries a set of records, then even after the server fails, the workflow can be resumed automatically by requerying.

Automatic recovery of a workflow process applies to a workflow process that runs in the server component. A workflow process running on a local database cannot be recovered.

Manual Recovery of a Workflow Process Instance

You can correct and resume a workflow process instance that has encountered errors. For example, if the Communications Server is not available, a workflow process sending an email notification will have a status of In Error. You can activate the Communication Server component, then resume the workflow process.

Instances marked for manual recovery are recoverable from the Workflow Instance Admin view. Navigate to Administration-Business Process > Workflow Instance Admin > Related Instances, then choose an option from the applet menu. Options for manual recovery include:

- **Resume Instance-Next Step.** The process instance skips the current step, and evaluates the branching conditions coming from the current step to determine the next step to execute.
- **Resume Instance-Current Step.** The process instance resumes from the current step. The current step is retried. If the current step is a Sub Process step, a new subprocess instance is started.

To manually recover a workflow process instance, you use the Workflow Instance Admin view. For more information, see "Workflow Instance Admin View" on page 252.

Resuming a Workflow Process Instance

A workflow process instance can resume only if the workflow's Call Depth setting is the highest among the workflow's related process instances. Resume Instance-Next Step and Resume Instance-Current Step are disabled if the workflow instance is part of a set of related instance and one or more of those other instances has a higher Call Depth level.

For example, assume there are multiple related instances with Call Depth settings of level 0,1,2,3, and 4. Assume you choose the record with level 3. In this case, these Resume controls are disabled because level 3 is not the highest Call Depth level set among the related instances.

Upgrading Siebel Workflow

This topic describes upgrading Siebel Workflow from prior versions.

Siebel Database Upgrade Guide is the primary source for upgrade information. For important upgrade information that pertains to workflow, such as Workflow Premerge, Repository Merge, Workflow Postmerge, and Logging, see the *Siebel Database Upgrade Guide*.

Run-Time and Repository Schema Upgrade

If there have been repository and run-time schema changes for the Workflow tables, or if new columns have been added to the repository and run-time schema, then you need to add the new columns to the Siebel database. Follow the instructions provided in *Siebel Database Upgrade Guide*.

Existing workflow definitions in the run-time deployment table are not changed during the upgrade. After the merge process is finished, use the Workflow Deployment view in the Siebel client to reactivate finished workflows to load the new workflow definitions into the run-time environment. For instructions on reactivating a workflow process, see *Siebel Database Upgrade Guide*.

Workflow Definition Merge

If you have upgraded to Siebel version 7.7, Siebel workflow definitions were moved from the run-time tables into the repository tables. If you are upgrading to version 8.0 from version 7.7 or version 7.8, a modified seed workflow or a custom workflow is merged into the version 8.0 repository during the repository merge process.

However, if you are upgrading to Siebel version 8.0 directly from a version prior to version 7.7, your custom workflows and modified seed workflows are copied over to the version 8.0 repository *as is*. You must manually merge your customizations with the new version 8.0 seed workflows.

In either instance, make sure you review your workflow customizations after the repository merge process. For more information, see "Considering a Seed Workflow Process" on page 53. For information about detailed merge algorithms, see *Siebel Database Upgrade Guide*.

Workflow Mode Requirements for an Upgraded Workflow Process

When modifying a version 7.0 workflow process that has been upgraded to version 7.7 or later, the workflow's mode can remain the same.

12 Administering Workflow Policies

This chapter describes concepts and procedures for administering workflow policies. It includes the following topics:

About Workflow Policy Administration

This topic describes administration tasks relevant to Workflow Policies. It includes the following topics:

Confirming Workflow Policy Installation

In this topic you make sure workflow policies are installed correctly.

Workflow Policies is installed as part of the Siebel Server and client installation and is activated by using your license key information. This topic describes only how to make sure Workflow Policies are correctly installed. For information about the installation process, see the installation guide for the operating system you are using.

To run Workflow Policies make sure Siebel Server components, including Workflow Management as well as Siebel Tools and the Siebel client, are installed.

Confirming the Repository Setting for Workflow Policies Installation

In the Siebel client, the .cfg file is used to configure Workflow Policies. You should make sure the DockRepositoryName entry specifies the correct repository name.

Confirming the Workflow Setup for Workflow Policies Installation

You must make sure that your license key includes Siebel workflow.

You also must make sure the Siebel Server is installed properly since Siebel workflow runs as a server component on the Siebel Server.

To confirm the workflow setup

1. Log in to the Siebel client as the Siebel administrator.
2. Navigate to Administration-Business Process.

   Under the list of views, Policies, Policy Groups, and so forth are visible. This indicates your license key is correct.
3. From a Siebel client that is configured to manage the server component groups, navigate to Administration-Server Management > Servers > Component Groups.
4. Make sure the Workflow Management component group is activated.

Administering Triggers on the Workflow Policy Server

This topic describes how to administer triggers on the workflow policy server. It includes the following topics:

Creating Database Triggers

The Generate Trigger (GenTrig) component on the Siebel Server allows you to create database triggers. Workflow Policies uses database triggers to identify which records match policy conditions. Run Generate Triggers when you:

- Create or delete new policies, including Assignment Policies, except for workflow policies that have Batch Flag set to TRUE.
- Amend policy conditions or policy criteria.

- Change activation or expiration dates of policies, including Assignment Policies.

To run Generate Triggers, you must have installed the Siebel Server, and the Siebel client you are using must be configured to access the Siebel Server Administration screens. For more information on installing Server Manager, see the installation guide for the operating system you are using.

CAUTION: If you have incorrectly defined a policy condition, running Generate Triggers can result in invalid triggers. An invalid trigger can prevent execution of normal user transactions. For this reason, thoroughly test your policies in your test environment before you migrate them to your production system.

Description of Generate Triggers Execution

Table 68 describes how generating triggers works, depending on how the EXEC parameter is set.

Table 68. Description of Steps Taken to Generate Triggers, Depending on the Value of EXEC

Execution When EXEC is True	Execution When EXEC is False
Generate Trigger component automatically creates the SQL script and applies it to the server database	1 Use the Generate Triggers component from a Siebel Server to create the SQL script file, which is placed in the root directory of the Siebel Server installation. 2 Use your database vendor's SQL tool to execute the SQL script file against the server database.

The default setting for EXEC is FALSE.

You can run the Generate Triggers component from the Server Manager graphical user interface or command-line mode. Both the GUI and the command-line use the same parameters. The triggers are only there to create indicators for the Workflow engine to check the policies' conditions.

Example of GenTrig Behavior for a Workflow Policy With Multiple Conditions

When two or more conditions are used in a workflow policy, Generate Triggers displays the OR logic instead of the AND logic.

Table 69 describes example conditions to use to create a workflow policy based on the Account object.

Table 69. Description of Conditions Used in Example of GenTrig Behavior with Multiple Conditions

Property	Condition 1	Condition 2
Condition Field	Account Modification Num	Account Last Update By
Operation	>	<>
Value	0	0-1

Multiple triggers created for multiple conditions of one policy is expected behavior. This separates the functionality of GenTrig and WorkMon. GenTrig simply monitors database record changes and inserts records in Workflow Policy-specific tables. WorkMon evaluates violations, checks to see if the conditions associated with the rule of the violation are met, and executes the actions associated with a policy.

You cannot use AND between triggers generated for multiple conditions of a policy, because GenTrig can monitor only database changes, and database changes that violate different conditions are not always concurrent. So using an AND condition causes GenTrig to miss many violations.

For example, assume a policy has the following conditions:

- SR area is Network.
- Activity Priority is 1-ASAP.

Two triggers are generated. One trigger monitors an SR being created or updated, and checks if the area equals Network. The other trigger monitors an activity being created or updated, and checks whether the Priority equals 1-ASAP.

But if you use AND triggers and a user creates an SR without an activity, the trigger is not violated since the activity does not exist. If later, a user adds an activity to the SR, there still is no trigger violation because the SR record does not change. This violation is be missed due to use of AND logic. If, however, you use OR for the triggers and have WorkMon evaluate the condition, even though there are multiple violations in the S_ESCL_REQ table, WorkMon only processes one request because the other requests are not evaluated to TRUE.

About Database Triggers and Database Administration

It is important to keep your database administrators informed of active workflow database triggers, as a database Update or Insert event causes the database trigger to react, regardless of how the event is executed.

For example, if you have workflow triggers on inserts to the S_SRV_REQ table, and the database administrator does a table export and import of these records, the triggers treat every record in the database as if it is a newly inserted record, which can result in inappropriate actions being taken on old records that were simply imported again.

NOTE: In this release, the Generate Triggers task now requires the Privileged User Name and Password instead of Table Owner ID and Password.

Running Generate Triggers

Tips you can consider when running Generate Triggers, especially if you are deleting a policy, include:

- Deleting a policy then running Generate Triggers does not remove the database trigger. When you delete a policy, you must run Generate Triggers with the remove parameter set to TRUE. This removes every trigger. You must then rerun Generate Triggers to reset the triggers for existing policies.
- You must stop and restart the workflow monitor agents when running Generate Triggers.

- Generate Triggers must be rerun whenever you change policy conditions or policy groups. It is not necessary for you to rerun Generate Triggers when changing policy actions. For more information about changing the group for a workflow policy, see "Moving a Workflow Policy to a Different Group" on page 302.
- For SQL Server, have your default database set correctly. To determine your default database, launch the SQL Server Enterprise Manager then navigate to the SQL Server Machine name. Next, click Security then click LOGIN. The default database is listed to the right.
- You must start a new WorkMon task anytime you drop or regenerate triggers. This refreshes the WorkMon cache to account for the new changes in the policy being monitored.
- When running Generate Triggers, table names have an 18 character limit. With a table for which the name has 18 characters, Generate Triggers fails with the error *18 character limit, table_name trigger fail*. This is caused by a DB2 SQL limitation for trigger names being limited to 18 characters. The trigger name is derived from the table name plus a suffix such as U, I, D, U1,I1,D1, and so forth.

To Generate Triggers using the Siebel client

1. In the Siebel client, navigate to Administration-Server Management > Jobs.
2. In the Jobs list, click New.
3. From the Component/Job drop-down list, choose Generate Triggers.

   This creates a new line entry but does not start the task.
4. In the Job Parameters list, click New to modify parameter settings.

   For component-specific parameters for Generate Triggers, see Table 70 on page 280. For a description of generic and enterprise parameters, see *Siebel System Administration Guide*.
5. Enter your Privileged User name and password.
6. In the Job Detail form applet, from the applet-level menu, choose Start Job.
7. To view changes to the state, refresh the screen by clicking Run Query from the applet menu.

   Upon completion, the Status field contains Success or Error. You should view the log details.

Component-Specific Parameters for Generate Triggers

Table 70 describes component-specific parameters for Generate Triggers.

Table 70. Description of Component-Specific Parameters for Generate Triggers

Name	Value	Usage
Remove	TRUE or FALSE (default)	Set to TRUE to generate DROP TRIGGER statements to clean up the triggers. Remove does not generate CREATE TRIGGER statements.
Trigger File Name	Valid filename on the Siebel Server	Define the name and output location for the SQL script file. The default is TRIGGER.SQL. The file is created in the root directory of the Siebel Server during installation.
EXEC	TRUE or FALSE (default)	Specify a value to determine if the SQL script file runs automatically or manually. If TRUE, the SQL script file runs automatically. EXEC should be set to FALSE if you are running a Sybase server with any Siebel version or MS_SQL server with Siebel versions 4.x. This is performed to prevent connected users from getting an error message when Siebel generates database triggers. Make sure no one is logged in to the database before you generate triggers. **NOTE:** If you are creating a large number of triggers because there are too many workflow policies, it is recommended that the triggers be applied by the user and not the Generate Triggers server process. The Exec parameter must be set to FALSE in this case.
Mode	ALL or WORK or ASGN	Set to ALL to create both Workflow Policy triggers and Assignment Manager triggers. Set to WORK to create only Workflow Policy triggers. Set to ASGN to create only Assignment Manager triggers.
Privileged User Name/ Privileged User Password	Assigned Privileged User name and password	Specify to make sure users enter a Privileged User name and password. The Table Owner is considered a Privileged User, so you can enter the Table Owner name and password in the Privileged User name and password fields.

Running the SQL Script File

Once Generate Triggers has finished, run the SQL script file if the EXEC parameter is FALSE.

To run the SQL script file

1. Connect to the database server as the Siebel table owner using your RDBMS vendor's SQL tool. For example, ISQL for Microsoft or SQL*Plus for Oracle.
2. Run the SQL script file specified by the Trigger File Name parameter. The default filename is TRIGGER.SQL. The default location of this file is the root of the directory in which the Siebel Server was installed. For example:

   ```
   C:\siebsrvr\trigger.sql
   ```

3. Make sure no errors are reported.

For example, the policy administrator, Bill Stevens, has finished creating policies in the test Siebel client and needs the database triggers set in the Siebel database for the new policies. Using the Generate Triggers component, he sets the file output name.

This creates a file, TRIGGER.SQL, for the database administrator containing the triggers that must be modified or created in the test database for these policies.

The database administrator then runs the following command in SQL*Plus to create the triggers in the Oracle database:

```
SQL>@<path>\mytrig.sql
```

The successful creation of each database trigger in the Oracle database is indicated on the screen. For information on the syntax required for other databases, see your database documentation.

NOTE: On an MS SQL server database, execute the script trigger.sql as the database owner, dbo, login for the Siebel database.

About Database Triggers and Remote Users

When a remote user synchronizes, the changes get incorporated into the database. For example, account information in the S_ORG_EXT table is updated on synchronization. If you run a workflow which creates database triggers that compare changes in the database against specific conditions, then the triggers fire and rows are written to the S_ESCL_REQ table if the changes are of interest to the workflow conditions during synchronization.

About the S_ESCL_REQ Table

When a trigger fires against a Workflow policy condition, a record is inserted in S_ESCL_REQ, the *escalation request* table. This table contains the rows in the database that can trigger a workflow policy to take action. After the workflow monitor agent processes a request, it removes the row from this table.

Frequently, because of the way triggers are created, the logic defined on a workflow policy condition is not included on the trigger itself. Also, the conditions in the trigger file might not be indicative of the policies actually being violated. When running workflow monitor agent the records in the S_ESCL_REQ table causes workflow to evaluate the conditions for the related policy. So the triggers are only there to create indicators for the workflow engine to check the workflow policy conditions.

- For more information about how S_ESCL_REQ is used, see "About Workflow Monitor Agent" on page 290.

Remedies to Address Problems Involving the S_ESCL_REQ Table

Remedies you can apply to avoid or fix problems in the S_ESCL_REQ table include:

- Use your database tools to monitor the size and efficiency of the table. If the table becomes very large, this can indicate that too many policies are being monitored and a new workflow policies process needs to be created to share the load. If rows are being monitored and not being removed after the time interval is met, this can indicate that a policy was deactivated without removing the database triggers. The triggers are continuing to send data that is not being acted on by a workflow policies instance.
- If creating a new business component object definition, do not set the Table property to S_ESCL_REQ. A business component cannot be based on the S_ESCL_REQ table.
- Expire rather than delete a workflow policy. For more information, see "Expiring a Workflow Policy" on page 303.
- Remove rows that belong to obsolete workflow policies. For more information, see "Deleting an Obsolete Workflow Policy" on page 306.

Administering Email Manager and Page Manager

This topic describes how to administer the email manager and page manager components. It includes the following topics:

Setting Up the Siebel Server for Email Manager

Some workflow policy actions allow you to send email messages to specific individuals. Considerations to weigh include:

- To send email using workflow policies and an SMTP/POP3-compliant mail system, both must be working properly on your network.
- The Email Manager component of the Siebel server must be running. This is part of the Communications Management component group, which must be configured.
- Email Manager uses profiles set up in Communications Manager. The Communications Outbound Manager does verification of the profile. You then must make sure this profile is configured correctly to log into the SMTP/POP3-compliant mail system.
- You must also set the Mail Profile parameter to the name of the communication profile you need to use for sending the email.

Date Format in an Email Message

A workflow policy program is executed by the Siebel server, which connects to Oracle through ODBC. As part of this process, the required data is retrieved from the database through this connection. The format of a date in an email message is the format returned from the ODBC driver, which might be different to that used by Oracle.

Setting Up the Communications Profile to Send Email Through Workflow

Sending email through Workflow involves creating a communications profile.

NOTE: To create a new communications profile, CompGrp *CommMgmt* must be configured. Make sure it is configured before beginning the following procedure. For more information, see *Siebel Communications Server Administration Guide*.

To create a Communications Profile

1. In the Siebel client, navigate to Administration-Communications > Communications Drivers and Profiles.
2. In the Communications Drivers list applet, query the Name field for the *Internet SMTP/POP3 Server* communications driver.
3. Click the Profiles view tab.
4. Create a new profile called [Profile Name].
5. In the Profile Parameters Overrides applet, create records for parameters using values described in the following table:

Parameter	Usage
From Address	Define the sender's email address for outbound communications.
POP3 Account Name	Define the account name for the POP3 mailbox from which to retrieve inbound communications.
POP3 Account Password	Define the password for the POP3 mailbox account.
POP3 Server	Define the host name or IP address of the machine on which the Internet POP3 server is running, as appropriate for your network configuration.
SMTP Server	Define the host name or IP address of the machine on which the Internet SMTP server is running, as appropriate for your network configuration.

For details on these parameters, see *Siebel Communications Server Administration Guide*.

After creating your communications profile, you must create the component definition of Email Manager. The Email Manager component executes email actions once the conditions of a Workflow policy are met.

Starting Email Manager

You can start Email Manager from the command line, or from the Email Manager Component view.

To start email manager from the command line

To start the Email Manager task with the [Profile Name] profile, in the server manager command line, use the following command to start a task for Email Manager:

```
start task for comp MailMgr with MailProfile=<Profile Name>
```

To start email manager from the server administration view

1. Navigate to Administration-Server Configuration > Servers > Components.
2. In the Components list, find the Email Manager component.
3. In the Component Parameters list, set Mail Profile = Test.
4. To automatically start up a task for Email Manager whenever the component is restarted or the Siebel Server services are restarted, set the component parameter Default Tasks = 1.

How Outbound Communications Manager Sends Email

The Outbound Communications Manager sends email messages according to the following process:

1. When the workflow policy is violated, workflow monitor agent inserts a record into the S_APSRVR_REQ table for workflow actions that invokes the Send Email workflow policy programs.
2. Email Manager picks up records from the S_APSRVR_REQ table, setting their status from QUEUED to ACTIVE then to SUCCEEDED during the course of the execution.
3. Outbound Communications Manager is invoked to log onto the [Profile Name] profile.
4. Outbound Communications Manager sends emails to recipients using the Send Message method of the Outbound Communications Manager business service.

About the Mail Profile Parameter

The Mail Profile parameter specifies the mail profile to use and is defined in Control Panel. The parameter establishes the connection between the application and the email system. If you do not specify a profile here, the default profile is used. The names must match exactly.

Setting Up the Siebel Server for Page Manager

Some workflow policy actions allow you to send page messages to specific individuals. The Page Manager component of the Siebel server must be running for you to send a page. Some actions can page specific individuals with alphanumeric or numeric pagers. Prerequisites to send a page using a workflow policy include:

- The server running the Page Manager component has access to a local or network modem.
- The Page Manager component of the Siebel server is running. Several parameters, similar to dial-up networking set up in Windows, must be set prior to running the Page Manager component.
- Enter the appropriate telephone numbers for paging in the Employee view. These are the numbers used by Workflow.
- Change the regional configuration to avoid inputting the country code prior to the telephone number. This can cause errors.
- Change the PAGE_TYPE list of values parameters to make the page manager accept an alphanumeric send. This means the language and the value displayed.

NOTE: Alphanumeric paging is more reliable than numeric paging because the pager messages are transmitted by the computer at the pager companies. This is not true for numeric paging, where pager messages are sent by emulating key presses on a phone. Failures in sending numeric pager messages are very difficult to detect.

The Page Manager component uses the industry standard protocol Telocator Alphanumeric Protocol (TAP) for alphanumeric paging. Check with your pager company for the phone number to send your alphanumeric paging.

Several parameters affect how the Page Manager component interacts with the modem. You can change these parameters in the Server Administration screen. The available parameters are listed. The modem parameters are the defaults for Hayes-compatible modems. Make sure the settings are compatible with your modem.

To run the Page Manager component

1. In the Siebel client, navigate to Administration-Server Management > Jobs.
2. In the Jobs list, click New.
3. From the Component/Job drop-down list, choose the Page Manager component.
4. In the Job Parameters list, click New.
5. Click Parameters in the Server Tasks applet and enter your parameters.

   For a list of parameters, see Table 71 on page 286.

Parameters for the Page Manager Component

The most important parameters are Modem Port, Dial Prefix, Long Distance Prefix, and Local Area Code. Change the values for these parameters to match your system. Table 71 describes default values used if you do not specify a parameter.

Table 71. Description of Parameters of the Page Manager Component

Parameter	Usage	Description
Modem Port	Define the component object model (COM) port where the modem is attached. Default is COM1..	Valid values are COM1, COM2, and so forth.
Dial Prefix	Define a number or sequence of numbers that needs to be dialed to get an outside line. Default is 9.	If no dial-out prefix is used, insert a comma (","). Note that if dialing 9 for an outside line is not required and you are using srvrmgr.exe from the command line, specifying a comma for this parameter returns an error. But, if you specify a hyphen (or other characters that cannot be dialed), an error is not returned. Example command: `SRVRMGR.EXE /g NT01022 /e SBLPRD_ENT502 /s SBLPRD_APP502 /u ***** /p ***** /c "START TASK FOR COMPONENT PageMgr WITH DialPrefix = '-'"`
Long Distance Prefix	Define the long-distance prefix. Default is 1.	This value is added in front of long-distance phone numbers. Set this parameter to equal an empty string if your location does not require a long-distance prefix to be dialed.
Local Area Code	Define the area code of your location. Default is [empty].	If the beginning digits of a phone number are equal to this code, they are removed before the phone number is dialed, and the long-distance prefix is not added.
Delay1	Define the number of seconds to wait between dialing a phone number and simulating key presses for the first set of numbers. Default is 12.	This applies only to numeric paging. It is ignored for alphanumeric paging.

Table 71. Description of Parameters of the Page Manager Component

Parameter	Usage	Description
Delay2	Define the number of seconds to wait between simulating key presses for the first and second set of numbers. Default is 4.	This applies only to numeric paging. It is ignored for alphanumeric paging. This is also ignored if the numeric pager does not have a personal identification number (PIN).
Modem Reset String	Define the modem command used to reset the modem. Default is ATZ.	To determine the correct command, see your modem documentation.
Modem Init String	Define the modem command used to initialize the modem. Default is AT&FQ0V1.	To determine the correct command, see your modem documentation. For example, some modems require a numeric value after &F.
Modem Dial String	Define the modem command used to dial the modem. Default is ATDT.	This parameter is typically rarely changed.
Modem Hangup String	Define the modem command used to hang up the modem. Default is ATH.	This parameter is typically rarely changed.
Modem Restore String	Define the modem command used to restore the power-up settings of the modem. Default is AT&F.	This parameter is typically rarely changed.

Troubleshooting Email Manager and Page Manager

Email Manager stops processing when it is unable to log on to the mail server (SMTP/POP3-compliant server and logs an error message in the trace files.

Page Manager stops processing if the modem is not available. The requests continue to accumulate in the Requests table. After you fix the processing problems, you must restart the servers. The servers continue processing from the point where they were interrupted.

If Email Manager is able to log on but has a problem sending a particular email, it logs an error message and continues on to the next request. If Page Manager is able to interface with the modem but has a problem with a given page send, it logs an error and moves on to the next request.

When workflow policies executes email and paging actions, it inserts email requests and paging requests into the database. These requests are inserted as records in the S_APSRVR_REQ table, which are then processed by Email Manager and Page Manager.

New requests have a status of QUEUED. After a request is picked up by Email Manager or Page Manager, but before it is processed, it has a status of ACTIVE. After a request is processed, the request's status becomes SUCCEEDED if the processing is successful, or FAILED if an error occurs.

To generate sending emails, Siebel Server uses the UNIX *Mail* command.

Use the following procedure to make sure your server platform can run the command to a valid recipient and to make sure email was successfully sent.

To make sure the server platform can run the command to a valid recipient

1. From the UNIX command prompt, type:

   `>mail recipient email address`

   Where `recipient email address` is a valid address.

2. Type a message ending with a period on the last line to indicate the end of the message.
3. Press enter.
4. If email written to S_APSRVR_REQ is not sent, even though the Email Manager trace file displays status SUCCEEDED, make sure the following Outlook settings on the server are set:

   - Send messages immediately
   - Check for new messages every *[x]* minutes

   Both of these options must be configured in Outlook for email messages to be sent successfully.

Executing a Workflow Policy with Workflow Action Agent

The Workflow Action Agent process submits a request to Email Manager and Page Manager when actions are to be taken.

NOTE: You must create a component definition for each Workflow Action Agent task.

Tasks performed by Workflow Action Agent include:

- Processes requests logged in S_ESCL_ACTN_REQ, the escalation action request table, for a single group.
- Invokes actions linked with the Workflow policy being processed.
- Logs email and page actions in the S_APPSRVR_REQ table for execution by Email Manager and Page Manager.
- Purges requests from S_ESCL_ACTN_REQ after processing.

If the Use Action Agent parameter is set to TRUE in the Monitor Agent process, you must perform the tasks described in "Starting the Workflow Action Agent Process" on page 289.

Starting the Workflow Action Agent Process

Start the Workflow Action Agent in the same way that you start the Workflow Monitor Agent. For more information, see "Starting Workflow Monitor Agent" on page 292.

Considerations to weigh include:

- For each workflow monitor agent, there is a workflow action agent. For a given Workflow Monitor Agent, start one, and only one, Workflow Action Agent.
- When using email consolidation, and when the parameter Use Action Agent is set to TRUE on the Workflow Monitor Agent, you must start Workflow Action Agent separately.

For more information on arguments for starting a Workflow Agent process, see *Siebel System Administration Guide*.

Shutting Down the Workflow Action Agent Process

You shut down the Workflow Action Agent in the same way that you shut down the Workflow Monitor Agent. For more information, see "Starting Workflow Monitor Agent" on page 292.

When restarting a workflow policy process, a Workflow Action Agent process immediately begins tracking relevant activities that have occurred since it was shut down.

Using Workflow Action Agent for Batch Policies

You can run Workflow Action Agent separately for batch policies.

To run workflow Action Agent for batch policies

1. Start Workflow Monitor Agent with the following parameters:
   - Group Name.
   - Processes the batch policies is TRUE.
   - Use Action Agent is TRUE.
2. Start Workflow Action Agent with the Group Name parameter.

The Workflow Monitor Agent continues processing and puts the qualified rows in the S_ESCL_ACTN_REQ table. The Workflow Action Agent executes actions for rows in S_ESCL_ACTN_REQ and deletes the rows from S_ESCL_ACTN_REQ afterward.

The Workflow Action Agent might be of use if the action being executed is long-running or intensive. However, in general Workflow Action Agent does not help with batch policies.

Executing a Workflow Policy with Workflow Monitor Agent

This topic includes the following topics:

About Workflow Monitor Agent

Workflow Monitor Agent checks when the conditions of a workflow policy is met, and executes actions once conditions are met. To execute your workflow policies, you must start Workflow Monitor Agent. You start and stop the Workflow Monitor Agent task in the Administration-Server views.

Table 72 describes some of the database tables involved with workflow policies.

Table 72. Description of Database Tables Involved with Workflow Policies

Table Name	Description
S_ESCL_REQ	**Escalation request.** This table holds the potential matching requests caused by applications. For more information, see "About the S_ESCL_REQ Table" on page 281.
S_ESCL_STATE	**Escalation state**. This table holds the time-based policy matches.
S_ESCL_ACTN_REQ	**Escalation action request**. (Optional) This table holds the requests to execute actions. This is only used if Use Action Agent is set to TRUE.
S_ESCL_LOG	**Escalation log**. This table holds a history of base table rows that have matched policies.

What Workflow Monitor Agent Does

Workflow Monitor Agent executes several server processes that monitor the Siebel database. Tasks performed by Workflow Monitor Agent include:

- Checks the S_ESCL_REQ table to see when the conditions of a policy are met.
- Monitors policies within a single group.

  You can only run one workflow monitor agent process against a specific group at one time. You can run multiple workflow monitor agent processes at the same time, but they must be against different groups. If you run two workflow monitor agent processes against the same group, deadlocks occur.
- Generates requests for workflow action agent in the S_ESCL_ACTN_REQ table. This occurs if you set Action Agent to True.

- Purges requests from the S_ESCL_REQ table after processing. When a database trigger is activated because a workflow policy condition is met, a record is inserted into the Escalation Request table, S_ESCL_REQ. Workflow Monitor Agent, Workmon, evaluates the request against the rules set up by the policies in the workflow policy group.

If you set Action Agent to True, and if Workmon determines that the request in the S_ESCL_REQ table has no duration defined in the policy, Workmon takes direct action and logs an entry into the S_ESCL_LOG table or sends it to the S_ESCL_ACTN_REQ table.

If Workmon determines that the request has a time element that must be met, the request is sent to the S_ESCL_STATE table along with the expiration time. The request stays in the S_ESCL_STATE table until the expiration time is met, or the request is removed because the conditions of the policy are no longer met. Workmon evaluates each of the requests that remains in the S_ESCL_STATE table for a time duration match or to determine if the condition still matches in the S_ESCL_STATE table. If you set Action Agent to True, then as each match occurs WorkMon takes direct action and logs an entry into the S_ESCL_LOG table or sends it to the S_ESCL_ACTN_REQ table.

NOTE: If a workflow policy has a specified duration, the duration time is calculated from the time WorkMon detects that the row is in violation of the policy, not from the time the row was inserted into the S_ESCL_REQ table. For example, if you create a policy and set the duration as one week, but then WorkMon is not started until several days after Generate Triggers is run, the policy action fires one week from when WorkMon is started, not one week from when the policy is created or Generate Triggers is run.

When the request for an action is made to the S_ESCL_ACTN_REQ table, Workflow Action Agent executes the action and logs an entry into the S_ESCL_LOG table.

About S_ESCL_REQ.REQ_ID

S_ESCL_REQ.REQ_ID can hold up to 9 digits, so the maximum value allowed is 999,999,999. Because the S_ESCL_REQ_S sequence does not have an upper boundary, the sequence for S_ESCL_REQ.REQ_ID can potentially generate a number greater than the maximum of 9 digits allowed, causing an overflow. To avoid this, it is recommended you perform one of the following actions:

- Use a database tool to increase the column size.
- Use a database tool to reset the S_ESCL_REQ_S sequence.

As a separate issue, you might encounter a message similar to *Status: Deleting requests ([a numeric identifier])*.

This message does not indicate the number of rows WorkMon is deleting. The numeric identifier indicates the REQ_ID from S_ESCL_REQ, which is generated by the database to uniquely number the rows in the database. For example, the ways in which REQ_ID is generated include:

- From the Sequence object in an Oracle database.
- From the identity column in an MS SQL Server database.
- From the UDF (User Defined Function) in a DB2 database.

There is no correlation between REQ_ID and the number of rows in S_ESCL_REQ.

Checking Modifications Into the Local Database

The Workflow Monitor Agent and Workflow Action Agent read workflow configurations from the server database. If you use Siebel Tools to modify a workflow policy object, column, or program in the local database, then you must check these objects into the server database. Otherwise, workflow monitor agent and workflow action agent have no knowledge about these modifications.

Also, you should restart workflow monitor agent and workflow action agent after modifying a workflow policy object, column, or program through Siebel Tools, or after modifying a workflow action through the workflow administration views in the Siebel client.

If you do not restart workflow monitor agent and workflow action agent, the modifications do not take effect until policies are reloaded, as specified in the Reload Policy parameter. Since the Reload Policy parameter is set to 600 seconds by default, in most cases 10 minutes elapse before the modifications take effect if you do not perform a restart.

It is not necessary to restart the Workflow Process Manager when modifying a workflow process. For more information, see "Modifying a Workflow Process" on page 58.

Using Workflow Monitor Agent

Before you start workflow monitor agent, you must create a separate server component definition for each workflow monitor agent task. You start workflow monitor agent from the server manager command-line interface.

Replication and Workflow Monitor Agent

Within the entire enterprise architecture of a Siebel deployment, there can be only one workflow monitor agent monitoring a particular workflow group.

For example, a regional node can be running a workflow monitor agent that monitors a group called Group 1. Meanwhile, in the headquarters, there is another WorkMon running, which monitors a group called Group 2. In this way, the organization is able to run workflow policies where needed, while working with the restriction of one WorkMon for one group. You cannot run more than one instance of workflow monitor agent and workflow action agent for a particular workflow group. However, you are allowed to have multiple workflow monitor agent and workflow action agent processes for different groups running at the same time.

Starting Workflow Monitor Agent

This topic describes procedures you perform when using workflow monitor agent. To start this predefined component, you must first create a new workflow monitor agent component definition.

In releases prior to Siebel version 7.0, you can start workflow monitor agent from the Server Tasks view in the Server Manager user interface. This is no longer the case. Starting with Siebel version 7.0, you must start workflow monitor agent from the Server Manager command-line interface.

To create a new workflow monitor agent Component Definition

1 Navigate to Administration-Server Configuration > Enterprises > Component Definitions.

2 In the Component Definitions list applet, create a new record using values described in the following table:

Field	Description
Name	(Name of the component.)
Component Type	[Workflow Monitor Agent]
Component Group	(Choose an existing component group)
Description	(Description of the component)
Alias	(Alias for the component. The alias cannot contain blank spaces.)

3 Save the record.

4 In the Component Parameters list applet, choose the Group Name parameter record then enter the name of the Workflow Policy Group for which this component processes requests.

5 (Optional) Make additional changes to the component parameters.

For more information, see "Workflow Monitor Agent Command-Line Interface Parameters" on page 295.

6 In the Component Parameters list applet, choose the Default Tasks parameter record then set the Value to 1.

This component automatically starts up and shuts down with the Siebel Server.

Do not set Default Tasks to a value greater than 1. Setting Default Tasks to a value greater than 1 can cause undesirable behavior, such as multiple tasks starting for the same workflow group.

7 In the Component Definitions list applet, choose Enable Component Definition from the applet menu.

The definition state changes from Creating to Active.

8 Stop then restart the Siebel Server so that the component definition takes effect.

NOTE: Whenever you make a change to a component parameter, the component must be stopped and restarted for the changes to take effect. Also, you must define a Workflow Monitor Agent component definition for each Workflow Policy Group.

To start workflow monitor agent using the server manager command-line interface

1 Start the server manager by entering the following into the Server Manager CLI:

```
srvrmgr /g <gateway server address> /s <Siebel Server name> /e <enterprise server
name> /u <server administrator username> /p <server administrator password>
```

2 Start a new Workflow Monitor Agent task in background mode by entering:

```
start task for component WorkMon with SleepTime=<time>,GroupName=<group name>
```

3 (Optional) Start a new Workflow Monitor Agent task to run in batch to process a Batch policy by entering:

```
start task for component WorkMon with GroupName=<group name>, BatchMode=Y
```

You must create a separate server component definition for each Workflow Monitor Agent task.

CAUTION: If a Workflow Process Manager task fails, and more tasks are started that also fail, then Workflow Monitor Agent exits with error after only one violation. This can result in Workflow Monitor Agent starting more, unneeded tasks when the WfProcMgr task fails. To avoid this condition stop, then restart wfprocmgr.

For more information, see "Workflow Monitor Agent Command-Line Interface Parameters" on page 295. For more information on using the Siebel Server Manager Command-line Interface, see *Siebel System Administration Guide*.

To stop or restart a workflow monitor agent Component

1 In the Siebel client, navigate to Administration-Server Management > Components.

2 Choose the component, then click Shutdown or Startup.

NOTE: When you make changes to the parameters of the component, the component must be stopped and restarted for the changes to take effect. Also, you must define a Workflow Monitor Agent component definition for each Workflow Policy Group.

Workflow Monitor Agent Command-Line Interface Parameters

Table 73 describes parameters that can be set by using the Workflow Monitor Agent command-line interface.

Table 73. Description of Parameters for WorkMon Command-Line Interface

Parameter Name	Display Name	Usage	Default Value
ActionAgent	Use Action Agent	Specify whether the Action Agent is automatically run with Monitor Agent. If set to FALSE (the default setting), the Workflow Action Agent server component starts within Workflow Monitor Agent, and actions are then executed by Workflow Monitor Agent. Several conditions apply when configuring Use Action Agent. For more information, see "Setting the Use Action Agent Parameter" on page 300	FALSE
ActionInterval	Action Interval	Define the action execution interval in seconds. This argument determines when actions for a given policy are executed again on a given base table row. The purpose of this argument limits the number of times actions are executed if a row keeps going in and out of a matching condition. In other words, if the same record repeatedly violates the same policy before the action interval expires, the record is removed from the S_ESCL_REQ table and the action is not performed again. Note: The default is 3600 seconds. If this parameter is used, the value *must* be greater than 0 (zero) or unexpected behavior can result.	3600
BatchMode	Processes the batch policies	Specify whether Monitor Agent is running in batch mode. When the value is set to TRUE, only the policies that have the Batch flag set to TRUE are evaluated. When FALSE, only the policies that have the Batch flag set to FALSE are evaluated. Note that when starting with BatchMode set to TRUE, Workflow Monitor Agent is run once. That is, it processes records in the table then exits.	FALSE

Table 73. Description of Parameters for WorkMon Command-Line Interface

Parameter Name	Display Name	Usage	Default Value
CheckLogCacheSz	Cache size of Policy violations	Define the number of policy violations to store in cache. This parameter determines how many violated policies can be stored in the cache. Based on the presence of the ROW_ID and RULE_ID in the map/cache, the cache determines whether or not an action is already initiated. The Workflow Monitor checks S_ESCL_LOG for policy violations for the same BT_ROW_ID and RULE_ID.	100
DeleteSize	Request delete size	Define the number of records to commit at a time. The minimum is 1. If Workflow Monitor encounters deadlocks, you can reduce the default to 125 with minimal performance degradation. Note: to avoid call stack errors, do not set the Request Delete Size value to zero.	500
GenReqRetry	Number of seconds to retry	Define the number of seconds to retry sending a Generic Request message.	120
GroupName	Group Name	Required. Define the workflow policy group that Monitor Agent works on.	(This cell is intentionally empty.)

Table 73. Description of Parameters for WorkMon Command-Line Interface

Parameter Name	Display Name	Usage	Default Value
IgnoreError	Ignore errors	Specify whether to ignore errors while processing requests. By default, the Workflow Monitor and Action agents do not ignore errors that occur while processing the requests. If Ignore Errors is set to TRUE and an error is encountered, the agent processes log the error condition, delete the request, then continue working. By setting this argument to FALSE, the agent processes exit on an error and send an email message to the mail ID specified by the Mailing Address argument. When running Workflow with Ignore Errors=TRUE, note that valid errors are ignored. Whereas, if Ignore Errors is set to FALSE, the agent stops and exits with the error. **NOTE:** It is recommended that you set Ignore Errors to FALSE so that valid errors are not ignored.	FALSE
KeepLogDays	Number of days to keep violation information	Define the number of days worth of violation information to be retained. Sets the number of days log information is stored. Log information older than the number of days set is automatically removed. This value can be set to 0 to prevent the purging of this log information. For more information, see "KeepLogDays Parameter and Workflow Monitor Agent" on page 300.	30
LastUsrCacheSz	Cache size of last user information	Define the number of last user information items to cache. When executing actions, the information about the last user to modify the base table row is available as a token substitution in the program arguments. By caching this information in the server, the throughput performance of executing actions can potentially increase.	100
MailServer	Mail Server	Define the name of email server to send notification of abnormal termination.	(This cell is intentionally empty.)

Table 73. Description of Parameters for WorkMon Command-Line Interface

Parameter Name	Display Name	Usage	Default Value
MailTo	Mailing Address	Define the mail address to review notification of abnormal termination. Mail to [mail ID] if a Workflow Agent process exits with an error condition. An error can be caused by the failure of an action to execute, invalid object definitions, and so forth.	(This cell is intentionally empty.)
NumRetries	Number of Retries	Define the number of retries for recovery. This parameter works with the Retry Interval and Retry Up Time parameters to reconnect MTS or Siebel Server components to the database if database connectivity is lost.	10000
ReloadPolicy	Reload Policy	Define the policy reload interval in seconds. This argument defines the frequency that policies are reloaded into the engine. This allows changes to be made on the screens and with the Generate Triggers component. The engine acts on the changes within some time frame. The default is 600 seconds.	600

Table 73. Description of Parameters for WorkMon Command-Line Interface

Parameter Name	Display Name	Usage	Default Value
Requests	Requests Per Iteration	Define the maximum number of requests read for each iteration. This controls the maximum number of requests WorkMon reads from the requests queue within one iteration. Between iterations WorkMon deletes processed requests from the requests queue and commits, optionally reloads policies from the database, checks for shutdown request, and optionally sleeps. In other words, you can think of the Requests Per Iteration parameter as a way to control the maximum amount of work WorkMon performs before taking these steps between iterations.	5000
Sleep Time	Sleep Time	Define the time in seconds that the Workflow Agent process sleeps after it has polled for events and fulfilled obligations to notify. Once the Workflow Agent process has finished its obligations, the Workflow Agent process stops polling for the time period set by the sleep interval. This parameter affects both the performance of the Workflow Agent process and the responsiveness of the application server. For more information, see "Sleep Time Parameter and Workflow Monitor Agent" on page 300.	60

Setting the Use Action Agent Parameter

Table 74 describes conditions in which to set the Use Action Agent parameter.

Table 74. Description of Setting Use Action Agent

When to Set Use Action Agent to False	When to Set Use Action Agent to True
In most cases, it is recommended you start Workflow Monitor Agent with Use Action Agent set to False, the default setting. This way, Workflow Monitor Agent monitors conditions as well as executes actions. Some possible actions that are executed include workflow processes, workflow programs, and email programs. If you start Workflow Action Agent to execute actions when Use Action Agent is True, and the action is executing a workflow process, you might encounter access violation errors and Workflow Action Agent can fail.	It is recommended you start Workflow Monitor Agent with Use Action Agent set to True when executing an action that involves email consolidation. If you are using email consolidation and Use Action Agent is set to True, then you must start Workflow Action Agent separately.

Sleep Time Parameter and Workflow Monitor Agent

You can increase the Sleep Time to adjust the processing frequency, because the WorkMon task waits for more requests when it goes to sleep. By default, the task sleeps every one minute (the default setting is 60 seconds). The WorkMon task wakes up, processes requests (if there are any), then goes back to sleep. WorkMon does this repeatedly.

KeepLogDays Parameter and Workflow Monitor Agent

If the Number of Days to Keep Violation Information parameter is set to 0, then WorkMon does not purge from S_ESCL_LOG. If you set this parameter to 0, then monitor the size of S_ESCL_LOG, because the size of this table affects performance. Use this parameter judiciously.

Because infrequent cleaning of S_ESCL_LOG can lead to slow performance, decide how often you need to start WorkMon using KeepLogDays, then restart WorkMon with a setting of 0 for this parameter.

Batch Processing Workflow Policies

This topic describes how to configure workflow policies to execute in batch mode.

Creating Batch Policies

You can create workflow policies as batch policies by checking the Batch check box in the Policies applet. When you start workflow monitor in batch mode, it checks for policies with the Batch check box marked. Each policy causes an SQL statement to be issued to identify the specific records that meet the policy conditions. The records identified are then processed in turn and the appropriate actions are carried out.

You can use the batch feature to consolidate email messages for a designated recipient by checking the Batch field in the Actions applet in the Workflow Policies view.

If you consolidate email messages, the recipient receives one email with the information of multiple actions rather than multiple emails. For example, you can create a workflow policy that sends an email to the director of sales each time a quote is submitted with a discount over 30%. If 20 sales representatives submit quotes with the 30% discount and the Batch field is checked, the director of sales receives one email listing the 20 quotes. If the Batch field is not checked, the director of sales receives 20 email messages.

NOTE: When creating a batch type workflow policy, the comparison operators IS ADDED, IS UPDATED, or IS DELETED *must* be used in conjunction with regular conditions. These comparison operators are considered special conditions intended for Dynamic mode when triggering rows to look up regular conditions.

Running a Workflow Policy in Batch Mode

The ability to run a Workflow Policy in batch mode allows you to evaluate more data in the database, not just the records that triggered the Workflow Policy. This feature is useful when defining a new policy that needs to be applied to historical data and for enforcing policies that have date conditions.

To run a Workflow Policy in batch mode

1. In the Workflow Policies view, set the Batch field on the workflow policy to True by checking the Batch check box in the Actions applet.
2. Associate the workflow policy with a workflow group that is used only for batch processing, using the control on the Group field.
3. When starting the workflow monitor and workflow action server components, specify the appropriate group name and set the parameter *Process the batch policies* to TRUE.

Using Batch Mode to Make Sure a Workflow Policy is Triggered Correctly

Consider the following example that demonstrates why previously existing records that meet the workflow policy conditions do not trigger the workflow policy:

Table 75. Example of Records That Meet a Policy But Do not Trigger the Policy

Field	Value
Workflow Policy Name	Account Active
Workflow Object	Account
Group	Test

A workflow policy based on a business object is created to monitor a field when:

- Workflow Condition, Account Status is Active.
- Workflow Action, Action: Send Email to Employee.
- Remove and generate triggers, then start a workflow monitor task for the workflow group *Test*.

Considerations you should weigh include:

- New Account records that are added which meet these conditions trigger the workflow policy.
- Existing Account records that are updated to meet the conditions also trigger the workflow policy.

NOTE: Account records that previously existed in the database before the workflow policy was created but also meet the workflow policy conditions *do not* trigger the workflow policy.

If triggers have been generated for this workflow policy, workflow monitor agent evaluates records only if they are updated, or if a new record is created. Therefore, old records are not affected.

To evaluate account records including those created prior to the creation of the workflow policy rule, make sure the workflow policy has the Batch flag checked and the workflow monitor agent is run with Batch Mode set to TRUE.

Moving a Workflow Policy to a Different Group

In response to changes in the business environment or you configuration, you might find it necessary to change the workflow group for a given workflow policy.

Before you can move a workflow policy from one group to another group, requests associated with that workflow policy must finished. If a workflow policy's group is changed while associated requests are pending, Workflow Monitor Agent will fail with the error Rule Not Found. If this occurs, restore the workflow policy to the policy's original group, wait for the requests to finish, then proceed with the change.

When moving an existing workflow policy from one workflow group to another workflow group, the triggers must be updated. The Triggers will contain the RULE_ID and GROUP_ID for the Workflow Policy. So, if the triggers have not been re-generated, rows will exist in S_ESCL_REQ the reference the original RULE_ID and GROUP_ID.

To move a Workflow Policy to a different Group

1. Run a generate triggers task to drop database triggers, thus making sure no more rows are inserted for workflow policies in the group you need to change. Use the following parameters:

   ```
   EXEC=TRUE

   Remove=TRUE
   ```

   TIP: Since triggers are being disabled you need to avoid having additional triggers being fired by workflow policies. Therefore, consider making these changes when either no users or only a minimal number of users are accessing the system.

   For more information, see "Running Generate Triggers" on page 278.

2. If a Workflow Monitor Agent task is already running, allow the task to finish.

   This makes sure the task processes the remaining rows in the S_ESCL_REQ table for the workflow policy group.

3. Query the S_ESCL_REQ table for the GROUP_ID to make sure no more rows are present for GROUP_ID.

4. If no monitor agent task is running for the group, start a task for the group with the proper group name.

5. Once the remaining rows in the S_ESCL_REQ table for the group have finished processing, stop the workflow monitor agent task for the group.

6. Reassign the workflow policy to the group by changing the workflow policy's group name field to the new group name.

7. Re-generate database triggers, using the following parameters:

   ```
   EXEC=TRUE

   Remove=FALSE
   ```

8. Re-start the Workflow Monitor Agent for the old group.

9. Start a new Workflow Monitor Agent for the new group.

Expiring a Workflow Policy

When expiring a workflow policy you should remove records associated with the workflow policy in the S_ESCL_REQ table. This is preferable to deleting a workflow policy.

If you encounter a *failed to load parent rows* error message, it can be due to the presence of rows in the S_ESCL_REQ, S_ESCL_STATE and S_ESCL_ACTN_REQ tables that reference expired policies. The procedure in this topic allows you to determine, by using SQL queries, if there are rows in these tables that reference policies that no longer exist in S_ESCL_RULE.

To avoid workflow manager exiting with error, make sure there are no outstanding records in the S_ESCL_REQ, S_ESCL_ACTN_REQ or S_ESCL_STATE tables before expiring the policies.

CAUTION: Setting Ignore Errors to True is primarily used to clean up the Workflow Tables. It is strongly discouraged to permanently set Ignore Errors to True.

To remove rows belonging to expired or deleted policies

1. Look for the following error message in WorkMon_xxx.log trace file:
   - [DBG33] 2000-01-24 08:49:30 Message: Rule not found
   - [DBG33] 2000-01-24 08:49:30 Message: Failed to load parent rows

   This error message indicates an error when Workflow Monitor agent attempts to process records from the S_ESCL_REQ, S_ESCL_STATE or S_ESCL_ACTN_REQ tables, and the policy that it is trying to review is expired or deleted. For more information about these tables, see "About Workflow Monitor Agent" on page 290.
2. Run the following query to determine if there are records that reference expired policies:

   ```
   select row_id, name, expire_dt

   from s_escl_rule

   where expire_dt is not null and expire_dt <= getdate()
   ```

   Note that sysdate is an Oracle date time function and the corresponding function for SQL Server is getdate().
3. Note the ROW_ID's for the expired policies exposed in Step 2.
4. Run the following query to determine the number of records in the S_ESCL_REQ table that reference expired policies:

   ```
   select * from s_escl_req where rule_id in (select row_id from s_escl_rule where
   expire_dt is not null and expire_dt <= getdate())
   ```
5. Repeat Step 4 for the S_ESCL_STATE and S_ESCL_ACTN_REQ tables.
6. If there are records in the S_ESCL_REQ, S_ESCL_STATE, S_ESCL_ACTN_REQ tables that reference expired policies, in the Siebel client, start the Workflow Monitor Agent with Ignore Errors set to TRUE.

   Workflow Monitor Agent should be able to bypass the error and clean the S_ESCL_REQ table.

   Note the caution information provided at the beginning of this topic.
7. Stop the task.
8. Restart the task with Ignore Errors set to FALSE.

   This makes sure other types of errors are not missed.

If the error continues after completing this procedure, continue with delete rows by RULE_ID, expire old policies, and drop triggers.

To delete rows by RULE_ID, expire old policies, and drop triggers

1. Make a backup copy of the database or the effected tables.
2. Identify the RULE_IDs that belong to policies that no longer must be enforced by Workflow.
3. Use SQL to manually delete the rows that reference expired or deleted policies.

   Delete these policies by RULE_ID. For example:

   ```
   delete from [table name] where RULE_ID = 'rule_id'
   ```

   Note that the Siebel application does not support direct delete or update statements on the Siebel database. Also, note that S_ESCL_REQ, S_ESCL_ACTN_REQ, S_ESCL_STATE are the only tables that do not have dependencies. Make sure you perform this procedure in accordance with Siebel Support guidelines.

   Performing the backup in Step 1 makes sure you have a backup you can use to restore the tables to their state before rows were deleted, if necessary.
4. For policies that should not be active, expire them by putting an older date time in the Expiration Date/Time field for the policies.
5. To drop triggers that reference expired policies, run the generate trigger server task with the parameters listed, below.

   Drop Triggers:

   ```
   EXEC = True

   Remove = True
   ```

   Generate Triggers:

   ```
   EXEC = True

   Remove = False
   ```

   If some policies have expired and not dropped and regenerated triggers, the triggers for those policies can still be causing records to be inserted into the Workflow tables.

   Dropping these triggers should prevent rows related to those expired policies from being created in the S_ESCL_REQ table.
6. If necessary, regenerate triggers.

   For more information, see “Running Generate Triggers” on page 278.

7 If the errors persist, create a service request.

Include a description of the steps you have taken to resolve the error, along with a copy of the Workflow Monitor trace files with the following trace flags set:

```
Error Flags = 2

SQL Trace Flags = 2
```

For more information, see "Getting Help With A Workflow Error" on page 264.

Deleting an Obsolete Workflow Policy

Triggers are still in effect at the time you expire or delete policies and shut down Workflow Monitor, and triggers related to expired or deleted policies continue to work since they are attached to database tables until they are deleted from the database. So, between the time you expire policies and the time you finish dropping and recreating triggers, there can be rows triggered for the expired policies in the S_ESCL_REQ table.

You can get the error ESC-00054 while starting Workflow Monitor because WorkMon looks for the policies referenced in the S_ESCL_REQ table and finds policies that are inactive or not present. The policies are stored in the S_ESCL_RULE table. Those rows must remain in the S_ESCL_REQ table and must not be processed at all.

Given that Use Action Agent is set to FALSE for Workflow Monitor, WorkMon also acts for Action Agent, bypasses the S_ESCL_ACTN_REQ table, and uses only the S_ESCL_REQ table. If you have rows in the S_ESCL_REQ table for policies that are expired or deleted, then you must delete them before starting Workflow Monitor. Reasons for doing this include:

- To avoid the error ESC-00054 when starting Workflow Monitor.
- To avoid taking up unnecessary space. Normally, Workflow Monitor does not process the rows at all. The rows remain unused in the S_ESCL_REQ table.

You can delete the rows by RULE_ID, which is the unique identifier for a Workflow Policy. Back up the database prior to doing the delete.

Example Queries for Locating Deleted and Expired Policies

The following are queries that can help you determine whether you have rows in the S_ESCL_REQ table that are related to deleted or expired policies.

Query to Locate Deleted Policies

To locate deleted policies, run the following query:

```
select RULE_ID, count(RULE_ID)

from S_ESCL_REQ a

where not exists

(select row_id

from S_ESCL_RULE b

where a.RULE_ID = b.ROW_ID)

group by RULE_ID
```

Query to Locate Expired Policies

To locate expired policies run the following query, where sysdate is the Oracle current date-time function:

```
select a.RULE_ID, b.NAME, count (a.RULE_ID), b.EXPIRE_DT, b.SUB_TYPE_CD

from S_ESCL_REQ a, S_ESCL_RULE b

where a.RULE_ID = b.ROW_ID

and b.EXPIRE_DT < sysdate

group by a.RULE_ID, b.NAME, b.EXPIRE_DT, b.SUB_TYPE_CD
```

If you determine that the workflow policies have expired unintentionally and you need to process the rows, then the tasks you can perform include:

- If the Workflow Monitor is running and you do not need to shut it down, reverse the expiration of the policies by entering a date-time greater than the current date-time in the Expiration Date field for policies.
- After a period of 10 minutes, the policy takes effect. This is because the Reload Policy parameter of Workflow Monitor is set to 600 seconds as the default.
- If you need the policy to take effect immediately, then you can shut down Workflow Monitor, if it is running, reverse expiration of the policies, then start Workflow Monitor.

You can also take these actions with a policy that has a duration.

An example of policies expiring unintentionally is someone forgetting to extend the dates and nobody changed the triggers since the policies expired.

Tracing and Reporting a Workflow Policy

This topic describes how you can trace a workflow policy and generate reports about workflow policies.

About Workflow Policies and Siebel Server Task Trace Files

Whenever you start a Workflow Policies server process, a Siebel Server task trace file is created so that you can check for error messages and other information about the process. The Siebel Server processes for which trace files are created include:

- Generate Triggers.
- Page Manager.
- Email Manager.
- Workflow Monitor Agent.
- Workflow Action Agent.

You can view trace file information in either the Siebel client or the Siebel Server.

To view trace files in the Siebel client

In the Siebel client, navigate to Administration-Server Management > Tasks > Log.

The Tasks applet lists the status of the Siebel Server tasks as running or started.

To view trace files in the Siebel Server

Ways that you can view the log directory for the Siebel Server include:

- You can use Windows Explorer to navigate to your Siebel Server log directory. Under \log, choose the server's name to see a file that lists the trace files for each server process.
- You can double-click the Trace File icon to access the trace file. You can view the trace file for application server tasks.

For more information on using trace files, see *Siebel System Administration Guide*.

About Tracing and Event Log Levels

Table 76 describes events workflow policies use for logging.

Table 76. Description of Events for Workflow Policies Logging

Event	Level	Description
SqlParseAndExecute	4	Traces SQL statements and execution times.
Object Assignment	3	Traces Workflow Monitor Agent while it is doing Dynamic Assignment. Use in conjunction with Rules Evaluation.
Rules Evaluation	4	Traces Workflow Monitor Agent while it is doing Dynamic Assignment. Use in conjunction with Object Assignment.
Task Configuration	4	Parameter configuration of the server task.

NOTE: Setting trace levels above default parameters affects performance. Reset trace levels to default parameters after troubleshooting is finished.

About Workflow Policies Analysis Charts and Reports

Workflow policies provide several charts for analyzing how frequently a workflow policy condition is met and the total number of policy instances occurring in a specified period of time. Workflow policies also provide reports that summarize workflow policy and workflow log information.

Using the Policy Frequency or Trend Analysis Chart

Policy Frequency Analysis provides you with information about the number of times a Workflow policy executes. Policy Trend Analysis provides you with information about policy execution trends.

To view the Policy Frequency Analysis or Trend Analysis chart

Navigate to Administration-Business Process > Policy Frequency Analysis. Applets in this view include:

- **Monitor Log.** Lists the workflow policies.
- **Workflow Policy Frequency / Trend Analysis.** The Workflow Policy Frequency Analysis applet displays a chart illustrating the execution frequency of a chosen policy. The Workflow Policy Trend Analysis applet displays the total number of workflow policy conditions met over a specified period of time. To toggle between the two analysis applets, use the toggle list on the chart applet.

Using Workflow Policies Reports

In the Workflow Policies and Log views, you can bring up a Reports page that you can print out.

To bring up the report

Choose Reports from View.

The Reports page that appears provides summary information of the Workflow Policy.

If you must review the business rules for your organization, you can print out the Reports page for each of your workflow policies.

Converting a Workflow Policy to a Workflow Process

In general, it is recommended you keep workflow policies simple, placing most business logic into a workflow process. To this end, in many cases you can convert a workflow policy to a workflow process.

Table 77 describes guidelines you can follow when converting a workflow policy to a workflow process.

Table 77. Guidelines for Converting a Workflow Policy to a Workflow Process

To Replace This Configuration Used in a Workflow Policy . . .	. . .Use This Configuration in a Workflow Process
A specialized operator, such as IS UPDATED or IS ADDED.	A runtime event.
A standard operator, such as = or <>.	A condition branch or decision step.
A workflow policy program that performs an operation at the database layer.	A Siebel Operation step at the object layer that implements logic which manipulates data at the data layer. Although a workflow process cannot call a workflow policy program, if there is a need to perform an operation at the database level it might be possible to use the Siebel Operation step to accomplish the same required functionality.
(This cell left intentionally empty.)	For sending email, since a workflow process uses the communication outbound manager service, it can use recipient group and exact email address.

About Testing, Troubleshooting and Migrating a Workflow Policy

This topic describes testing, troubleshooting, and migrating a workflow policy. It includes the following topics:

Testing a Workflow Policy

Testing your workflow policy before implementing it in your production environment improves the likelihood that action recipients receive accurate and useful information, and that the overall results meet the workflow process requirements.

Correctly testing your workflow policies and eliminating problems is critical before implementing the policies in your production environment.

CAUTION: Your test environment and production environment must have identical versions of the software.

To test a workflow policy

1. Develop a testing and migration strategy for introducing changes into the production environment.

   Some of the considerations for creating a test and migration environment are discussed in "Planning a Test and Migration Strategy for a Workflow Policy" on page 183.

2. Make sure you have installed the Siebel Server workflow policy components on the Siebel Server.

   For more information, see *Siebel System Administration Guide*.

3. Make sure email and pager server processes required by your workflow policy are running.
4. Make sure your Workflow Agent processes are running.
5. Make sure the workflow policies, workflow policy conditions, and workflow policy actions perform as expected:
   - Test your workflow policies by entering data that meets the workflow policy conditions you defined in the policy.
   - Make sure the policies, conditions, and actions are correctly defined.
   - Make sure the policies, conditions, and actions accurately define the transactions.

     The correct columns should be monitored.

6 Make sure the actions perform as expected and occur when expected:

- Make sure your database triggers are created.
- Make sure the action interval and sleep times are correctly defined.
- Check your action by making sure the proper action executes. For example, you can check that the email arrives or the pager goes off. You can monitor Workflow Agent progress using the Workflow Policies Log view.

Using the Workflow Policy Log to Monitor Workflow Policy Execution

The Workflow Policy Log view displays a log of the records that meet a policy condition tracked by the Workflow Monitor Agent process. You access the Workflow Policy Log view from the Siebel client.

Fields contained in the view include:

- **Policy.** The name of the policy.
- **Workflow Object.** The name of the workflow policy object.
- **Object Identifier.** The ID of the workflow policy object for which the policy condition was met.
- **Object Values.** Identifying information for the row that met the policy condition.
- **Event Date/Time.** The date and time the policy condition was met.

Once you have verified that the workflow policies work as expected, you can migrate the workflow policies to your production environment.

Troubleshooting a Workflow Policy

Because a workflow policy is based on database triggers, a workflow policy can take effect on a database record only after the record is committed. If you have a policy that is based on multiple database tables, the policy takes effect only if the records on those tables are committed.

For example, Opportunity Revenue is stored in the S_OPTY_POSTN table, and lead quality is stored in the S_OPTY table. A policy with conditions Opportunity Revenue > 10M and Lead Quality is High takes effect only when the records are committed on both tables.

Also keep in mind that multiple business components can be created for the same database table using search specifications. If you are creating a workflow policy component to monitor a business component, be sure to include the fields that are being used in search specifications as workflow policy columns. The workflow policy column can then be used in the policy conditions to allow appropriate behavior to be enforced.

Troubleshooting a Workflow Policy That Does Not Trigger

You can troubleshooting a workflow policy action that does not trigger.

To troubleshoot a workflow policy action that does not trigger

1. Make sure your test record meets your workflow policy conditions.
2. Make sure the Siebel client configuration file is pointing to the correct enterprise server.

   One error that can occur if the server is incorrect is ESC-00053, *Error loading rule definitions*.
3. Check the workflow policy activation date/time.
4. Check the monitor task:
   a. Check to see if the monitor is awake and running against the correct group.
   b. Search the Task Information log for the ROW_ID of your test record:
      - If ROW_ID does not exist, run GENERATE TRIGGERS.
      - Update your test record.
5. Check the Action Agent task:
   a. Check to see if the Action Agent is awake and running against the correct workflow policy group.
   b. Search the Task Information log for the ROW_ID of the test record.
6. Make sure your triggers are generated.

Tracing a Workflow Policy

Workflow Policies uses the General Events event for logging. To view informational messages, set the log level to 3. To view debugging information, set the log level to 4. For more information, see "Tracing and Reporting a Workflow Policy" on page 308.

Migrating Workflow Policies to the Production Environment

To migrate fully tested policies to your production environment, you must follow a process similar to the one used for implementing the policies in your test environment.

To migrate workflow policies to the production environment

1. Back up your production environment database.
2. Migrate your test repository environment into your production repository environment.

   The process is described in the upgrade guide for the operating system you are using.
3. Reenter your workflow policy action types, workflow policies, and workflow policy groups in the test environment into the production environment.
4. When reentering definitions into the production environment, make sure you enter them exactly as they are defined in the test environment.

   Note that it is not necessary to reenter information that you have entered using Siebel Tools.

5 In the Siebel client, navigate to Administration-Server Management > Jobs.

6 In the Jobs list, click New.

7 From the Component/Job drop-down list, choose Generate Triggers.

This creates a new line entry but does not start the task.

8 In the Job Parameters list, click New to modify parameter settings.

For a description of the component-specific parameters for Generate Triggers, see "Administering Triggers on the Workflow Policy Server" on page 276. For a description of generic and enterprise parameters, see *Siebel System Administration Guide*.

9 Choose Submit Query.

For more information on trace files, see "Tracing and Reporting a Workflow Policy" on page 308. For more information on using trace flags, see *Siebel System Administration Guide*.

NOTE: To help prevent invalid triggers from being applied to your production environment, apply your database triggers to your test environment before you apply them to your production environment.

13 Example Workflow Processes

This chapter provides examples on which you can model your workflow process development efforts. It includes the following topics:

Example Workflow Process That Creates an Activity for a Sales Representative

This topic gives one example of using workflow to create an Activity record for a sales representative. You might use this feature differently, depending on your business model.

In this workflow, when a user creates a new opportunity record the opportunity's revenue is evaluated. If the value of the opportunity is over $10,000, an activity record is created that provides the sales representative a work item to follow up on the deal. The workflow is invoked by the WriteRecord run-time event.

The tasks you perform to implement the workflow described in this example include:

Defining The Workflow Process

Perform the procedures included in this topic to create a workflow process for this example. Make sure you perform the procedures in the order they are presented.

Creating a New Workflow Process Object Definition

To start creating a new workflow process, you first define a new workflow process object definition.

To create a new workflow process object definition

1. In the Object Explorer in Siebel Tools, choose the Project object type.
2. In the Projects Object List Editor (OBLE), create a new project named Workflow Examples.
3. Make sure the Locked property contains a checkmark.

   For brevity, note that example workflow processes in this book use the Workflow Examples project. In a development environment, you should use a project that most closely fits your development requirements. For more information, see *Using Siebel Tools*.
4. In the Object Explorer, choose the Workflow Process object type.

   For more information, see "About the Workflow Process Object Hierarchy" on page 65.
5. In the Workflow Processes OBLE, right-click then choose New Record to create a new workflow process object definition using values described in the following table:

Property	Value
Process Name	Large Opportunity Creates Activity
Workflow Mode	Service Flow
Business Object	Opportunity
Project	Workflow Examples

   For more information, see "Reference of Workflow Process Object Definition Properties" on page 422.
6. From the application-level menu, choose File > Save.

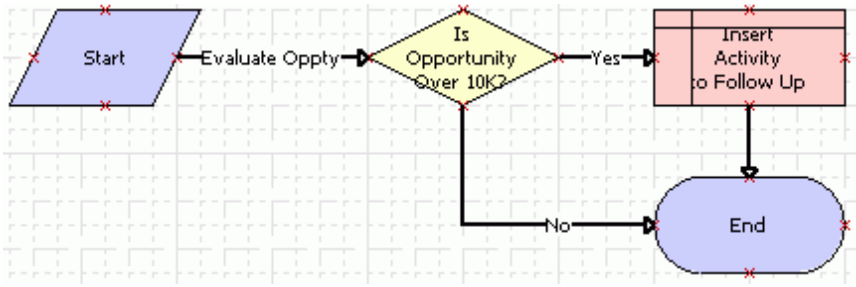
Adding Steps and Connectors to a Workflow Process

This topic describes how to add steps and connectors to a workflow process.

To add steps and connectors to a workflow process

1 In the Workflow Processes OBLE, right-click the record you created in "Creating a New Workflow Process Object Definition" on page 316, then choose Edit Workflow Process.

2 Add steps and connectors until your workflow process resembles the workflow illustrated in the following diagram:



For more information, see "About Workflow Process Steps and Connectors" on page 87, and "Diagramming a Workflow Process" on page 61.

3 Click the Decision Point, then use the Properties window to change the Name property to the value described in the following table:

Property	Value
Name	Is Opportunity Over 10k?

The Properties window is context-sensitive. When you click a step or connector in the design canvas, the step's or connector's properties are displayed in the Properties window.

If the Properties window is not visible, right-click a step, then choose View Properties Window.

4 Clicking each remaining step in succession, use the Properties window to change each step's Name property according to values described in the following table:

Step Type	Name Property
Siebel Operation	Insert Activity to Follow Up
End	End

5 Clicking each connector in succession, use the Properties window to change the connector's Name property according to values described in the following table:

Connector	Name Property
Between the Start Step and the Decision Point	Evaluate Oppty
Between the Decision Point and the Siebel Operation	Yes
Between the Decision Point and the End step	No

TIP: Note that the connector emanating from the Start step defaults to the name Connector 0. In some cases you can suppress this text. To suppress the text label for a connector, right-click the connector then choose Edit > Hide Text.

6 From the application-level menu, choose File > Save.

Next, define properties and arguments for workflow process steps.

Defining Properties and Arguments for a Workflow Process Step

In this topic, you define properties for workflow process steps.

To define properties for workflow process steps

1 Click the Insert Activity to Follow Up step, then use the Properties window set properties using values from the following table:

Property	Value
Business Component	Action
Operation	Insert

2 With the Insert Activity to Follow Up step still chosen, add an input argument in the MVPW using values from the following table:

Field Name	Type	Value
Description	Literal	This is a large opportunity. Please call the customer ASAP.

For more information, see "About Process Properties" on page 74.

Note that you must first specify the Field Name, then the Value.

3 Add a second process property using values from the following table:

Field Name	Type	Value
Type	Literal	To Do

Defining the Invocation Mechanism for a Workflow Process

In this topic, you define the run-time event that invokes the workflow process.

To define the run-time event that invokes the workflow process

1. Choose the No connector, then use the Properties window to make sure the Type property is set to Default.
2. Choose the Yes connector, then set the Type property to Condition.
3. In the design canvas, click the Evaluate Oppty connector then use the Properties window to set properties using values from the following table:

Property	Value
Event Object Type	BusComp
Event Object	Opportunity
Event	WriteRecord
Type	Default

When defining these properties, enter them in the top to bottom order displayed in the table. Since Event Object and Event are context sensitive, based on the value entered in Event Object Type, you must define Event Object Type first.

The business scenario dictates that when an opportunity's revenue is greater than $10,000, an activity record for the sales representative to follow up with the customer is inserted in the Activities view. You define this logic by using condition criteria on the decision point.

Next, you define the condition criteria for the workflow process.

Defining Condition Criteria for the Workflow Process

in this topic, you define the condition criteria for the workflow's Decision Point.

To define condition criteria for the workflow's Decision Point

1. In the design canvas, right-click the Yes connector, then choose Edit Conditions.

   The Compose Condition Criteria dialog box appears. Conditional logic can be defined for a connector emanating out of a Start Step, Decision Point, Wait Step, or User Interact Step. Note that condition criteria is defined on the connector emanating from a step. It is not defined directly on a step.

2. In the Compare To dropdown picklist, choose Business Component.

   The Compare To picklist is used to indicate whether an object, expression, or process property used in the comparison. In this example, the comparison is made against the run-time value of a field from the Opportunity business component, as specified in the Object picklist. Note that the items in the Object picklist are context sensitive: they change based on the value chosen in the Compare To picklist.

3 In the Object dropdown picklist, choose Opportunity.

4 In the Operation dropdown picklist, choose Greater Than.

The Operation picklist in this example is used to specify that the condition applies if the value defined in the Values section of the Compose Condition Criteria dialog is Greater Than the run-time value of the business component Revenue field. The Revenue field is specified in the Field picklist.

5 In the Field dropdown picklist, choose Revenue.

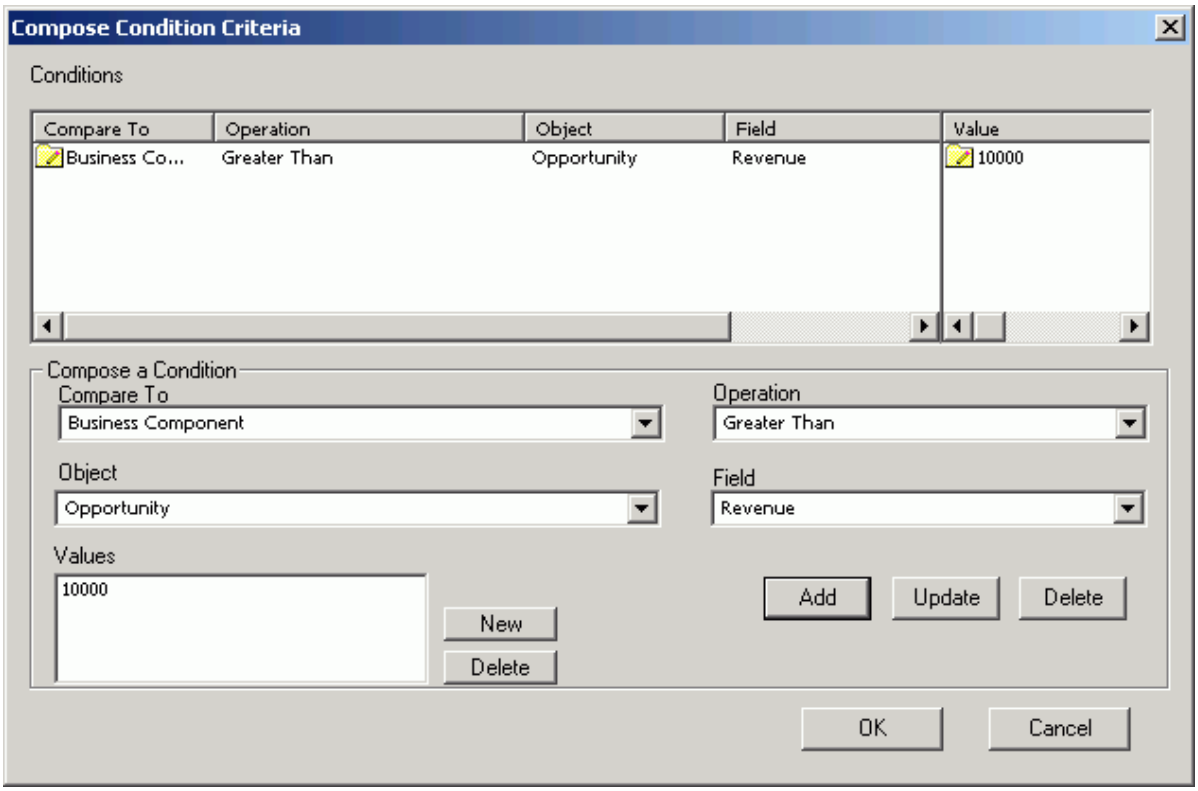
6 In the Values box, click the New button to add a new value.

7 In the Add Value dialog box, type 10000, then click OK.

8 In the Compose Condition Criteria dialog box, click the Add button.

The condition you defined in the lower portion of the Compose Condition Criteria dialog is added to the Conditions list in the upper portion of the dialog. At this point, you can add more criteria. For this example, you only add a single condition criteria.

The condition you set in the Compose Condition Criteria dialog box is displayed in the following image:



If the Revenue field on an Opportunity is greater than 10000, then the Yes branch is taken, otherwise the No branch is taken.

9 Click OK.

10 From the application-level menu, choose File > Save.

Modifications you can perform after defining a condition include:

- To update a condition, click the condition in the Conditions window of the Compose Condition Criteria dialog box, modify the condition, then click Update.
- To delete a condition, click the condition in the Conditions window of the Compose Condition Criteria dialog box, then click Delete.

For more information, see "Defining Conditional Logic on a Branch Connector" on page 114.

Next, you test the workflow process.

Testing the Workflow Process

In this topic, you test the workflow process you created in "Defining The Workflow Process" on page 316.

You must prepare this example for testing the workflow process by creating an opportunity that matches the test criteria. You create this opportunity, and note the opportunity record's row ID. This row ID is then used in the workflow process properties to run the test.

To prepare this example for testing

1 Validate the workflow process.

For more information, see "About the Validate Tool" on page 223.

2 Launch the Siebel client, then navigate to the Opportunities list applet.

3 Add an opportunity using values from the following table:

Name	Revenue
Large Opportunity to Test Workflow	$400,000

4 Right-click the record you just created, then choose About Record.

5 Note the value of the row Id in the Row # field.

6 In Siebel Tools, in the Process Designer, click the canvas background.

When you click an empty space on the background in the canvas, the Multi Value Property Window (MVPW) displays the process properties for the workflow process.

7 In the MVPW, find the Object Id process property, then set the property's Default String argument field to the Opportunity's Row Id you identified in Step 5.

Next, you simulate the process and examine simulation results.

To simulate the process then examine simulation results

1 Use the Process Simulator to step through the entire workflow process.

For more information, see "Preparing and Using the Process Simulator" on page 232.

2 In the Siebel client, navigate to the Opportunities screen.

3 In the Opportunities list applet, drill down on the Large Opportunity to Test Workflow opportunity name.

4 Click the Activities view tab.

A new Activity record should be present, and it should contain the description you specified in "Defining Properties and Arguments for a Workflow Process Step" on page 318. If so, the workflow process simulation finished successfully.

5 Remove modifications you made for testing:

a In Siebel Tools, close the Simulator.

b In the WF Process Props OBLE, choose the Object Id process property for your workflow process.

c In the Properties window, delete the row ID in the Default String property.

d In the WF Process Props OBLE, step off the record to save your changes.

Next, deploy and verify your workflow process.

Deploying and Verifying the Workflow Process

When you have finished a successful simulation of your workflow process you are ready to deploy it, then verify that the workflow implements the required functionality.

To deploy and verify a workflow process

1 Deploy the workflow process, with the following modifications:

- In the Siebel client, navigate to Administration-Business Process > Workflow Deployment.
- In the Repository Workflow Processes applet, query for the Large Opportunity Creates Activity workflow process.
- Click the Activate button.
- In the Active Workflow Processes applet, query for the Large Opportunity Creates Activity workflow process.

  Note that the record displays a status of Active.

For more information, see "Process of Deploying a Workflow Process" on page 237.

2 In the Siebel client, add a new opportunity record with a Revenue that is greater than $10,000.

3 Step off the record, return to the record, drill down on the record's name, then verify that an activity record associated with the opportunity is automatically added and that it includes the custom description you defined earlier in this example.

If you must modify the workflow process, in Siebel Tools use the Revise feature on the WF/Task Editor toolbar. For more information, see "About the WF/Task Editor Toolbar" on page 72.

Example Workflow Process That Traverses a Record Set to Close Obsolete Service Requests

This topic gives one example of closing a set of obsolete service requests. You might use this feature differently, depending on your business model. It includes the following topics:

The business case in this example identifies then closes obsolete service request records. A service request is considered obsolete if it is more than one year old with a Status that is not Closed or Cancelled.

The workflow process identifies obsolete service request records, then traverses the resulting record set, changing the Status for each obsolete service request record to Cancelled, the Sub-status to Resolved, the Close Date to the current date, and adds a message to the Description indicating the SR was cancelled due to obsolescence.

For more information about the technique used in this example, see "Using the Siebel Operation Step to Traverse a Record Set" on page 101.

Creating a New Business Component

Some requirements for this workflow process include:

- The workflow process must process a record from the primary business component of the business object on which the workflow process is based.
- A looping workflow process that processes a batch of records can loop only through the records of a non primary business component. This non primary business component must be based on the same business object that is defined for the workflow process. In this example, that business object is Service Request.
- The non primary business component must be established as a child of the primary business component's business object.

The Service Request business component is the primary business component of the Service Request business object. To satisfy these requirements you create a new, non primary business component, Service Request No Link. This new business component is a child of the primary Service Request business component.

The child business component contains the minimum number of fields required to meet the needs of this business case, and those fields point to the same table columns as their counterpart fields in the primary business component. Thus, when a field is modified in the child, the same field is modified in the primary business component. Both the primary and the child business component modify the same business component record.

A relationship must be established between the child and the primary business component's business object. To accomplish this, the child is defined as a Business Object Component on the primary business component's Business Object object definition.

To create the child business component

1. In the Object Explorer, click the Business Component object type.
2. In the Business Component OBLE, right-click then choose New Record.
3. Define the following properties for the new business component object definition. Accept default values for other properties, as described in the following table:

Property	Value
Name	Service Request No Link
Project	Workflow Examples
Class	CSSBCBase
Table	S_SRV_REQ

4. With Service Request No Link chosen in the Business Components OBLE, expand the Business Component object type in the Object Explorer, then click the Field object type.
5. In the Field OBLE, right-click then choose New Record to add each of the fields listed in the table below to the Service Request No Link business component. Accept default values for properties not described in the following table:

Name	Column	Text Length	Type
Status	SR_STAT_ID	30	DTYPE_TEXT
Sub-Status	SR_SUB_STAT_ID	30	DTYPE_TEXT
Closed Date	ACT_CLOSE_DT	(leave empty)	DTYPE_UTCDATETIME
Description	DESC_TEXT	2,000	DTYPE_TEXT

Fields in the child business component correspond to fields with the same name in the primary business component. Since a field on the child references the same table and column as the field's counterpart on the primary business component, the field modifies the same record. Note that since Created is a system field, it is not explicitly defined as a field on the primary business component, and is therefore not defined on the child.

TIP: To reduce the number keystrokes you must perform, define the Column property first. When you define the Column property, then step out of the Column field in the OBLE, the Text Length and Type properties automatically populate.

Next, establish a relationship between the child and the primary business component's Business Object.

To establish a relationship between the child and the primary's business object

1. In the Object Explorer, click the Project object type.
2. In the Project OBLE, locate then click the Service project.
3. Make sure the Locked property for the Service project contains a check mark.
4. In the Object Explorer, click the Business Object object type.
5. In the Business Object OBLE, query for the Service Request business object.
6. In the Object Explorer, expand the Business Object object type to expose the Business Object Component object type.
7. In the Object Explorer, click the Business Object Component object type.
8. In the Business Object Components OBLE, right-click then choose New Record.
9. Set the Bus Comp property to Service Request No Link. Do not specify a link.
10. From the Tools application-level menu, Choose Tools > Compile Projects.
11. Choose Locked Projects, then compile to the repository that is used by your sample database.

    This is typically [Siebel client root directory]\OBJECTS\[language]\siebel.srf.

Defining the Workflow Process

Actions this workflow process performs include:

- Accepts a record of the Service Request business component. This record is not processed.
- Identifies obsolete Service Requests by querying the Service Request No Link business component for records whose Created date is more than one year old and whose Status is not Closed or Cancelled.
- Traverses records in the query result set, modifying the Status, Sub-Status, Close Date and Description fields to reflect the record's obsolete condition.

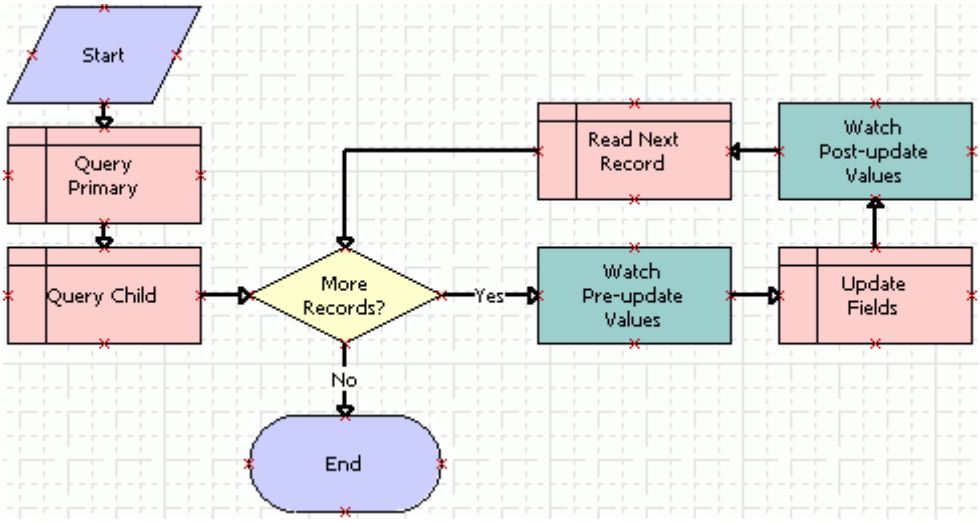
To create the workflow process

1. In Siebel Tools, create a new workflow process object definition using values described in the following table:

Property	Value
Process Name	Close Obsolete SRs
Workflow Mode	Service Flow
Business Object	Service Request

To view an example, see "Creating a New Workflow Process Object Definition" on page 316.

2 Add steps and connectors until your workflow process resembles the workflow illustrated in the following diagram:



The two business service steps allows you to use the Watch window to debug the workflow. These are removed near then end of the configuration instructions for this example.

For more information, see "About Workflow Process Steps and Connectors" on page 87, and "Diagramming a Workflow Process" on page 61.

3 Click anywhere on the canvas background to display the MVPW for the workflow process.

4 Add seven new process property records using values described in the following table:

Name	In/Out	Business Object	Data Type
SR Close Date	In/Out	Service Request	String
SR Created Date	In/Out	Service Request	String
SR Description	In/Out	Service Request	String
SR Status	In/Out	Service Request	String
noRecord	None	Service Request	String
numAcceptedRecords	None	Service Request	Number
vRowId	None	Service Request	Number

The first four process properties in the table, SR Close Date, SR Created Date, SR Description and SR Status, are defined to provide a basis for using the Watch window in this example. They are not required to use the traverse record set technique. The last three records in the table, noRecord, numAcceptedRecords, and vRowId, are required to use the traverse record set technique. For more information, see "About Process Properties" on page 74.

TIP: To simplify creating a process property, create one process property record then fully define its properties. Right-click this newly defined process property then choose Copy Record. A new process property record is created with the same values as the original process property record. For the new record, modify only those properties that require modification from the original.

5 Choose File > Save.

Next, define the Query Primary workflow step.

To define the query primary workflow step

Click the Query Primary workflow step, then use the Properties window to define values described in the following table:

Property	Value
Business Component	Service Request
Operation	Query

Next, define the Query Child workflow step.

To define the query child workflow step

1 Click the Query Child workflow step.

2 Use the Properties window to define values described in the following table:

Property	Value
Business Component	Service Request No Link
Operation	Query

3 With the Query Child workflow step still chosen, click the MVPW, then click the Search Spec Input Arguments tab. Add a new record using values described in the following table:

Argument Field	Value
Expression Business Component	Service Request No Link
Filter Business Component	Service Request No Link
Search Specification	"([Created] < Timestamp()-365.0) AND ([Description] = 'Test SR') AND ([Status] <> 'Closed') AND ([Status] <> 'Cancelled')"

Enter the argument for the Search Specification exactly as displayed. The entire specification must be enclosed within double quotes. Each search string must be enclosed with single quotes. A space is required before and after operators, such as the equal sign (=) and not equal to (<>).

4 Click the Output Arguments tab. Add a new record using values described in the following table:

Argument Field	Value
Property Name	NumAcceptedRecords
Type	Output Argument
Output Argument:	NumAffRows

The Query operation uses NumAffRows as a container to hold the total number of rows returned from the query. In this example, the value in NumAffRows is stored in NumAcceptedRecords, a process property, which makes the value available later in the workflow on the Yes connector to trap for the case where no records are returned in the query.

5 Add a new output argument using values described in the following table:

Argument	Value
Property Name	noRecord
Type	Literal
Value	false

You initialize noRecord at this point in preparation for the Yes connector downstream of the decision point. When one or more records are returned in the query, the Yes condition uses the value in noRecord to check for the end of the record set being processed by the Read Next Record step.

6 Add a new output argument using values described in the following table:

Argument	Value
Property Name	vRowId
Type	Business Component
Business Component Name	Service Request No Link
Business Component Field	Id

VRowId provides a way to display the record number in the Watch window while the workflow processes each record in the record set. It is used for debugging purposes in this example. The traversing a record set technique does not mandate this variable.

Next, define conditional logic for the decision point.

To define conditional logical for the Decision Point

1 Click the Yes connector.

2 In the Properties window , set the Type property to Condition.

3 Right-click the Yes connector, then choose Edit Conditions.

4 Define the following condition, then click Add:

Property	Value
Compare To	Process Property
Operation	Greater Than
Object	NumAcceptedRecords
Value	0

This condition traps for the instance where no records were returned from the query. If this condition is met, the workflow process takes the *No* connector.

5 Define the following condition, then click Add:

Property	Value
Compare To	Process Property
Operation	All Must Match (Ignore Case)
Object	noRecord
Value	false

This condition determines when the workflow has reached the end of the record set in cases where at least one record was returned in the query. For more information, see "Defining Conditional Logic on a Branch Connector" on page 114.

6 Click OK.

7 Click the No connector. Make sure the Type property in the Properties window is set to Default.

Next, define the Watch Pre-update Values workflow step.

To define the watch pre-update values workflow step

1 Click the Watch Pre-update Values workflow step.

2 Use the Properties window to define values described in the following table:

Property	Value
Business Service Name	Workflow Utilities
Business Service Method	Echo

The Workflow Utilities business service provides a way to view variables in the Watch window. For more information, see "About the Workflow Utilities Business Service" on page 382.

3 In the MVPW, click the Output Arguments tab. Add four new output arguments using values described in the following table:

Property Name	Type	Business Component Name	Business Component Field
SR Close Date	Business Component	Service Request No Link	Closed Date
SR Description	Business Component	Service Request No Link	Description
SR Created Date	Business Component	Service Request No Link	Created
SR Status	Business Component	Service Request No Link	Status

This workflow step provides data the Workflow Utility business service uses to display field values for the business component record currently being processed. These values are displayed in the Watch window. Values displayed reflect their state prior to being updated.

Depending on your testing requirements, you can display more descriptive record data, such as data from the Abstract field. To display other record data, create another Field object definition, similar as you did with fields described in Step 5 on page 325. In the workflow, define a process property that references that field, then add an Output Argument on the Workflow Utility business service that references the new process property.

Next, define the Update Fields workflow step.

To define the update fields workflow step

1 Click the Update Fields workflow step.

2 Use the Properties window to define values described in the following table:

Property	Value
Business Component	Service Request No Link
Operation	Update

3 In the MVPW, click the Field Input Arguments tab. Add three new input arguments using values described in the following table:

Field Name	Type	Value
Closed Date	Expression	Timestamp()
Description	Literal	This SR's Status set to Cancelled due to this SR's obsolescence
Status	Literal	Cancelled

Next, define the Watch Post-update Values workflow step.

To define the watch post-update values workflow step

1 Click the Watch Post-update Values workflow step.

2 Use the Properties window to define values described in the following table:

Property	Value
Business Service Name	Workflow Utilities
Business Service Method	Echo

3 In the MVPW, click the Output Arguments tab. Add four new output arguments using values described in the following table:

Property Name	Type	Business Component Name	Business Component Field
SR Close Date	Business Component	Service Request No Link	Closed Date
SR Description	Business Component	Service Request No Link	Description
SR Created Date	Business Component	Service Request No Link	Created
SR Status	Business Component	Service Request No Link	Status

Next, define the Read Next Record workflow step.

To define the read next record workflow step

1 Click the Read Next Record workflow step.

2 Use the Properties window to define values described in the following table:

Property	Value
Business Component	Service Request No Link
Operation	NextRecord

3 In the MVPW, click the Output Arguments tab, then add a new output argument using values described in the following table:

Argument Field	Value
Property Name	noRecord
Type	Output Argument
Output Argument	NoMoreRecords

4 Add a second argument using values described in the following table:

Argument Field	Value
Property Name	vRowId
Type	Business Component
Business Component Name	Service Request No Link
Business Component Field	Id

5 From the Tools application-level menu, choose File > Save.

Next, you use the Process Simulator and Watch window to test the workflow.

Using the Process Simulator and the Watch Window to Test the Workflow

In this example, you first modify several service request records by using the Siebel client to delineate a set of test records the workflow process traverses. Then, you use the Process Simulator and Watch window to test the workflow process.

For more information, see "About the Process Simulator Watch Window" on page 228.

CAUTION: Although you are only simulating the workflow, the Update Fields step in this workflow process modifies records in the query results. It is important to remain cognizant of the fact that the traversing a record set technique processes a set of records. If you have a step in the workflow that modifies record values, the *values for most if not all of the records in the database that meet the search criteria are modified*.

To prepare service request test records

1 Log into the Siebel client, connected to the Sample database.

2 Navigate to Service Requests > Service Request List.

3 Locate three records that have an Opened date that is more than one year from the date that the workflow process is tested. For example, if you test the workflow on February 15, 2008, make sure the Opened date for each service request is prior to February 15, 2007.

4 Modify the three records so that there are three service requests in the database that have a Description of Test SR, and that one request is Open, one Closed and one Cancelled. Use the values described in the following table:

Status	Description
Open	Test SR
Closed	Test SR
Cancelled	Test SR

These modifications provide a way to test the workflow process on a small, constricted set of records.

5 Create a new service request. Set Status to Open and enter Test SR in the Description.

6 Log out of the Siebel client.

Next, test the workflow process.

To test the workflow process

1 Validate the workflow process.

For more information, see "Validating the Workflow Process" on page 231.

2 In Siebel Tools, click anywhere in the background of the Process Designer, right-click, then choose Simulate.

For more information, see "Preparing and Using the Process Simulator" on page 232.

3 Click the Start Simulation button. Wait for the Siebel client to launch, control to return to Siebel Tools, and for the simulator to automatically highlight the Query Primary workflow process step.

For caution information, see "Using the Process Simulator and the Watch Window to Test the Workflow" on page 334.

4 From the Tools application-level menu, choose View Debug Windows > Watch, then click the push pin to dock the window.

Note that you must initiate the simulation, as described Step 3, prior to opening the Watch window.

5 Click the expand icon located next to PS:Property Set, then expand the Process Properties entry in the Property Set list.

Process properties defined for this workflow process are displayed along with each property's current value. To check this list, stop the simulation then compare the list of properties in the Watch window to the list of records in the MVPW.

No values are displayed in the Process Properties section of the Watch window until after the simulator calls the Workflow Utilities business service when the simulator executes the Watch Pre-update Values step.

6 Click the Simulate Next button, then expand the BusComp entry in the Watch window.

The simulator executes the Query Primary Step, then highlights the Query Child step.

The Query Primary step performs an unrestricted query on the Service Request business component, returning every service request record that exists in the database.

The Watch window now displays a BusComp entry that contains a list of record values for the current business component record.

7 Click the Simulate Next button.

The simulator executes the Query Child step, then highlights the More Records decision point.

The search specification on the Query Child workflow process step results in the creation of a record set that contains four service request records. In essence, Query Child populates the Service Request No Link business component with these metadata records.

In addition to several system variables, the Process Properties section of the Watch window now contains values for two process properties that reference important variables used with the traversing a record set technique:

Property	Value	Explanation
NumAcceptedRecords	1	NumAcceptedRecords is a process property that references NumAffRows, which is a variable used with the traversing a record set technique. NumAffRows maintains a count of the number of rows returned from the query on the Query Child workflow step. In this example, one record met the conditions of the search specification defined on the Query Child step.
noRecord	false	noRecord is a process property that references NoMoreRecords, which is a variable used with the traversing a record set technique. NoMoreRecords is a flag that indicates if records remain to be processed in the record set. The NextRecord operation sets NoMoreRecords to *true* when there are no more records to traverse in the record set. For every iteration through the record set, the NextRecord operation on the Read Next Record workflow step writes the value of NoMoreRecords to the noRecord process property.

The BusComp section of the Watch window displays only record data from the query that was performed on the primary business component. This information does not change for the remainder of the simulation.

8 Click the Simulate Next button.

The simulator executes logic associated with the More Records decision point, then highlights the Watch Pre-update Values step.

The *Yes* connector performs two tests to determine if flow continues or halts. A value of *0* in NumAcceptedRows indicates no records matched the search specification on the Query Child workflow step. A value of *true* in noRecord indicates there are no more records to process in the record set. If either case is true, the workflow process ends.

Since the value for NumAcceptedRecords is 1 and noRecord is false for your workflow, the Yes connector is taken.

9 Click the Simulate Next button.

The simulator executes the Watch Pre-update Values step, then highlights the Update Fields step.

Several values under the Process Properties entry in the Watch window populate when this step is executed. Four service request fields that are modified during the example have been defined. Writing to the Watch window both before and after the Update Fields workflow step allows you to clearly determine if the business component fields are updated appropriately.

Property	Value	Explanation
SR Close Date	(empty)	Since the search specification on the Query Child workflow step excludes Closed records, the Close Date is empty.
SR Description	Test SR	Your test records include Test SR in the Description. This value is provided here to make sure only those records in test record set are evaluated.
SR Status	Open	The search specification on the Query Child workflow step excludes Closed and Cancelled service requests, but allows Open, Pending, Field Visit, and Quoted. However, there is only one record in the resulting record set, and its value is Open.
noRecord	false	Although there is only one record in the record set, the NextRecord operation has not yet been executed, so noRecord contains false.

10 Click the Simulate Next button.

The simulator executes the Update Fields workflow step, then highlights the Watch Post-update Values workflow step.

Although the Update Fields workflow step updated three service request fields, changes to values in those fields are not reflected until the workflow explicitly writes the updated information to the Watch window.

11 Click the Simulate Next button.

The simulator executes the Watch Post-update Values workflow step, then highlights the Read Next Record workflow step.

The Watch window displays the updated values for the business component record:

Property	Value	Description
SR Close Date	11/12/2007 14:21:04	Set by the Input Argument on the Update Fields process step.
SR Description	This SR's Status set to Cancelled due to this SR's obsolescence	Set by the Input Argument on the Update Fields process step.
SR Status	Cancelled	Set by the Input Argument on the Update Fields process step.
noRecord	false	NextRecord operation has not yet been executed, so noRecord still contains false.

12 Click the Simulate Next button.

The simulator executes the Read Next Record workflow step, then highlights the More Records decision point. In the Watch window, values for the process properties remain constant except for noRecord. Since there is only one record in the record set, the NextRecord operation sets noRecord to *true*.

13 Click the Simulate Next button.

The simulator executes the logic associated with the Decision Point, then highlights the End workflow step.

Since this simulation modifies records in the record set, to run the simulation again you must first reset values that were changed in the record set. In this case, for the record that the workflow process modified, you must reset SR Description to SR Test, and reset Status to Open.

This example tests a small set of service request records. For a more thorough test, create more service request records that meet or do not meet the conditional logical defined in the workflow process. Simulate the workflow again, using the Watch window to confirm the modifications accurately reflect the workflow's required outcomes.

Preparing the Workflow for Production

To prepare the workflow for a production environment, you can remove some of the configuration that is specific to supporting debugging activities.

CAUTION: The traversing a record set technique has the potential to modify a large number of records in your database. To avoid unneeded modifications to numerous records, when using this technique be very careful your workflow is fully tested and achieves the required results. In this example, removing the Description condition from the Search Specification broadens the search to consider and possibly modify every Service Request that is over one year old and is not Closed or Cancelled.

To prepare the workflow process for a production environment

1. Delete the Watch Pre-update Values and Watch Post-update Values workflow steps.
2. In the Multi Value Properties Window, you can delete SR Close Date, CR Created Date, SR Description, and SR Status.
3. Click the Query Child workflow step.
4. In the Multi Value Properties Window, click the Search Spec Input Arguments tab.
5. In the Search Specification, delete the Description string:

Test Search Specification	Production Search Specification
"([Created] < Timestamp()-365.0) AND ([Description] = 'Test SR') AND ([Status] <> 'Closed') AND ([Status] <> 'Cancelled')"	"([Created] < Timestamp()-365.0) AND ([Status] <> 'Closed') AND ([Status] <> 'Cancelled')"

   For caution information, see the caution at the beginning of this topic.

6. If required, restore your test records to their original data.
7. If you made modifications to the Service Request No Link business component, such as adding new fields, compile your work again.
8. Identify then implement a way to invoke the workflow process.

   This example is based on a maintenance administrative task that should occur on a regularly scheduled interval, such as daily, weekly or monthly. One way to invoke it is to configure a repeating component job for the Workflow Process Manager component. For more information on how to configure a component to run repeatedly at specific times over specific intervals, see "About the Repeating Component Request" on page 166.

Example Workflow Process That Attaches an Activity Plan to an Opportunity

This topic gives one example of using workflow to attach an activity plan to an Opportunity record. You might use this feature differently, depending on your business model. It includes the following topics:

In this example, you define a workflow that creates a child activity plan for an opportunity record, then navigates the user to a view that displays the new plan. The workflow then waits for the user to enter more data and save changes before continuing.

You use the Process Simulator and the Siebel client to test the workflow.

Creating the New Workflow Process

This topic describes how to create a workflow process that attaches an activity plan to a large opportunity. You create a new workflow process object definition, define step properties, then validate the workflow.

To create a new workflow process object definition

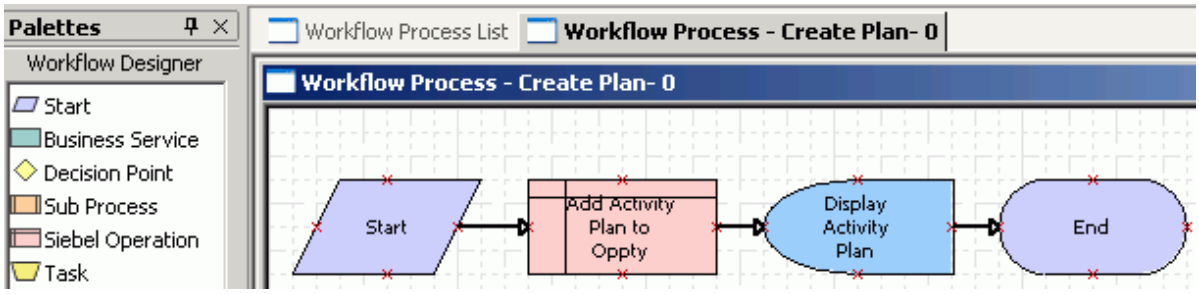
1 In Siebel Tools, create a new workflow process object definition using values described in the following table:

Property	Value
Process Name	Create Plan
Business Object	Opportunity
Workflow Mode	Interactive Flow

To view an example, see "Creating a New Workflow Process Object Definition" on page 316.

2 In the Workflow Processes OBLE, right-click the new workflow process object definition, then choose Edit Workflow Process.

3 Add steps and connectors until your workflow process resembles the workflow illustrated in the following diagram:



For more information, see "About Workflow Process Steps and Connectors" on page 87, and "Diagramming a Workflow Process" on page 61.

Next, define the workflow step properties.

To define workflow step properties

1. In the Process Designer canvas, click the Add Activity Plan to Oppty step.
2. In the Properties window, define the business component to use for the operation and the operation to be performed by entering values described in the following table:

Property	Value
Business Component	Activity Plan
Operation	Insert

When you perform an Insert operation on a business component, you must supply a value for required fields in the business component. In particular, to insert a new activity plan, you must provide the name of the activity plan template.

3. With the Add Activity Plan to Oppty step still chosen, in the MVPW add a new input argument using values described in the following table:

Field Name	Type	Value
Template	Literal	A valid activity template name, such as Introductory Sales Call.

For more information, see "About Process Properties" on page 74.

4. In the Process Designer canvas, choose the Display Activity Plan step.
5. Use the Properties window to define which view is displayed for the Display Activity Plan user interact step. Use values described in the following table:

Property	Value
User Interact View	Opportunity Activity Plan

6 Click the connector located between the Display Activity Plan step and the End step, then define properties in the Properties window using values described in the following table:

Property	Value
Event Object Type	BusComp
Event	WriteRecordUpdated
Event Object	Activity Plan
Type	Condition

Defining a run-time event signals the end of the Display Activity Plan user interact step. This is achieved by defining a conditional branch emanating from the Display Activity Plan step.

For this example, you need to wait for the user to make and save a change to the Activity Plan record before continuing the workflow process. WriteRecordUpdated is the run-time event that signals this change.

For the Event and Event Object picklists to populate correctly for this example, you must specify the Event Object Type first.

Next, test the workflow process.

Testing the Workflow Process

In this topic, you validate and simulate the workflow process you created in "Creating the New Workflow Process" on page 340.

Because this example workflow is based on the Opportunity business object and the workflow attempts to add an activity plan to an existing opportunity, you must specify the row Id for a specific opportunity to allow the Process Simulator to execute the workflow's logic.

Before you can test your workflow process, you create an opportunity that matches the test criteria. You create this opportunity, and note the opportunity record's row ID. This row ID is then used in the workflow process properties to run the test.

To prepare example data for the simulation

1 Launch the Siebel client, logging in as SADMIN/SADMIN to the Sample database.

2 Navigate to the Opportunities list applet.

3 Choose an Opportunity record, right-click, then choose About Record.

4 Note the value of the row ID in the Row # field, then click OK to close the About Record dialog box.

5 In Siebel Tools, in the MVPW, find the Object Id process property, then set the property's Default String field to the Opportunity row Id you identified in Step 4.

If the MVPW is not visible, from the application-level menu choose View > Windows > Multi Value Property Window.

6 Next, you test the workflow process.

To test the workflow process

1 Validate then simulate the workflow process.

For more information, see "Process of Testing a Workflow Process" on page 231.

2 Once you reach the user interact step in the simulation, use the Siebel client to make a change to the Activity Plan, then save your changes. For example, change values in the Planned Start Date field or the Description field.

Note that you cannot reach the End step by clicking the Next button once the Opportunity Activity Plan view is displayed in the Siebel client. Since you specified a condition on the branch, you must satisfy the condition to allow the workflow logic to proceed to the End step.

3 Acknowledge the message that displays at the end of the workflow.

When the last step is reached, the Siebel client displays a message reporting *Simulation terminated! Please check the Watch window for details.*

4 Enter Alt-Tab to return to Siebel Tools, then click Next to finish the End step.

5 In the Watch window, view the Simulator Status field.

If *Simulation ended successfully* is displayed, the workflow process ended without error.

Next, you configure an invocation method for the workflow process.

Configuring an Invocation Method for the Workflow Process

Now that you have built and simulated a workflow process, you must decide where and how it is invoked from the user interface. For this example, you add a button to the Opportunity List applet to invoke the workflow. For more information, see "Invoking a Workflow Process" on page 137.

To add a new button to the Opportunity list applet and configure it to invoke a new event

1 In Siebel Tools, choose the Applet object type in the Object Explorer.

TIP: If you have hidden the Object Explorer, type CTRL+E to re-expose it. You can also hide the Watch window.

2 In the Applets OBLE, choose the Opportunity List Applet applet.

3 Right-click then choose Edit Web Layout.

4 In the Controls/Columns window, set the Mode to 3: Edit List.

By default Mode is set to 1: Base. You must set it to 3: Edit List. If the Controls/Columns window is not visible, choose View > Windows > Controls Window.

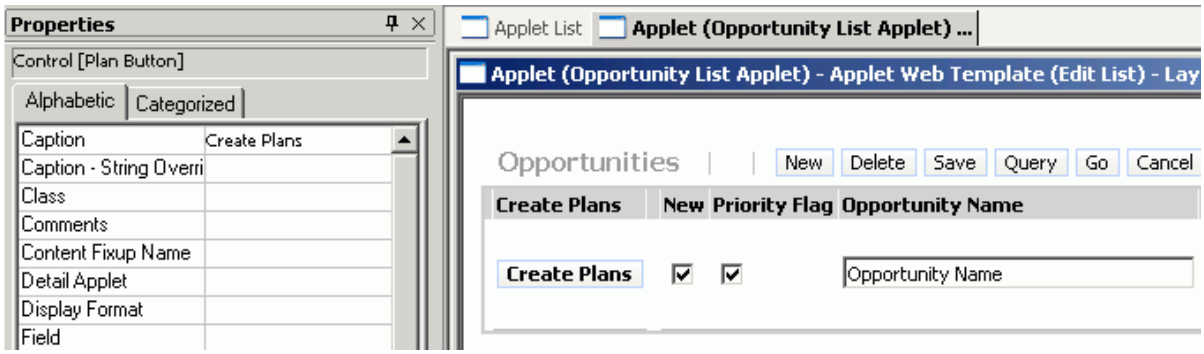
5 Drag a Button control From the palette to the layout, then set the control's properties using values described in the following table:

Property	Value
Caption-String Override	Create Plans
HTML Type	MiniButton
MethodInvoked	EventMethodCreateActivityPlan

The value displayed in the table does not appear in the MethodInvoked picklist. You must manually enter the value for MethodInvoked.

Methods that use the naming format EventMethod[a string value] do not require an applet or business component script to allow the event. If you do not use this special naming convention you must write a WebApplet_PreCanInvokeMethod script to allow the event.

6 Preview the applet. Make sure the new Create Plan button appears as displayed in the following image:



7 Save the applet changes.

8 Compile your changes to the SRF in your Siebel client directory.

Next, configure the workflow process to invoke when the run-time event is detected.

To configure the workflow process to invoke when the run-time event is detected

1 In the Object Explorer, choose the Workflow Process object.

2 In the Workflow Processes OBLE, query for the Create Plan workflow process you created in "Creating the New Workflow Process" on page 340.

3 Right-click the process then choose Edit Workflow Process.

4 Click the connector between the Start step and the step named Add Activity Plan to Oppty, then use the Properties window to set the connector's properties using values described in the following table:

Property	Value
Type	Condition
Event Object Type	Applet
Event Object	Opportunity List Applet
Event	PreInvokeMethod
Subevent	EventMethodCreateActivityPlan
EventCancelFlag	TRUE

This configuration supplies a start condition for the workflow so that the workflow is invoked whenever EventMethodCreateActivityPlan occurs on the Opportunity List Applet.

5 Validate then simulate the workflow process.

For more information, see "Process of Testing a Workflow Process" on page 231.

Next, deploy and activate the workflow process.

Deploying a Workflow Process Invoked By a Runtime Event

Before the new workflow process can be called in the Siebel client, you must deploy it.

To deploy the workflow process

Follow the procedures described in "Process of Deploying a Workflow Process" on page 237 with the following modifications:

1 In the Active Workflow Processes applet, set the Monitoring Level to 4-Debug.

2 Make sure you reload run-time events.

3 Make sure a run-time event was created for this workflow process:

a Navigate to Administration-Runtime Events > Events.

b Query the Subevent for `EventMethodCreateActivityPlan`.

This is the method you are using to invoke the workflow. One record should be returned. Next, you test the workflow process by using the Siebel Client.

Verifying Workflow Process Functionality

In this topic you verify that the workflow process implements the required functionality.

To verify the workflow process implements the required functionality

1. In the Siebel client, navigate to Opportunities > All Opportunities.
2. In the All Opportunities list, click the Create Plan button.

   This invokes the workflow process for an opportunity. The workflow process navigates you to the Opportunity Activity Plan view
3. Edit the plan Description field and save the change.
4. Navigate to Administration-Business Process > Workflow Instance Monitor.
5. In the Name field, query for the workflow process named Create Plan.

   The related instance data is displayed in the bottom applet.
6. Choose the Step Instances tab to view step instance details.

If your workflow process runs without error, you can turn off monitoring for the workflow process.

To turn off monitoring for the workflow process

1. Navigate to Administration-Business Process > Workflow Deployment.
2. In the Active Workflow Processes applet, query the name field for the workflow process named Create Plan.
3. Set the Monitoring Level field to 0-None.

Example Workflow Process That Manages Research Activities for a Service Request

This topic gives one example of invoking a workflow process by using a run-time event to assist with research activities associated with an SR. You might use this feature differently, depending on your business model. It includes the following topics:

The business case used in this example is that a workflow detects a service request being taken ownership of, then automatically creates an activity of type Research. The workflow then provides the SR owner with initial items that can be used in research.

In this example you create a simple workflow process that is initiated by a run-time event. You become familiar with elements involved in setting up this workflow process, and you see that the workflow process and run-time event works.

Creating the Workflow Process

This topic describes how to create the workflow process.

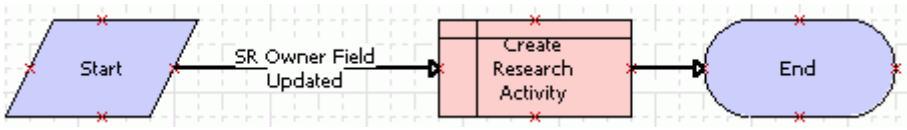
To create the workflow process

1 In Siebel Tools, create a new workflow process object definition using values described in the following table:

Property	Value
Process Name	SR Assigned-Auto Create Activities
Business Object	Service Request
Group	(Leave this field empty.)
Workflow Mode	Service Flow
Auto Persist	No
Description	Automatically generates a research activity when SR ownership change is detected by using RTE

To view an example, see "Creating a New Workflow Process Object Definition" on page 316.

2 Add steps and connectors until your workflow process resembles the workflow illustrated in the following diagram:



For more information, see "About Workflow Process Steps and Connectors" on page 87, and "Diagramming a Workflow Process" on page 61.

Next, define the workflow logic.

Defining the Workflow Logic

This topic describes how to define the workflow logic.

To define the workflow logic

1 Click the SR Owner Field Updated connector.

2 In the Properties window, define properties for the connector using values described in the following table:

Property	Value
Type	Condition
Event Object Type	BusComp
Event	SetFieldValue
Event Object	Service Request
Subevent	Owner

3 Click the Create Research Activity step, then use the Properties window to define values described in the following table:

Property	Value
Operation	Insert
Business Component	Action

Next, specify fields to populate when inserting a new business component record.

To specify fields to populate when inserting a new business component record

1 With the Create Research Activity step chosen, create two new input argument records using values described in the following table:

Field Name	Type	Value
Type	Literal	Research
Description	Literal	Research the following: Siebel Bookshelf

For more information, see "About Arguments for a Workflow Process Step" on page 76.

When performing inserts into a business component, make sure required fields for the business component are specified in this step. Alternative, specify values in the pre-default property for business component fields. To access these properties, expand the Business Component object type in the Object Explorer, then click the child Field object type.

2 From the application-level menu, choose File > Save.

Next, test and deploy the workflow process.

Testing and Deploying the Workflow Process

This topic describes how to deploy the workflow process.

To test and deploy the workflow process

1. Validate then simulate the workflow process.

   For more information, see "Process of Testing a Workflow Process" on page 231.

2. Navigate to the Workflow Processes OBLE.
3. In the Process Name property, query for the workflow process named SR Assigned-Auto Create Activities.
4. Publish and activate the workflow by completing the deploy workflow task.

   For more information, see "Process of Deploying a Workflow Process" on page 237.

5. If you must debug the workflow process, in the Siebel client, in the Active Workflow Processes applet, set the Monitoring Level to 4-Debug.

   For more information, see "Diagnosing a Failed Workflow Process in a Production Environment" on page 262.

Next, verify workflow process functionality.

Verifying Workflow Process Functionality

This topic describes how to verify the workflow process implements the required functionality.

To verify workflow process functionality

1. In the Siebel client, navigate to Service Requests > All Service Requests.
2. Create a new service request record then step off the record.
3. In the Service Request list applet, click the hyperlink in the SR # field.
4. Examine the Activities tab. If the workflow executed correctly, values displayed in the Activities list applet include:
   - Type: Research
   - Description: Will research the following: Siebel Bookshelf

Example Workflow Process That Manages Service Request Creation

This topic gives one example of using a workflow to manage a new Service Request. You might use this feature differently, depending on your business model. It includes the following topics:

In this example, tasks that the workflow automatically performs include:

- Detects when an end-user creates a new service request.
- Assigns the new service request to the creator.
- Changes the SR Sub-status to Assigned to reflect the ownership.
- Sets the commit time on the SR, depending on the priority.
- Creates an activity with a description of research steps the SR owner can follow to resolve the SR.

You create a simple workflow process that is initiated by a run-time event. You become familiar with elements involved in setting up this workflow process, and you observe how a workflow process works in conjunction with a run-time event to make several modifications to a service request record.

Creating the Workflow Process

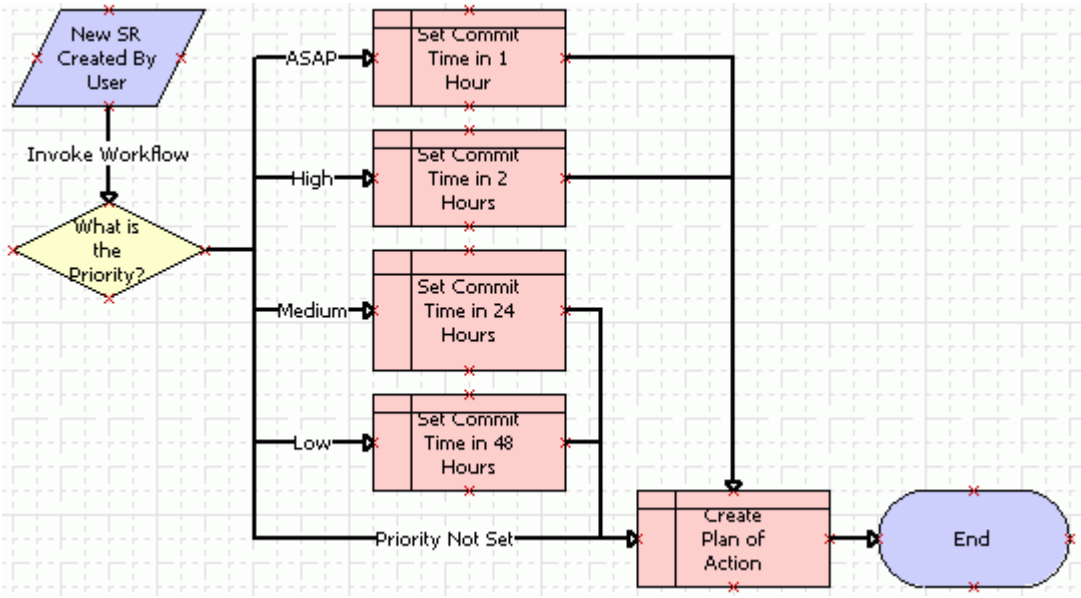
To create the workflow process

1 In Siebel Tools, create a new workflow process object definition using values described in the following table:

Property	Value
Process Name	Manage New SR
Business Object	Service Request
Group	(Leave this field empty.)
Workflow Mode	Service Flow
Auto Persist	No
Description	For new SRs, automatically modifies several SR fields. Considers SR priority during modifications.

To view an example, see "Creating a New Workflow Process Object Definition" on page 316.

2 Add steps and connectors until your workflow process resembles the workflow illustrated in the following diagram:



For more information, see "About Workflow Process Steps and Connectors" on page 87, and "Diagramming a Workflow Process" on page 61.

Next, define the connectors for this workflow process.

Defining Connectors for the Workflow Process

This topic describes how to define connectors for the workflow process.

To define the connectors for this workflow process

1 Click the Invoke Workflow connector, then use the Properties window to define values described in the following table:

Property	Value
Type	Condition
Event Object Type	BusComp
Event	WriteRecordNew
Event Object	Service Request

This configuration invokes the workflow process whenever a WriteRecordNew run-time event is detected on a Service Request.

2 Click each connector in succession that is emanating from the Decision Point.

In the diagram in Step 2 on page 351, these are labeled ASAP, High, Medium, and Low. As you click each connector, make sure the Type property in the Properties window is set to Condition.

3 Click the connector labeled Priority Not Set, then specify properties using values described in the following table:

Property	Value
Type	Default

4 Right-click the ASAP connector, then choose Edit Conditions.

5 In the Compose Condition Criteria dialog box, define the condition using values described in the following table:

Property	Value
Compare To	Business Component
Operation	All Must Match (Ignore Case)
Object	Service Request
Field	Priority
Value	1-ASAP

Conditions in this workflow are defined for the ASAP, High, Medium and Low connectors. Each conditional statement corresponds to a Priority value. Depending on how each priority evaluates in the priority connector condition logic, the workflow adjusts the Commit Time in the Subsequent Siebel Operation step.

For more information, see "Defining Conditional Logic on a Branch Connector" on page 114.

6 Right-click the High connector, then choose Edit Conditions to access the Compose Condition Criteria dialog box.

7 In the Compose Condition Criteria dialog box, define the condition using values described in the following table:

Property	Value
Compare To	Business Component
Operation	All Must Match (Ignore Case)
Object	Service Request
Field	Priority
Value	2-High

8 Right-click the Medium connector, then choose Edit Conditions to access the Compose Condition Criteria dialog box.

9 In the Compose Condition Criteria dialog box, define the condition using values described in the following table:

Property	Value
Compare To	Business Component
Operation	All Must Match (Ignore Case)
Object	Service Request
Field	Priority
Value	3-Medium

10 Right-click the Low connector, then choose Edit Conditions to access the Compose Condition Criteria dialog box.

11 In the Compose Condition Criteria dialog box, define the condition using values described in the following table:

Property	Value
Compare To	Business Component
Operation	All Must Match (Ignore Case)
Object	Service Request
Field	Priority
Value	4-Low

TIP: For clarity, in this example you are directed to create several connectors, placing them on the canvas consecutively. You can use an alternative technique when your workflow has multiple connectors, and where each of these connectors contain similar definitions.

In this example, you can create a single branch connector, define the conditional logic, then copy and paste this connector for use elsewhere in the workflow. You can then define the variance on each connector, in this case the Value. The Siebel Operation steps in this workflow can also be defined using the same cut and paste technique.

12 From the application-level menu, choose File > Save.

13 Close the workflow process designer.

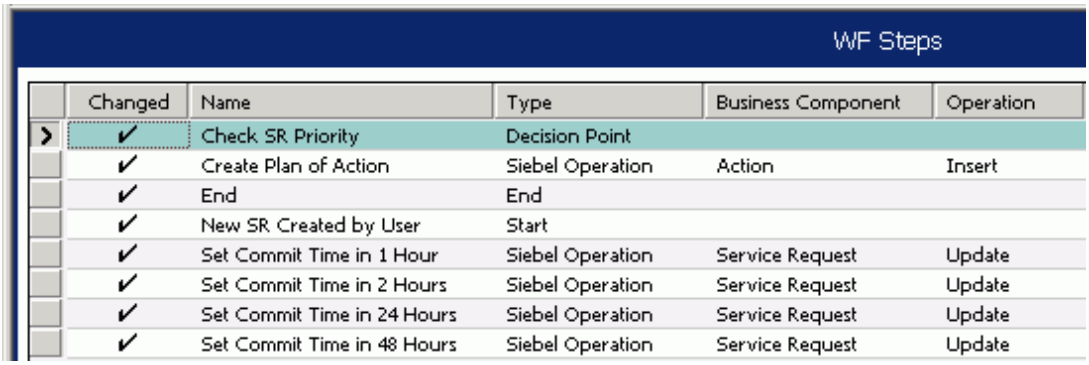
Next, define properties for the Siebel Operation steps.

Defining Properties for the Siebel Operation Steps

This topic describes how to define properties for the Siebel Operation steps.

To define properties for the Siebel Operation steps

1. From the application-level menu, choose View > Options.
2. Click the Object Explorer tab, then scroll down to the Workflow Process entry in the Object Explorer Hierarchy window.
3. In the Object Explorer Hierarchy window of the Development Tools Options dialog box, expand the Workflow Process hierarchy, then expand the WF Step hierarchy.
4. Make sure there is a check mark next to the WF Step I/O Argument hierarchy, then click OK.
5. in the Workflow Processes OBLE, query the Process Name property for the Manage New SR workflow process object definition.
6. Expand the Workflow Process object type in the Object Explorer (OE), then click the WF Step object type in the OE.
7. Right-click in the blue banner of the WF Steps OBLE, then choose Columns Displayed. Arrange columns to reflect the order displayed in the screen print in Step 8.
8. Set the Business Component property and the Operation property for each of the Siebel Operation steps in the workflow. The resulting properties must resemble those displayed in the following image:



Defining Properties in the Process Designer and the OBLE

Some workflow properties can be defined in the process designer or the OBLE, depending on your personal preference. In this procedure, you use the OBLE to define several properties for workflow steps. To define these properties by using the workflow process designer, you use the Properties window.

Next, define input arguments for the Set Commit Time Siebel operation steps.

To define input arguments for the set commit time Siebel Operation steps

1. In the Workflow Processes OBLE, locate then right-click the Manage New SR workflow process. Choose Edit Workflow Process to open the process designer.
2. Click the Set Commit Time in 1 Hour Siebel Operation step.

3 In the MVPW, create three new input arguments using values described in the following table:

Field Name	Type	Value	Business Component Name	Business Component Field
sub-status	Literal	Assigned	(Leave this field empty.)	(Leave this field empty.)
Owner	Business Component	(Leave this field empty.)	Service Request	Created By Name
Commit Time	Expression	[Created]+0.4166	(Leave this field empty.)	(Leave this field empty.)

For more information, see "About Process Properties" on page 74.

These arguments provide instructions to the Siebel Operation the actions the operation takes when the ASAP branch is satisfied. In this case, the sub-status field is set to Assigned, the Owner field is assigned to the value in Created By Name, and the Commit Time is set to one hour after the time the SR was created.

Commit Time has a Type property of DTYPE_UTCDATETIME, which is a datetime field. Created contains a time stamp of when the SR record was created. When a number is entered in a datetime field, days are represented by integers and hours, and minutes and seconds are represented by fractions. For more information, see "Using the Timestamp" on page 421.

NOTE: The entire expression in the Value property for the Commit Time Field Input Argument must be enclosed in double quotes. A space must precede and follow the plus sign (+).

4 Click the Set Commit Time in 2 Hours Siebel Operation step.

5 In the MVPW, create three new input arguments using values described in the following table:

Field Name	Type	Value	Business Component Name	Business Component Field
sub-status	Literal	Assigned	(Leave this field empty.)	(Leave this field empty.)
Owner	Business Component	(Leave this field empty.)	Service Request	Created By Name
Commit Time	Expression	[Created]+0.	(Leave this field empty.)	(Leave this field empty.)

6 Click the Set Commit Time in 24 Hours Siebel Operation step.

7 In the MVPW, create three new input arguments using values described in the following table:

Field Name	Type	Value	Business Component Name	Business Component Field
sub-status	Literal	Assigned	(Leave this field empty.)	(Leave this field empty.)
Owner	Business Component	(Leave this field empty.)	Service Request	Created By Name
Commit Time	Expression	[Created]+24	(Leave this field empty.)	(Leave this field empty.)

8 Click the Set Commit Time in 48 Hours Siebel Operation step.

9 In the MVPW, create three new input arguments using values described in the following table:

Field Name	Type	Value	Business Component Name	Business Component Field
sub-status	Literal	Assigned	(Leave this field empty.)	(Leave this field empty.)
Owner	Business Component	(Leave this field empty.)	Service Request	Created By Name
Commit Time	Expression	[Created]+48	(Leave this field empty.)	(Leave this field empty.)

Next, define Input Arguments for the Set Commit Time Siebel Operation steps.

To define input arguments for the create plan of action Siebel Operation step

1 Click the Siebel Operation step named Create Plan of Action.

This step creates an activity for the service request with a description that contains a set of actions the owner can follow.

2 In the MVPW, create two new input arguments using values described in the following table:

Field Name	Type	Value
Type	Literal	Research
Comment	Literal	Plan of Action: 1. Make sure customer logged description. 2. Reproduce the behavior. 3. Request logs.

3 Save your work, then close the Process Designer.

Next, deploy the workflow process.

Testing, Deploying and Verifying the Workflow Process

This topic describes how to test and deploy the workflow process, then verify the process implements the required functionality.

To test and deploy the workflow process

1. Validate then simulate the workflow process.

   For more information, see "Process of Testing a Workflow Process" on page 231.

2. Deploy the workflow process.

   For more information, see "Process of Deploying a Workflow Process" on page 237.

   While performing the deploy procedure, make sure you reload run-time events.

Next, verify workflow process functionality.

To verify workflow process functionality

1. In the Siebel client, navigate to Service Requests > All Service Requests, then create a new service request record.
2. Set a value for the Priority field, or leave it blank.
3. Save the record.

   When you step off the record, notice that the Siebel client delays slightly, and an hourglass is displayed. This indicates the workflow process is being executed.

4. Choose the service request record you just saved, then examine the Owner, Substatus and Date Committed fields. Verify they have been populated correctly.
5. Click the Activities tab.

   Notice that the new activity record created by the workflow process is displayed.

6. Examine the Comments field to verify it is populated according to the input arguments you specified on the Siebel Operation named Create Plan of Action.
7. Test out a few scenarios by creating service request records with different Priority values and see how the workflow process Decision Point behaves.

Example Workflow Process That Manages Service Request Creation then Navigates the User

This topic gives one example of using a workflow to manage a new Service Request and navigate the user to a view. You might use this feature differently, depending on your business model. It includes the following topics:

The business case used in this example is the same as the business case used in "Example Workflow Process That Manages Service Request Creation" on page 349, with one addition. In this example, the workflow automatically navigates the user to the Service Request > Activities view, where the user can peruse newly created activities associated with the service request.

Copying then Modifying an Existing Workflow Process

This topic describes how to copy and modify an existing workflow process.

To copy then modify an existing workflow process

1. Log into Siebel Tools as SADMIN/SADMIN connected to the sample database.
2. In the Object Explorer, choose the Workflow Process object type.
3. In the Workflow Processes OBLE, right-click the workflow process named Manage New SR, then choose Copy Record.

   This example assumes you have created and tested "Example Workflow Process That Manages Service Request Creation" on page 349.

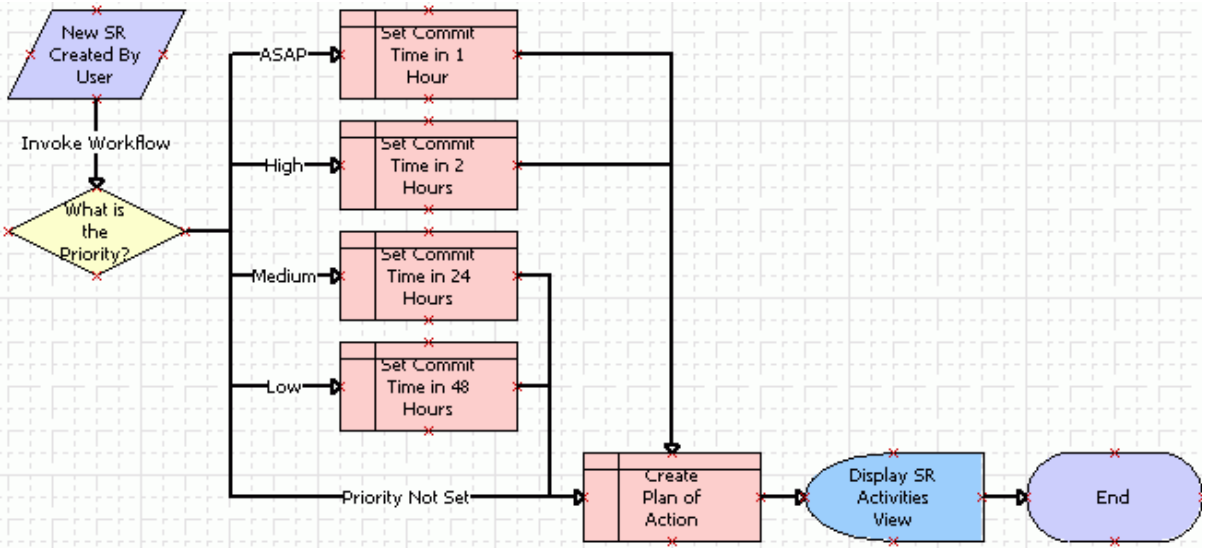
   In this example, you can use the Revise feature on the WF/Task Editor Toolbar instead of the copy feature. The Copy feature is used here in the event you need to continue working on the Manage New SR workflow at some point in the future without the additional modification that is defined here.
4. Set the properties for the new workflow process object definition:

Property	Value
Process Name	Manage New SR-Navigate User to SR Activity
Workflow Mode	Interactive Flow

5. Right-click the Manage New SR-Navigate User to SR Activity workflow process object definition, then choose Edit Workflow Process to open the Process Designer.

6 Drag then drop a User Interact step from the palette to the canvas, placing it between the Create Plan of Action and End steps.

7 Add connectors between the Create Plan of Action step and the User Interact step, and between the User Interact step and the End step, as illustrated in the following image:



8 Click the User Interact step then use the Properties window to set properties using values described in the following table:

Property	Value
Name	Display SR Activities View
User Interact View	Service Request Detail View

TIP: To confirm the value to set for the User Interact View property, in the Siebel client navigate to Service Requests > All Service Requests, then click the Activities view tab. From the application-level menu, choose Help > About View. The required view name is displayed.

9 Click the connector that connects the User Interact step with the End step, then use the Properties window to set properties using values described in the following table:

Property	Value
Type	Condition
Event Object Type	BusComp
Event Object	Service Request
Event	WriteRecordUpdated

10 From the application-level menu, choose File > Save.

Next, test and deploy the workflow process.

Testing and Deploying the Workflow Process

This topic describes how to test and deploy the workflow process.

To test and deploy the workflow process

1. Validate then simulate the workflow process.

   For more information, see "Process of Testing a Workflow Process" on page 231.

2. Deploy the workflow process.

   For more information, see "Process of Deploying a Workflow Process" on page 237.

Next, verify workflow process functionality.

Verifying Workflow Process Functionality

This topic describes how to verify the workflow process implements the required functionality.

To verify the workflow process implements the required functionality

1. In the Siebel client, navigate to Administration-Business Process > Workflow Deployment.
2. Deactivate previous workflow processes that have their business object set to Service Request. This way, other workflow processes do not run or interfere with invocation of the workflow process for this example:
   a. In the Active Workflow Processes applet, query the Business Object field for Service Request.
   b. From the application-level menu, choose Edit > Select All.
   c. Click the Menu button in the Active Workflow Processes applet, then choose Deactivate Process.
3. In the Repository Workflow Processes applet, query the Name field for the New SR-Take User to Activity workflow process.
4. Click the Activate button.
5. In the Active Workflow Processes applet, query the Name field for the New SR-Take User to Activity workflow process.

   One row is returned for every version that was activated.
6. Make sure only the latest version is active.
7. Set the Monitoring Level to 4-Debug.
8. Click the Menu button in the Repository Workflow Processes applet, then choose Reload Runtime Events.
9. Navigate to the Service Request list view.

10 Create a new service request record, specify a Priority level, then step off the record.

The application delays, displaying an hourglass. Then the Owner, sub-status, and Date Committed fields are populated, and a new activity record is created that is associated with the new service request.

Also, the application automatically navigates the user to the Service Request > Activities view to display the new activity record.

11 Confirm that a process instance exists for New SR Created-Take User to Activity in the Workflow Process Log view.

Example of Externalizing Properties Used by Siebel Workflow

This topic provides an example for externalizing properties. It includes the following topics:

The goal of this example is to provide a framework to externalize Workflow properties such that a workflow process using business services that interact with external systems does not require repository changes when migrating between Siebel instances in development, test, and production.

Concepts presented in this example can be extended for other business services provided by Siebel and can help avoid changes to workflows when migrating the repository between development, test, and production Siebel instances.

Externalizing Properties Overview

A workflow process that uses a business service that requires interaction with external systems must be changed when the repository is migrated between development, test, and production.

An example if this is EAI HTTP Transport, which requires the URL of the external system for the HTTPRequestURLTemplate input argument. The URL for the workflow process that uses the EAI HTTP Transport business service would point to an integrated test system's external URL in the testing instance and point to a production external system URL in the production instance.

The basic idea for the design of the framework is to store in a file the input arguments of a business service that are in use in a given workflow process. In the Service_PreInvokeMethod event of the Business Service step, add script to read values of these substitutable input arguments of a business service in use in a workflow process from a file, and set the Input Argument of the business service with values read from a file. This way, the file residing on the production Siebel instance server contains values for production external systems, and the file residing on the dev/test Siebel instance server contains values for dev/test external systems. In other words, input argument values are not hard coded for such a service in the workflow design, but read at run time from a file in the Service_PreInvokeMethod Event method script.

This framework relies on a file, with a particular name, on each Siebel Server being available for reading. Otherwise the business service fails and Siebel Workflow throws an error at that Business Service step, which is the required behavior.

For a business service where the script is added to the Service_PreInvokeMethod event, it is expected that an input argument called ExternalSystem is defined, and the argument's value matches the XML tag name of the child of the root element in the file.

This script code sets the child elements of the second-level parent, that is the child of the root element, of the file as input arguments.

Recommended Requirements for Externalizing Properties

Recommended requirements for externalizing Workflow properties for certain business services include:

- The framework makes transparent the migration of repository workflows between development, test, and production Siebel instances. That is, workflows no longer require changes during migration between various Siebel instances.
- The framework supports current business services as well as future requirements.
- The framework supports usage scenarios of a given business service used in various workflows.
- The framework is easy to manage and has a net effect of reducing the total cost of ownership of operational aspects of running the Siebel instance.

About the EAI Business Service Input Arguments

The EAI HTTP Transport and EAI SQL Adapter business services, when used in the workflow process as a business service step, have input arguments that might require changes when migrating the repository between development, test, and production Siebel instances. Table 78 describes potential required changes.

Table 78. Description of Input Arguments That Might Require Changes When Migrating Repository Data

Business Service	Business Service Methods	Business Service Method Input Arguments
EAI HTTP Transport	Send SendReceive	HTTPRequestURLTemplate
EAI SQL Adapter	Query	ExtDBODBCDataSource ExtDBPassword ExtDBTableOwner ExtDBUserName

Assume that a given Siebel instance has integration points with third-party HR and Finance systems. To use the EAI HTTP Transport and EAI SQL Adapter business services, the input arguments listed in Table 78 must be supplied in the definition of the workflow process business service step where the business service is referenced.

Example XML Hierarchy

The following is an arbitrary XML Hierarchy file containing input arguments for the EAI HTTP Transport and EAI SQL Adapter business services, and their values. The parameters under Finance and HR are for illustration purposes only. The file is named ebizint.xml.

You can have a number of parameters with appropriate values. For a given usage scenario of a business service, parameters that are not needed do no harm. Also, this XML Hierarchy is arbitrary. The hierarchy displayed below is for illustration purposes only:

```
<?xml version="1.0" encoding="UTF-8" ?>

<Systems>

<Finance>

<TargetWebAddress>http://finance</TargetWebAddress>

<ExtDBODBCDataSource>financesource</ExtDBODBCDataSource>

<ExtDBUserName>financeuser</ExtDBUserName>

<ExtDBPassword>financepassword</ExtDBPassword>
```

```
<ExtDBTableOwner>financeowner</ExtDBTableOwner>
</Finance>
<HR>
<TargetWebAddress>http://hr</TargetWebAddress>
<ExtDBODBCDataSource>hrsource</ExtDBODBCDataSource>
<ExtDBUserName>hruser</ExtDBUserName>
<ExtDBPassword>hrpassword</ExtDBPassword>
<ExtDBTableOwner>hrowner</ExtDBTableOwner>
</HR>
</Systems>
```

Example eScript

Given the structure of the example XML file displayed in "Example XML Hierarchy" on page 363, the following eScript code example can be added to the Service_PreInvokeMethod event of the EAI HTTP Transport and EAI SQL Adapter business services:

```
function Service_PreInvokeMethod (MethodName, Inputs, Outputs)
{
var ExtSystem, XMLReadSvc, XMLReadSvcInputs, XMLReadSvcOutputs;
var ExtSystemParent, Child
var errorStr;

try
{
XMLReadSvcInputs = TheApplication () .NewPropertySet ();
XMLReadSvcOutputs = TheApplication () .NewPropertySet ();
XMLReadSvcInputs.SetProperty("FileName", "C:\\ebizint.xml");
XMLReadSvcInputs.SetProperty("EscapeNames", "false");
XMLReadSvc = TheApplication () .GetService ("EAI XML Read from File");
ExtSystem = inputs.GetProperty ("ExternalSystem");
XMLReadSvc.InvokeMethod("ReadXMLHier", XMLReadSvcInputs, XMLReadSvcOutputs);
```

```
//ExtSystem = {“HR”, “Finance”, “HRSensitive”}
ExtSystemParent = XMLReadSvcOutputs.GetChild(0) .GetChild(0);
for (var i =0; i < ExtSystemParent.GetChildCount(); i++)
{
Child = ExtSystemParent.GetChild(i);
if (Child.GetType() == ExtSystem)
{
for (var j=0; j < Child.GetChildCount(); j++)
{
var ExtType = Child.GetChild(j) .GetType();
var ExtValue = Child.GetChild(j) .GetValue();
Inputs.SetProperty (ExtType, ExtValue);
}
}
}
}
catch (e)
{
errorStr = e.toString();
TheApplication () .Trace(errorStr);
TheApplication () .RaiseErrorText(errorStr);
return (CancelOperation);
}
finally
{
ExtSystem = null;
XMLReadSvc = null;
XMLReadSvcInputs = null;
XMLReadSvcOutputs = null;
ExtSystemParent = null;
```

```
Child = null;
errorStr = null;
}
return (ContinueOperation);
}
```

Figure 25 displays an example usage of the EAI HTTP Transport business service, which has the above eScript code defined in the Service_PreInvokeMethod event.

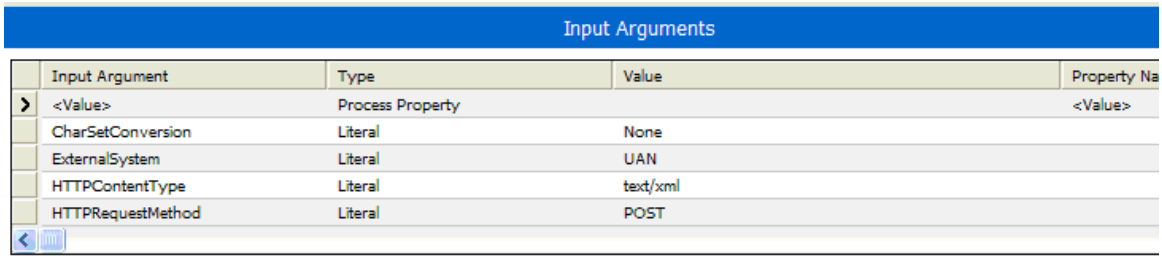


Figure 25. Example Usage of EAI HTTP Transport Business Service

Maintaining the XML File

There is no other ongoing maintenance required for ebizint.xml. Whenever the values in the file change this file needs to be updated with the correct values on the Siebel Servers. An example change that requires an update to ebizint.xml is a change in the external system URL.

14 Reference Materials for Siebel Workflow

This chapter provides reference information for Siebel workflow. It includes the following topics:

Siebel Workflow Glossary

This topic describes terms used with workflow processes. It includes the following topics:

Workflow Process Terms

Table 79 describes common terms for Workflow Processes.

Table 79. Description of Terms Used with Workflow Processes

Term	Definition	Additional Description
Arguments	Data passed to or received from a process or step.	(This cell is intentionally empty.)
Branch	A possible outcome of a workflow process step.	A branch is followed by a step in the workflow process definition. If the conditions for the branch are met, the work item proceeds to the step following the branch.
Branch Connector	A connector that emanates from a Start step, decision point, Wait step, or User Interact step.	Conditional logic is implemented on the branch connector.

Table 79. Description of Terms Used with Workflow Processes

Term	Definition	Additional Description
Business Object	A group of one or more business components.	A business object represents an entity in the Siebel application that you must monitor. A workflow process is based on one and only one business object. Business objects are defined in Siebel Tools.
Business Process	A process that is associated with operational objectives and business relationships.	A business process is a set of one or more linked procedures, which collectively realize a business objective. An example of a business process is managing a new service request.
Business Service	A type of step in a process in which an automated call is made to a service, such as the Outbound Communications service that handles inbound and outbound messaging.	A workflow process definition can have one or more business service steps.
Condition Criteria	The rules or principles that are used for evaluating the logical flow that should be taken on a branch.	(This cell is intentionally empty.)
Connector	A definition of the relationship between two workflow process steps.	(This cell is intentionally empty.)
Decision Point	A type of step in a workflow process definition in which the work item branches off to different steps depending on a set of conditions.	A Decision Point consists of possible branches for that point in the business process. Each branch consists of one or more conditions that must be met for a work item to follow that branch. A workflow process definition can have one or more Decision Points.
End Step	A type of workflow process step that specifies when a process instance is finished.	(This cell is intentionally empty.)
Error Exception	A type of workflow process step that executes in response to a deviation from normal processing.	An exception can be a system error or a user-defined error.
Expression Business Component	The name of the business component to evaluate an expression that is used as part of a search spec input argument.	(This cell is intentionally empty.)

Table 79. Description of Terms Used with Workflow Processes

Term	Definition	Additional Description
Filter Business Component	The name of the business component to provide the group of records on which a search is performed in response to a Siebel Operation step.	The Filter Business Component is part of the search spec input argument.
Input Argument	A mechanism for providing data and configuration information as input to a workflow process step.	(This cell is intentionally empty.)
Output Argument	A mechanism for providing data and configuration information as output from a workflow process step.	(This cell is intentionally empty.)
Process Property	A storage field in a workflow process that is used to contain values for use in steps as input and output arguments or for performing evaluations.	(This cell is intentionally empty.)
Process Simulator	A graphical flowchart interface used for debugging a workflow process.	(This cell is intentionally empty.)
Search Spec Input Argument	A type of input argument that allows you to restrict the records that are considered when a Siebel Operation step is executed.	(This cell is intentionally empty.)
Siebel Operation	A type of workflow process step that handles database operations such as insert, query, or update of a business component record or field.	(This cell is intentionally empty.)
Start Step	A type of step that defines the conditions for initiating an instance of a workflow process. When the conditions have been met, the process instance is initiated.	A workflow process definition has one and only one start step.
Workflow Step	An activity within a workflow process. Steps are logically linked together to create a process definition.	(This cell is intentionally empty.)
Step Branch	A workflow object definition that provides capabilities for branching logic within a workflow process.	(This cell is intentionally empty.)

Table 79. Description of Terms Used with Workflow Processes

Term	Definition	Additional Description
Step Instance	The instance of a process definition step that is initiated.	A start step is initiated when conditions defined for the start step have been met. A Decision Point is initiated when conditions for a branch connector have been met. Other steps are initiated when the previous step has finished.
Step Recipient	The intended recipient of a subprocess step.	(This cell is intentionally empty.)
Stop Step	A type of workflow process step that specifies the conditions that cause a process instance to terminate prior to completion.	(This cell is intentionally empty.)
Subprocess	A workflow process embedded into another workflow process as part of the workflow process definition.	A subprocess has its own workflow process definition. A Sub Process is a type of step. There can be one or more Sub Process steps in a workflow process.
Task	A type of step that calls a Task from within a business process.	A Task step is similar to a Sub Process step in that the task step represents a container for a further set of steps below the level of the steps in the workflow in which the task step is defined.
User Interact Step	A type of step that is used to control the flow of Siebel views within an application.	A workflow process that contains one or more User Interact steps guides a user through a specified flow of Siebel views based on the user's actions, or executes a specified set of actions.
Wait Step	A type of workflow process step that specifies when a process instance pauses in execution and the duration of the pause.	(This cell is intentionally empty.)
Watch Window	A Window in Siebel Tools that dynamically displays business component record values and process property values for a workflow process.	These values are associated with and are manipulated by the workflow process being simulated.
Workflow Mode	A property on the workflow process object definition that is used to characterize a workflow's run-time behavior as a Service Flow, Interactive Flow, Long Running Flow, or 7.0 Flow.	(This cell is intentionally empty.)

Table 79. Description of Terms Used with Workflow Processes

Term	Definition	Additional Description
Workflow Process	The representation of a business process.	A workflow process comprises one or more steps that indicate when a business process starts and ends and includes information about individual activities within the business process.
Workflow Process Instance	An instance of a workflow process that is initiated.	A process instance is initiated when the input conditions for a process definition have been met. A process instance consists of one or more step instances and contains one or more work items.
Work Item	The representation of the work being processed in the context of a step within a process instance.	A work item is an instance of a business object.

Workflow Policy Terms

Table 80 describes common terms for Workflow Policies.

Table 80. Description of Terms Used with Workflow Policies

Term	Definition	Additional Description
Business Object	A group of one or more business components.	A business object represents an entity in Siebel that you must monitor. A workflow policy object is based on one and only one business object. Business objects are defined in Siebel Tools.
Business Rule	The definition of how an organization needs to carry out a process in the organization's operations.	(This cell is intentionally empty.)
Object Type	An entity in Siebel Tools displayed as a node on the Object Explorer.	For example, workflow policy objects, workflow policy components, workflow policy columns, and policy programs are object types.
Policy Action	An event that Siebel executes when policy conditions are true and the workflow policy properties are satisfied.	Policy actions are based on programs. Policy actions are defined in the Action applet of the Workflow Policies view. Once you define a policy action, it can be used in a workflow policy.

Table 80. Description of Terms Used with Workflow Policies

Term	Definition	Additional Description
Policy Condition	A policy condition is an expression that is compared against the data in the Siebel database.	The result of the comparison is true or false. Workflow policy conditions are defined in the Workflow Policies view. A policy condition is created by defining a workflow policy column, defining a comparison operator, then entering or choosing a value, if appropriate.
Workflow Program	The definition of an event.	Types of events are Send Email, Send Page, Database Operation, Send Broadcast Message, and Run External Program. Different properties are associated with a program based on the event type. Some of the properties that can be defined for a program include the fields that can be substituted into a message, the possible recipients of a message, and the database columns that you must update. Programs are defined in Siebel Tools.
Workflow Policy	A systematic expression of a business rule.	A workflow policy contains one or more policy conditions and one or more policy actions. If the policy conditions for a workflow policy are true, then the policy action occurs. That is, when policy conditions are met. A workflow policy is contained by one workflow policy group and is related to one workflow policy object. A workflow policy contains more properties that govern the policy's behavior. Workflow policies are defined in the Workflow Policies view.
Workflow Policy Column	A column that defines the column on the Siebel database table that you must monitor.	You use workflow policy columns when defining workflow policy conditions for a workflow policy. A workflow policy column must be associated with a workflow policy component for it to be used in a workflow policy. A workflow policy column that is associated with a workflow policy component is called a workflow policy component column. Workflow policy columns are defined in Siebel Tools.
Workflow Policy Component	Components that define the Siebel database tables you must monitor. Workflow policy components also define the relationships between tables.	Workflow policy components contain workflow policy columns. Workflow policy components are defined in Siebel Tools.

Table 80. Description of Terms Used with Workflow Policies

Term	Definition	Additional Description
Workflow Policy Component Column	A workflow policy column that is associated with a workflow policy component.	Workflow policy component columns define the database columns that can be used in workflow policy conditions for a workflow policy. Workflow policy component columns are defined in Siebel Tools.
Workflow Policy Group	A group of one or more workflow policies.	Workflow policy groups allow you to group workflow policies sharing common required behavior. Siebel Server processes monitor workflow policy groups. For example, workflow policies that must be monitored hourly are different in a workflow policy group than those that are monitored weekly. Workflow policy groups are defined in the Policy Groups view.
Workflow Policy Object	A group of one or more workflow policy components.	A workflow policy object represents an entity in the Siebel application that you must monitor. A workflow policy is based on one and only one workflow policy object. Workflow policy objects are defined in Siebel Tools.

Predefined Business Services

This topic describes some of the predefined business services that can be used in a workflow process. It includes the following topics:

For more information about predefined business services, see *Business Processes and Rules: Siebel Enterprise Application Integration*.

About the Server Requests Business Service

This topic describes the predefined Server Requests business services. It includes the following topics:

You can use the predefined Server Requests business service to send generic requests to the server request broker. Processing modes in which the Server Requests business service can send requests include:

- Asynchronous.
- Synchronous.
- Schedule.

Using Synchronous and Asynchronous Modes

When in synchronous mode, the Server Requests business service sends a request to the server request broker then waits for a response. Otherwise, it sends the request but does not wait for a response.

It is recommended you use the Server Requests business service, in most cases, rather than the *Workflow Process Manager (Server Request)* business service. It is more efficient to make an asynchronous workflow process call. When a synchronous workflow call is made the workflow process runs in the Object Manager, causing the user to wait for the process to end.

Specifying the SRBroker Parameters

When invoking the Server Requests business service to submit a component request, you must specify SRBroker parameters in the input property set and component specific parameters in a child property set. Considerations to weigh include:

- When invoking the Server Requests Business Service to invoke a Siebel Component, the component parameter names must be referenced by the parameter's Alias Name. Note that the Alias Name usually does not contain spaces.
- Failure to use the correct format results in an error message. Table 81 describes correct and incorrect formats.

Table 81. Description of Component Parameter Format for Invoking the Server Requests Business Service

Correct Format	Incorrect Format
ObjReq.SetProperty "BCName", "Account"	ObjReq.SetProperty "Buscomp Name", "Account"

- Component parameters must be specified with their Alias Names in the child property set.
- There is no validation of these Alias Names.
- These arguments do not appear in the picklist in the Administration-Business Process views.

Passing Parameters to Server Components

If you need to pass parameters to the server component that are not listed as available arguments, you can create a custom business service that contains the necessary parameters. Alternatively, you can create a component job that has the parameters defined as part of the job definition. For more information on component jobs, see *Siebel System Administration Guide*.

Methods for the Server Requests Business Service

Methods available for this service include:

- **Submit Request.** Use this method to submit a request to the server request broker.
- **Cancel Request.** Use this method to cancel server requests that are currently awaiting to be run.

Submit Request Arguments

Table 82 describes Submit Request method arguments.

Table 82. Description of Submit Request Method Arguments for the Server Requests Business Service

Argument	Description
Component	Required if Component Job is not entered. Enter the name of the server component to run.
Component Job	Required if Component is not entered. Enter the name of the component job to run.
Delete After	Optional. Number of iterations before deleting the request. Works with Delete After Units. The default value is 0 (zero).
Delete After Units	Optional. The units to measure the iterations for the Delete After argument. The default value is “NoReq” for synchronous where the request is not saved to the database and “Eon” for asynchronous where the request is never deleted. Other possible values include: ■ ASAP ■ SECONDS ■ MINUTES ■ HOURS ■ DAYS ■ WEEKS ■ MONTHS ■ YEARS
Description	Optional. A description of the server request.
Enterprise Server	(This cell is intentionally empty.)
Hold Flag	Optional. For asynchronous requests only. Flag to indicate whether or not to hold the request.
Maximum Execution Time	For future use.
Method	Optional. Only applicable for service-based server components. For example, Workflow Process Manager or Communications Manager. Specify the business service method to invoke.

Table 82. Description of Submit Request Method Arguments for the Server Requests Business Service

Argument	Description
Mode of Server Request	Required. This tells the server request broker how to handle the server request. While in Auto mode, the server request broker sets the mode to Synchronous or Schedule, depending if the Siebel client is connected or mobile. Potential values include: ■ DirectDB: Asynchronous request, written directly to S_SRM_REQUEST **NOTE:** It is recommended you use this argument to invoke an asynchronous request. When DirectDB mode is used, the data in S_SRM_REQUEST is not lost. If you use Async when the Workflow Process Manager reaches MaxTasks, it cannot process every SRM request. ■ Async: Asynchronous ■ Sync: Synchronous ■ Schedule: Schedule ■ Auto: Automatic configuration
Number of Retries	Maximum number of times the request is retried in case of error..
Request ID Needed	Optional. This is only applicable to asynchronous and schedule mode. If this is set to false, these two server requests return even faster.
Repeat Amount	Amount of units of time to repeat.
Repeat Interval	Optional. The interval for repeating requests.
Repeat From	Optional. Possible values are Scheduled Start, Actual Start, and End.
Repeat Interval Units	Optional. Unit of intervals for repeating requests.
Repeat Number	Optional. The number of repetitions for repeating requests.
Retry On Error	Used for resubmitting the request, if it errors out. Default setting is FALSE.
Repeat Unit	Unit of time to repeat.
Request ID	Used for request key based routing.
Server Name	Optional. Enter the specific server that this request is run from.
Start Date	Optional. Start date and time.
Storage Amount	Optional. Enter the amount of time that the server request is stored in the database in the event that the server is down.
Storage Units	Optional. Enter the units to measure the iterations for the Storage Amount argument. The units are the same as Delete After Units.

Cancel Request Method Arguments

Table 83 describes Cancel Request method arguments.

Table 83. Description of Cancel Request Method Arguments for the Server Requests Business Service

Argument	Description
Request ID	Required. This is the ID of the server request to be cancelled.
Repeat Number	Optional. This is the number of repetitions of the repeating server requests that are to be cancelled.

Example of Using the Server Request Business Service to Invoke a Workflow From Script and Pass Values

This topic gives one example of how to use the predefined Server Requests business service to invoke a workflow process from script and to pass field values to process properties. You might use workflow processes differently, depending on your business model.

```
//Example: Passing Field Values to Process Properties and invoke workflow from script
using the Server Request BS

//function Invoke_Process()

{

var svc = TheApplication().GetService("Workflow Process Manager(Server Request)");

var Input = TheApplication().NewPropertySet();

var Output = TheApplication().NewPropertySet();

var bo = TheApplication().ActiveBusObject();

var bc = bo.GetBusComp("Opportunity");

var rowId = bc.GetFieldValue("Id");

var accountId = bc.GetFieldValue("Account Id");

Input.SetProperty("ProcessName", "My Opportunity Process");

Input.SetProperty("Object Id", rowId);

// Pass the value of the Account Id field to the Account Id process property

Input.SetProperty("Account Id", accountId);

svc.InvokeMethod("RunProcess", Input, Output);


}
```

Example of Using the Server Requests Business Service to Invoke an EIM Task

This topic gives one example of how to use the predefined Server Requests business service from within a workflow process to invoke an EIM task. You might use workflow processes differently, depending on your business model.

You can use a workflow process to start a server task. For example, you can start the EIM task to export base tables to IF tables or to load IF tables into base tables. The required EIM component-specific parameters must be passed in a child property set. You can either use a wrapper business service for EIM, such as the Synchronous Assignment Manager Requests business service, or you can use the predefined Server Requests business service.

Since the Server Requests business service is designed to invoke a variety of server tasks, it does not contain definitions for component-specific parameters. Instead, component-specific parameters are passed down in a child property set which are not declared in the repository. To make workflow pass values in a child property set, you use the *dot* notation of [type].[property]. In this example, since the Server Request Manager service does not care about the child type, but takes the first child, you pass all parameters in the same child, such as EIM.Config.

To create a workflow process that uses the Server Requests Business Service

1 In Siebel Tools, in the Workflow Processes Object List Editor (OBLE), create a new workflow process object definition with the following values:

Property	Value
Process Name	EIM Export to IF (Tools)
Business Object	Account
Workflow Mode	Service Flow

To view an example, see "Creating a New Workflow Process Object Definition" on page 316.

2 Open the Process Designer for the workflow process you created in Step 1, then create a workflow that resembles the workflow in the following diagram:

For more information, see "About Workflow Process Steps and Connectors" on page 87, and "Diagramming a Workflow Process" on page 61.

3 Click the Export EIM workflow step, then use the Properties window to define values described in the following table:

Property	Value
Business Service Name	Synchronous Server Requests
Business Service Method	SubmitRequest

4 With the Export EIM workflow step still chosen, in the MVPW add a new input argument using values described in the following table:

Argument Field	Value
Input Argument	Component
Type	Output Argument
Value	EIM

For more information, see “About Process Properties” on page 74.

5 Add another input argument to the Export EIM workflow step using values described in the following table:

Argument Field	Value
Input Argument	EIM.Config
Type	Output Argument
Value	accnt.ifb

6 Make sure the Server section of the client .cfg file contains the configuration described in the following table:

Variable	Value
GatewayAddress	Gateway_Machine_Name
EnterpriseServer	Enterprise_Server_Name
RequestComponent	SRMSynch
RequestServer	Siebel_Server_Name

7 To prepare the workflow process for testing, set up accnt.ifb in the Siebel_Server\Admin directory.

8 Validate then simulate the workflow process.

For more information, see “Process of Testing a Workflow Process” on page 231.

If the workflow executes successfully it finishes without errors. There is an EIM task log created in the Server Tasks screen in the Siebel client each time you step through the Process Simulator.

9 Deploy the workflow process.

Once the workflow is Active, you can invoke it from a workflow policy, script, or as a sub-process from another workflow process. For more information, see “Process of Deploying a Workflow Process” on page 237.

About the Workflow User Event Service Business Service

The Workflow User Event Service business service is used for one-way communication from the Siebel client to the Workflow Process Manager server component to send notification that a user event has occurred.

A User event can be generated anywhere in the Siebel enterprise, wherever a Siebel business service is used, by calling the Workflow User Event Service business service.

To trigger a long-running flow to be run in WFProcMgr, you must send a notification.

For usage information, see “About the User Event” on page 155.

This service has one method, GenerateEvent. Table 84 describes GenerateEvent method arguments.

Table 84. Description of GenerateEvent Method Arguments for the Workflow User Event Service Business Service

Argument	Description
UserEventName	The name of a user event is an agreement between the creator, an external entity, and the recipient, the workflow definition. It has no special significance, except that the incoming event name and the workflow instance definition must specify exactly the same user event name to successfully communicate with each other. A user event name must be unique. It is recommended to logically name user the event to reflect the business purpose it serves. For example, Event Transferring Send Order Confirmation from Vitira To Siebel-V2.
CorrelationValue	Used to match an incoming message with a workflow instance using business data such as an order number. Correlation is the process of matching an incoming message with a workflow instance using business data such as an order ID. When a workflow process communicates with an external entity the external entity many not be aware it is communicating with a workflow process. In such cases, it is difficult for the external entity to use a Siebel identifier, such as the workflow process instance Id, to identify the recipient. It is more convenient to use business data, such as an order number, to identify the recipient. The correlator serves this purpose. Correlation applies to user events reaching a long-running workflow, which can specify a process property as a correlator. **NOTE:** Only one process property can be used as a correlator.
<Value> (Payload)	When the user event is created, the user can specify data as payload. This data is delivered to the workflow instance that receives the event. When the workflow is defined to wait for a user event, the definition can specify a process property to receive this payload data. The payload is a single value. Only one payload can be passed. **NOTE:** If your situation calls for sending complex or structured data, it is recommended you use the XML converter to convert the data into an XML document, then pass the resulting XML string as the payload of the event. The receiving workflow can then call the XML converter again to recover the original data structure.

About the Workflow Utilities Business Service

The Workflow Utilities business service contains generic utilities that are used most typically in a test environment. To view an example that uses the Workflow Utilities Business Service, see "Example Workflow Process That Traverses a Record Set to Close Obsolete Service Requests" on page 324.

About the Echo Method

The Echo method of the Workflow Utilities business service returns a mirror image of the input arguments. Echo copies its inputs to its outputs. For example, Echo can be used in building or formatting a mail body before calling the SendMessage method of the Outbound Communications Manager.

Table 85 describes Echo method arguments.

Table 85. Description of Arguments for the Echo Method

Argument	Description
Input Arguments	This method accepts input arguments.
Output Arguments	An exact copy of the input arguments.

The Echo method was known as the Return Property Values method in Siebel version 6.x. In Siebel version 7.0.3 the display name Return Property Values was omitted from Siebel tools.

Using an Output Parameter With the Echo Method

When using the Echo Method of the Workflow Utilities business service you specify output parameters that are populated with values from the active record of the business component currently being manipulated by the workflow process.

For example, assume you must acquire the value in the Price List Id field from the active record in the Account business component. You can use the Echo method on the Workflow Utilities business service with no input argument specified and one output argument.

Table 86 describes an example output parameter specified on the Workflow Utilities business service.

Table 86. Example of Output Argument Used with Echo Method to Acquire a Value

Property	Value
Property Name	Echo Variable
Type	Business Component
Business Component Name	Account
Business Component Field	Price List Id

In this example, as a result of running the workflow process, the *Echo Variable* process property is populated with the value in the Price List Id field of the active Account business component record.

Table 87 describes methods on the Workflow Utilities business service.

Table 87. Description of Methods on the Workflow Utilities Business Service

Method Name	Description
Sleep	Sleeps for the number of seconds specified by [value].
PropSetToText	Converts a hierarchical input property set into a single string.
TextToPropSet	Converts a single string into a hierarchical output property set.
DynamicDispatch	Invokes a service.
Echo	Echo inputs to outputs.

About the Workflow Admin Service Business Service

The Workflow Admin Service business service allows a workflow process to perform administrative tasks on multiple workflow processes specified by a searchspec. Some example administrative tasks include deploy, activate, export and import. This topic includes the following topics:

- Methods of the Workflow Admin Service Business Service on page 384
- Invoking the Workflow Admin Service Through the Business Service Simulator on page 386

For usage instructions, see "Invoking the Workflow Admin Service Business Service" on page 250.

Methods of the Workflow Admin Service Business Service

Methods of the Workflow Admin Service business service include:

- **Export**. Exports workflow processes specified by a search specification into a specified directory. Each workflow process is exported to a separate .XML file with a file name being the concatenation of process name and version number.
- **Import**. Imports workflow export files specified by a search specification in a specified directory.
- **Deploy**. Deploys workflow processes specified by a search specification.
- **Activate**. Activates workflow processes specified by a search specification.
- **DeleteDeploy**. Deletes workflow deployment records specified by a search specification.

Table 88 describes arguments for the methods on the Workflow Admin Service business service.

Table 88. Description of Arguments on the Workflow Admin Service Business Service

Method Name	Argument Type	Argument Name	Description
Export	Input	ExportDir	The directory to which the export files are written to. For example, D:\workflows.
		FlowSearchSpec	The search specification used to identify the workflow processes to be exported. For example, [Process Name] like 'User Reg*'
		Repository	The repository from which workflow processes are exported. The default value is Siebel Repository.
	Output	NumFlowExported	The number of workflow processes that have been exported by the service.
Import	Input	ImportDir	The directory where the workflow export files are located.
		FileSearchSpec	The search specification used to identify the workflow export files to be imported. For example, User*.xml
		Repository	The repository into which workflow processes are imported.
		ProjectName	The Tools project into which workflow processes are imported. The project needs to be locked for the workflow import to succeed.
	Output	NumFlowImported	The number of workflow processes that have been imported by the service.
Deploy	Input	FlowSearchSpec	The search specification used to identify the workflow processes to be deployed. Note that a default search specification, [Status] = "In Progress", is automatically added to FlowSearchSpec. It is not necessary to explicitly specify the search specification.
	Output	NumFlowDeployed	The number of workflow processes that have been deployed by the service.

Table 88. Description of Arguments on the Workflow Admin Service Business Service

Method Name	Argument Type	Argument Name	Description
Activate	Input	FlowSearchSpec	The search specification used to identify the workflow processes to be activated. Note that a default search specification, [Status] = "COMPLETED", is automatically added to FlowSearchSpec. It is not necessary to explicitly specify the search specification.
	Output	NumFlowActivated	The number of workflow processes that have been activated by the service.
DeleteDeploy	Input	FlowSearchSpec	The search specification used to identify the workflow deployment records to be deleted.
	Output	NumFlowDeleted	The number of workflow deployment records that have been deleted by the service.

Invoking the Workflow Admin Service Through the Business Service Simulator

It is recommended to invoke the Workflow Admin Service business service through the Business Service Simulator when you need to activate or deploy a large number of workflows in bulk. This way you can supply the search specification for each set of workflows. It is not recommended to run the Workflow Admin Service business service from within a workflow process because you must change the input search specification inside the workflow process for each set of workflows involved, resulting in the need to revise the workflow process each time it calls the Workflow Admin Service business service.

About the Outbound Communications Manager Business Service

The Outbound Communications Manager business service can be used to send notifications through fax and email, such as notifications to contacts or employees. For information on methods and arguments, see *Siebel Communications Server Administration Guide*.

Specifying a Substitution With the Outbound Communications Manager Business Service

You can specify a substitution variable when calling the Outbound Communications Manager business service from a workflow process.

To create a test workflow process that uses a substitution variable when calling the Outbound Communications Manager

1 In Siebel Tools, create a new workflow process object definition using values described in the following table:

Property	Value
Process Name	Send SR Notification
Workflow Mode	Service Flow
Business Object	Service Request

To view an example, see “Creating a New Workflow Process Object Definition” on page 316.

2 Add steps and connectors until your workflow process resembles the workflow illustrated in the following diagram:

For more information, see “About Workflow Process Steps and Connectors” on page 87, and “Diagramming a Workflow Process” on page 61.

3 Click the Send Notification business service step, then use the Properties window to define values described in the following table:

Property	Value
Business Service Name	Outbound Communications Manager
Business Service Method	SendMessage

4 With the Send Notification step still chosen, click the MVPW, then add a new input argument using values described in the following table:

Input Argument	Type	Value	Business Component Name
MsgBody	Expression	"The Service Request Number [SR Number] has been created with the abstract [SR Abstract]"	Service Request

For more information, see “About Process Properties” on page 74.

5 Add another input argument to the Send Notification step using values described in the following table:

Input Argument	Type	Value
CommProfile	Literal	(Enter the appropriate profile. The profile must correspond with a profile that appears in the Administration - Communications > All Configurations > Profiles view in the Siebel client. Type in the profile name exactly as it appears in the client.)

6 Validate then simulate the workflow process.

For more information, see "Process of Testing a Workflow Process" on page 231.

7 Implement this technique in your production workflow.

Sending an Email to the Owner of a Product Defect

You can use the Outbound Communications Manager to send an email to the owner of a product defect. This technique involves setting up a recipient group.

To send an email to the owner of a Product Defect using a workflow process, you start by creating the custom recipient group for a product defect owner.

To create the custom recipient group for a Product Defect Owner

1 In the Siebel client, navigate to Administration - Data > List of Values, then create a new record using values described in the following table:

Property	Value
Type	COMM_RECIP_SRC
Display Value	Product Defect
Language - Independent Code	Product Defect
Parent LIC	(Leave this field empty.)

2 Create a second record using values described in the following table:

Property	Value
Type	COMM_RECIP_SRC
Display Value	Product Defect Owner
Language - Independent Code	Comm Employee
Parent LIC	Product Defect

3 In Siebel Tools, from the application level menu choose View > Options, click the Object Explorer tab, expand the Applet tree, make sure the Applet Toggle object type has a checkmark, then click OK.

4 Navigate to the Applets OBLE, then query the Name column for Comm Source List Applet.

5 From the application-level menu, choose Edit > Copy Record, then specify properties using values described in the following table:

Property	Value
Name	Comm Source Product Defect List Applet
Class	CSSFrameListCommSrc

6 In the Applets OBLE, query the Name column for Comm Source List Applet, right-click the record, then choose Lock Object.

7 In the Object Explorer, expand the Applet object type, choose the Applet Toggle object type, then add a new record in the Applet Toggles OBLE using values described in the following table:

Property	Value
Applet	Comm Source Product Defect List Applet

8 In the Object Explorer, choose the Link object type, then add a new record in the Links OBLE using values described in the following table:

Property	Value
Name	Comm Request/Product Defect
Parent Business Component	Comm Request
Child Business Component	Product Defect
Inter Table	S_COMM_REQ_SRC
Inter Parent Column	COMM_REQ_ID
Inter Child Column	SRC_ROW_ID

Make sure you define properties in the order as they appear listed in the table.

9 Navigate to the Business Objects OBLE, query the Name column for Comm Request, right-click the record then choose Lock Object.

10 In the Object Explorer, expand the Business Object object type, then add a new record in the Business Object Components OBLE using values described in the following table:

Property	Value
BusComp	Product Defect
Link	Comm Request/Product Defect

The work you have performed thus far is required to create a custom recipient group for a product defect owner. For more information, see *Siebel Communications Server Administration Guide*.

Next, you create the product defect/employee link.

To create the Product Defect/Comm Employee link

1 In the Object Explorer, choose the Link object type, then add a new record in the Links OBLE using values described in the following table:

Property	Value
Name	Product Defect/Comm Employee
Parent Business Component	Product Defect
Child Business Component	Comm Employee
Source Field	Owned By Id
Destination Field	Id

2 Navigate to the Business Objects OBLE, query the Name column for Product Defect, right-click the record then choose Lock Object.

3 In the Object Explorer, expand the Business Object object type, then add a new record in the Business Object Components OBLE using values described in the following table:

Property	Value
BusComp	Comm Employee
Link	Product Defect/Comm Employee

4 Compile your changes.

Next, you create a template that defines the email message that is sent.

To create a template that defines the email message

1 In the Siebel client, navigate to Communications > My Templates, then add a new record using values described in the following table:

Property	Value
Name	eMail Notification - Product Defect
Channel Type	Email

2 Click the Advanced view tab, then set the Recipient Group property to *Product Defect Owner*.

3 Specify other information in the Advanced form, as required.

For a detailed explanation of the available fields, see *Siebel Communications Server Administration Guide*.

Next, you can create a test workflow process that calls the template.

To create a test workflow process that calls the template

1 In Siebel Tools, in the Workflow Processes OBLE, create a new workflow process object definition with the following values:

Property	Value
Process Name	Email Notification of Product Defect
Business Object	Service Request
Workflow Mode	Service Flow

To view an example, see "Creating a New Workflow Process Object Definition" on page 316.

2 Open the Process Designer for the workflow process you created in Step 1, then create a workflow that resembles the workflow in the following diagram:

For more information, see "About Workflow Process Steps and Connectors" on page 87, and "Diagramming a Workflow Process" on page 61.

3 Click the Call Template step then use the Properties window to define values described in the following table:

Property	Value
Business Service Name	Outbound Communications Manager
Business Service Method	CreateRequest

4 With the Call Template step still chosen, in the MVPW define three new input arguments using values from the following table:

Input Argument	Type	Value
PackageNameList	Literal	eMail Notification - Product Defect
RecipientGroup	Literal	Product Defect Owner
RequestName	Literal	Comm Request created by AP Send Updated CR Email

For more information, see "About Process Properties" on page 74.

5 With the Call Template step still chosen, in the MVPW define one more input argument using values from the following table:

Input Argument	Type	Value
SourceIdList	Process Property	Object Id

6 Validate then simulate the workflow process.

For more information, see "Process of Testing a Workflow Process" on page 231.

7 Implement this technique in your production workflow.

Other Business Services used with a Workflow Process

This topic mentions other business services that are sometimes with a workflow process.

About the Synchronous Assignment Manager Requests Business Service

The Synchronous Assignment Manager Requests business service is for assigning an object using Assignment Manager rules. For more information, see *Siebel Assignment Manager Administration Guide*.

This service has one method available, Assign. This method sends a request to the assignment manager server component.

Assign Arguments

Table 89 describes Assign method arguments.

Table 89. Description of Arguments for the Assign Method

Argument	Description
Assignment Object Name	Required. This is the object that you need to assign.
Object Row ID	Required. This is the row ID for the object you need to assign. To assign the work item for the workflow process, set this to the Object Id process property.
Reply	It is an output argument of this method.

Avoiding Errors Due to Locked Records

The Synchronous Assignment Manager Requests business service attempts to assign records that meet the appropriate criteria, even if they are locked. To prevent errors in your process due to locked records, set up a condition in your workflow process or workflow policy to skip records that do not meet the condition ASGN_USR_EXCLD_FLG = N.

About the Report Business Service

The Report business service automates sending, scheduling, printing, saving, and emailing reports. It also automates administrative jobs such as synching new users.

For more information, see *Siebel Reports Administration Guide*.

About the FINS Data Transfer Utilities Business Service

The FINS Data Transfer Utilities business service allows you to transfer data from a source business component to a destination business component without using script.

For more information, see *Siebel Finance Guide*.

About the FINS Validator Business Service

The FINS Validator business service allows you to validate data based on predefined rules. It is developed through Application Administration, not through script. The Validator business service supports custom messages. For more information, including invoking the FINS Validator Service from a workflow, see *Siebel Finance Guide*.

About the Dynamic UI Business Service

This business service and associated administration views allow you to create and render views containing a single read-only applet in the Siebel Financial Services application. For more information, see *Siebel Finance Guide*.

Predefined Workflow Policy Programs

Workflow Policies comes with a number of predefined workflow policy programs that you can use to meet specific business requirements. This way, you can modify existing, predefined objects rather than create them. This topic provides reference information for predefined workflow programs. It includes the following topics:

About Predefined Workflow Policy Program Types

This topic describes the predefined workflow policy program types that can be used in a workflow process. It includes the following topics:

Overview of Predefined Workflow Policy Program Types

A workflow policy program is a generic event that actions are based on. A workflow policy program defines the particular action that takes place when the conditions of a workflow policy are met. To view object definitions for predefined workflow policy programs, navigate to the Workflow Policy Programs OBLE in Siebel Tools.

Table 90 describes workflow policy programs that have been created from program types.

Table 90. Description of Predefined Workflow Policy Program Types

Program Type	Program	Description
Send Page	Send Page	Send a generic page message.
	Send Opportunity Page	Send a page regarding an opportunity.
	Send Quote Page	Send a page regarding a quote.
	Send SR Page	Send a page regarding a service request.

Table 90. Description of Predefined Workflow Policy Program Types

Program Type	Program	Description
Send Email	Send Email	Send a generic email message.
	Send Opportunity Email	Send an email regarding an opportunity.
	Send Quote Email	Send an email regarding an opportunity quote.
	Send SR Email	Send an email regarding a service request.
Send Broadcast Message	Send Message Broadcast	Send a generic broadcast message.
	Send SR Message Broadcast	Send a broadcast message regarding a service request.
	Send Opportunity Message Broadcast	Send a broadcast message regarding an opportunity.
Database Operation	Change SR Close Date to Today	Update the service request's close date to today's date.
	Change SR Owner	Change the service request's owner.
	Change SR Group	Change the service request's group.
	Change SR Owner to Manager	Change the service request's owner to the current owner's manager.
	Change SR Priority	Change the service request's priority to a new value.
	Change SR Severity	Change the service request's severity to a new value.
	Change SR Status	Change the service request's status to a new value.
	Change SR Sub-status	Change the service request's sub-status to a new value.
	Create SR Activity	Create a service request activity.
	Create Opportunity Activity	Create an opportunity activity.

The table above provides an inventory of predefined workflow policy programs, and describes common actions you can use by inserting your own message text.

The Send Page Program Type

The Send Page Arguments applet displays if you choose the Send Page workflow policy program type in the Workflow Policies Actions applet.

Table 91 describes arguments and valid values for the Send Page workflow policy program type.

Table 91. Description of Arguments and Values for the Send Page Workflow Policy Program Type

Argument	Valid Values When Used by Action
Numeric Message Template	Numeric message when pager is numeric.
Alpha Message Template	Text message when pager is alphanumeric. "Current" is a reserved word in Siebel Workflow. Do not use this word in messages.
Available Substitutions	Dynamic fields that you can use in the Alpha Message Template. When the action executes, the substitution value is populated with the value from the record that meets the workflow policy conditions.

Considerations for setting the Send Page arguments include:

- Workflow policies automatically determine the correctly formatted message depending on what type of pager the person being paged has.
- If neither of the message arguments has a value, Workflow Policies logs an error message and the action is not finished.
- You can send only pages to employees. The pager information for an employee is stored in the Employee Administration view. The Siebel database currently does not store pager information for contacts.
- Messages support substitution of values that come from the Available Substitutions field.

Configurations That Specify Numeric Paging

Numeric paging is inherently unreliable because of a lack of a computer protocol for sending numeric pages. If you must send a numeric page, you can use the Pager Pin field in the employee table to control the delay between dialing the paging phone number and sending the numeric message. Add commas to the Pager Pin field. Each comma is roughly equal to a half second delay. Avoid using the numeric paging feature in mission critical applications.

The Send Email Message Program Type

When you choose the Send Email workflow policy program in the Actions applet, the Send Message Arguments applet displays along with the Recipients applet.

The Send Message Arguments applet allows you to create an email template used to build the message sent to the recipient specified in the Recipients applet.

Table 92 describes arguments and valid values for the Send Email workflow policy program type.

Table 92. Description of Arguments and Values for the Send Email Workflow Policy Program Type

Argument	Valid Values When Used by Action
Subject	Subject line of email message.
Message Template	Text of message. Maximum length is 2000 characters, including variable substitutions. *Current* is a reserved word in Siebel Workflow. Do not use this word in a message.
Repeating Message	Message that is repeated when the Consolidate flag is checked on the Workflow Policies view. *Current* is a reserved word in Siebel Workflow. Do not use this word in messages.
Available Substitutions	Dynamic fields that you can use in Subject, Message Template, and Repeating Message. When the action executes, the substitution value is populated with the value from the record that meets the policy conditions.

Sending Email to an Email Address Held in Any Column

Workflow Manager can be configured to send an email message to an email address that is defined in a custom field, including to an address stored in a column other than S_EMPLOYEE.EMAIL_ADDR. For more information, see 475546.1 (Doc ID) on Oracle*MetaLink* 3.

The Send Broadcast Message Program Type

The Send Message Broadcast Arguments applet appears if the Send Message Broadcast workflow policy program is chosen in the Actions applet.

Table 93 describes arguments and valid values for the Send Broadcast Message workflow policy program type.

Table 93. Description of Arguments and Values for the Send Broadcast Message Workflow Policy Program Type

Argument	Valid Values When Used by Action
Activation	Date and time for which the broadcast message is active. The variable CURRENT can be used when specifying the activation date. For more information, see "Entering Date Calculations in the Conditions Applet" on page 211.
Expiration	Date and time when the broadcast message expires. The variable CURRENT can be used when specifying the activation date. For more information, see "Entering Date Calculations in the Conditions Applet" on page 211.
Abstract	Short description of the broadcast message.
Message Template	Text of message to broadcast. Maximum length is 2000 characters, including variable substitutions. *Current* is a reserved word in Siebel Workflow. Do not use this word in a message.
Severity	Severity of message to broadcast.
Available Substitutions	Dynamic fields that you can use in the Abstract and Message Template. When the action executes, the substitution value is populated with the value from the record that meets of the policy conditions.

Activating the Check New Broadcasted Message Workflow Policy

If you use a workflow policy that contains a workflow policy program Type property that is set to Send Broadcast Message, and you implement broadcast message caching on an object manager component, then you must activate the Check New Broadcasted Message workflow policy, which belongs to the Siebel Messaging policy group.

The Check New Broadcasted Message policy monitors the S_BRDCST_MSG table and invokes the Notify Broadcasted Message workflow process to broadcast a new message added to the table.

For information about activating a workflow policy, see "Creating Database Triggers" on page 276. For information on configuring broadcast message caching, see *Siebel Applications Administration Guide*.

Consolidating Messages Sent Through Send Broadcast Message

It is not possible to consolidate multiple broadcast messages into a single message that is then displayed to a user. A user receives all broadcast messages sent by siebel workflow, each as a separate message.

The Database Operation Program Type

Siebel Business Process Designer has a number of database operation programs already predefined. You just define the parameters. The Arguments applet appears if you choose a database operation program such as Create Opportunity Activity in the Actions applet.

Table 94 describes arguments and valid values for the Database Operation workflow policy program type.

Table 94. Description of Arguments and Values for the Database Operation Workflow Policy Program Type

Argument	Valid Values When Used by Action
Name	Name of column to be updated.
Required	Indicates the argument is required.
Value	Updated value of the column. You can use substitutions in the value if they were defined in the program. The syntax for adding substitutions to the value is square brackets around the variable. For example, [SR Num].

Examples of Predefined Workflow Policy Programs

This topic includes examples of using predefined workflow policy programs. It includes the following topics:

The examples use predefined workflow policy programs included with Workflow Policies. These programs can be viewed in Siebel Tools' Object Explorer by choosing the Workflow Policy Program object type.

This topic also includes "Predefined Workflow Policy Programs for Siebel Marketing" on page 403.

Example of a Predefined Workflow Policy Program That Manages SR Close Date

This topic gives one example of using a predefined workflow policy program to automatically close SRs that have been marked as resolved but not closed. You might use this feature differently, depending on your business model.

Using this program, you can define a policy such that if a Service Request has an activity of type Resolution, and the SR is open for more than five days, the SR close date is changed to today's date.

When the policy triggers the workflow policy program, the program enters the current system date into the Close Date field of the Service Request record.

Table 95 describes arguments for the Change SR Close Date to Today program.

Table 95. Description of Program Arguments to Change SR Close Date to Today

Argument Name	Comment
Primary ID	Contains the row ID of the Service Request record meeting the policy condition.
Primary Table Operation Type	Specifies the table (S_SRV_REQ) and what action is to take place (Update).
Sql Statement	`select sys_extract_utc(current_timestamp) from &Table_Owner.S_DUAL` This statement calls the Siebel function now() to obtain the current date and uses the table &Table_Owner.S_DUAL to hold the value temporarily. The S_DUAL table is used to hold temporary values. Math functions can also be performed. For example, the SQL statement `select {fn now()}+7 from &Table_Owner.S_DUAL` returns the current date plus seven days. Different RDBMS have different formats for the same function. For example, in MS SQL, the function GetDate() is used to return the current date. **NOTE:** The column name length must be less than 30 characters. For example, the following statement violates this rule because it is more than 30 characters: `TO_CHAR(CREATED,'dd month yyyy','NLS_DATE_LANGUAGE=FRENCH')`.
Sql Statement Outputs	The *Today* variable obtains its value from the SQL statement.
New Close Date (Column)	Specifies the column in the record to be updated (ACT_CLOSE_DT).
New Close Date	Specifies the field to be updated to the value of Today.
Update Row ID	The row ID of the record you need to update. This is the same as the value of the Primary ID.

Example of a Predefined Workflow Policy Program That Assigns an SR Owner

This topic gives one example of using a predefined workflow policy program to automatically assign an SR that has not yet been assigned. You might use this feature differently, depending on your business model.

If an open service request is not assigned for a certain length of time, this workflow policy program can be used to assign, that is change the owner, of a service request to the expert in the area of the specific service request. This allows the correct people to see the incoming service request and assign it appropriately.

This workflow policy program allows you to choose a new owner from a picklist and put it into the field of the Service Request record matching the policy condition.

Table 96 describes arguments for the Change SR Owner program.

Table 96. Description of Program Arguments to Change SR Owner

Argument Name	Comment
Primary ID	Contains the row ID of the Service Request record meeting the policy condition.
Primary Table Operation Type	Specifies the S_SRV_REQ table and to take the Update action.
New Owner (Column)	Specifies to update the Owner_EMP_ID field in the record to be updated.
New Owner	Indicates that a picklist is displayed for assigning a new owner. The picklist is defined by columns picklist set to Picklist SR Owner, Source set to ID, and Applet set to SR Owner Pick Applet.
Visible	When checked, indicates the picklist is visible to the user.

Example of a Predefined Workflow Policy Program That Escalates an SR

This topic gives one example of using a predefined workflow policy program to assign an open SR to a manager. You might use this feature differently, depending on your business model.

If the service request is not closed within a specific duration of time, assign the service request to the owner's manager. This allows a proper response time to service calls.

Tasks performed by this workflow policy program include:

- Uses the Primary ID as input into a SQL statement.
- Uses a query SQL statement to retrieve the current value of the field Manager.
- Sets the New Owner field to default to the current value of Manager.
- Allows the user to update the New Owner field optionally through a picklist.

Table 97 describes arguments for the Change SR Owner to Manager program.

Table 97. Description of Program Arguments to Change SR Owner to Manager

Argument Name	Comment
Primary ID	Contains the row ID of the Service Request record meeting the policy condition.
Primary Table Operation Type	Specifies the S_SRV_REQ table and to take the Update action.
New Owner (Column)	Specifies to update the Owner_EM_ID field in the record to be updated.
New Owner	Indicates that a picklist is displayed for assigning a new owner.

Sql Statement	Policy Monitor requires definitions that are contained in the workflow policy object, workflow policy components, and workflow policy columns. In working and coding a workflow policy program using Siebel tables, explicitly joining the base table through SQL code is required.
	
	This example SQL statement joins four tables, giving access to data from these four tables:
	
	`SELECT MGRPOS.PR_EMP_ID FROM`
	`&TABLE_OWNER.S_POSTN POS, &TABLE_OWNER.S_EMPLOYEE EMP, &TABLE_OWNER.S_POSTN`
	`MGRPOS, &TABLE_OWNER.S_SRV_REQ SR`
	`WHERE`
	`SR.ROW_ID = ?`
	`AND`
	`SR.OWNER_EMP_ID = EMP.ROW_ID`
	`AND`
	`EMP.PR_POSTN_ID = POS.ROW_ID`
	`AND`
	`POS.PAR_POSTN_ID = MGRPOS.ROW_ID`
	
	Considerations to weigh for this example include:
	
	■ Only one field is retrieved.
	
	■ `SR.ROW_ID = ?` uses a question mark as a placeholder for inputting the value of the Primary ID. The Primary ID is substituted for the question mark.

Sql Statement Inputs	Set to the value of Primary ID.
Sql Statement Outputs	Set to the value of Manager.

Example of a Predefined Workflow Policy Program That Sends a Quote by Using a Page

This topic gives one example of using a predefined workflow policy program to automatically send a quote depending on the relationship between the quote's value and the revenue value of opportunity to which the quote references. You might use this feature differently, depending on your business model.

If a created quote has a value less than some percentage of the opportunity's revenue, it is considered heavily discounted, then send a page to a designated employee.

This workflow policy program sends out a pager message. The SQL statement is configured for the different RDBMS syntax.

There are four SQL statements, one default and three specific to an RDBMS: Informix, Oracle, and SQL Anywhere.

The SQL Statement *default* query retrieves five values from four tables using an outer join specified by *=:

```
select
q.QUOTE_NUM, q.REV_NUM, o.NAME, a.NAME, a.LOC
from
&Table_Owner.S_DOC_QUOTE q, &Table_Owner.S_ORG_EXT a, &Table_Owner.S_OPTY o
where
q.ROW_ID = ? and q.OPTY_ID *= o.ROW_ID and q.TARGET_OU_ID *= a.ROW_ID
```

The SQL statement *Oracle* query retrieves five values from four tables using an outer join specified by (+):

```
select
q.QUOTE_NUM, q.REV_NUM, o.NAME, a.NAME, a.LOC
from
&Table_Owner.S_DOC_QUOTE q, &Table_Owner.S_ORG_EXT a, &Table_Owner.S_OPTY o
where
q.ROW_ID = ? and q.OPTY_ID = o.ROW_ID (+) and q.TARGET_OU_ID = a.ROW_ID (+)
```

The SQL Statement is required. However, if an SQL Statement with <SQL style> is present, this takes precedent over SQL Statement.

The SQL statement outputs define five variables to hold the result of the query statement. These variables are Quote Number, Revision, Opportunity, Account, and Site.

In an outer join, there might not be an associated table, in which case the variable is set to null.

Predefined Workflow Policy Programs for Siebel Marketing

This topic describes how to use workflow policy programs to assist with executing a marketing campaign.

Overview of Marketing Workflow Policy Programs

The Workflow Policy programs in Siebel Marketing were designed to allow a marketer to create complex campaign policies to automate the different stages of the campaign. Actions are based on the type of workflow policy program used by Workflow Policies to create campaign policies.

Workflow policy programs that create actions to execute campaigns include:

- **Send Campaign Email.** Sends email to contacts and prospects associated with a campaign.
- **Create Campaign Contact Activity.** Creates an activity record for the contacts or prospects that are the target of the marketing campaign.
- **Assign to Campaign.** Takes a contact or a prospect and assigns it to a chosen campaign.

Using the Send Campaign Email Workflow Policy Program

The Send Campaign Email program provides marketers with the ability to send emails to the targets of the marketing campaign, such as contacts or prospects.

Send Campaign Email uses Available Substitutions in the Send Message Arguments applet, such as Prospect First Name, to allow for personalization of campaign emails.

You use the Recipients applet in the Workflow Policy view in the Siebel client to choose the Recipient Type. The campaign contacts and prospects to whom the email is sent is visible in the Contacts/ Prospects applet in the Campaign Administration view.

Modifying Available Substitutions

To add a new substitution to the Available Substitutions list, you modify the SQL Statement Outputs property for the workflow policy program.

To add available substitutions for the Send Campaign Email Workflow Policy Program

1. In Siebel Tools, choose Send Campaign Email in the Workflow Policy Programs OBLE.
2. Right-click, then choose Lock Object.
3. In the Workflow Policy Program Arguments OBLE, choose the SQL Statement Outputs property.
4. In the Default Value property, add the substitution.

The Default Value property contains a comma-separated list of substitution variables. These variables are used for holding the result of the query statement. The list specified in the Default Value property appears in the Available Substitution field in the Send Message Argument applet.

Using the Create Campaign Contact Activity Workflow Policy Program

The Create Campaign Contact Activity workflow policy program creates an activity record for the contacts or prospects that are the targets of the campaign. In the Arguments applet, you specify the data that populates the Contact Activity record.

Table 98 describes valid values for the Arguments applet.

Table 98. Description of Program Arguments to Create Email Activity

Argument	Value
Name	**Description.** Use text to describe activity. **Status.** Choose the activity status such ,as planned or active, from the picklist. **Type.** Choose the Activity type from the picklist.
Required	This value indicates whether the argument is required.
Value	Text or picklist.

Using the Assign to Campaign Workflow Policy Program

This workflow policy program adds the contact or prospect to the list of campaign contacts or prospects for the designated campaign.

Table 99 describes the value for the New Campaign argument.

Table 99. Description of Program Argument to Assign to Campaign

Argument	Value
New Campaign	Picklist that allows you to choose a campaign to which you assign the contact or prospect.

Example of Workflow Policies That Manage a Marketing Campaign

This topic gives one example of using workflow policies to manage a marketing campaign. You might use this feature differently, depending on your business model.

In this example, a marketer needs to run a two-tier campaign with different actions taken depending on how the campaign recipient responds. The marketer is calling the campaign the *CD-ROM Promotion*. The marketer needs the campaign to work in the following ways:

- An email is sent telling recipients they can receive a discount by ordering a new product over the phone. The marketer needs to keep track of the recipients and to give them two weeks to respond.
- At the end of the two week period, recipients who did not respond to the offer are assigned to a new campaign.

Activities the marketer must perform to set up this campaign include:

- Define the actions to be used by the policies. For more information, see "Defining Workflow Policy Actions for the Marketing Campaign" on page 406.

- Create a workflow policy group for the campaign. For more information, see "Creating the Workflow Policy Group for the Marketing Campaign" on page 408.
- Create policies for the two tiers of the campaign. For more information, see "Creating Policies for the Marketing Campaign" on page 408.

Defining Workflow Policy Actions for the Marketing Campaign

Workflow policy actions required for this example include:

- **Send Campaign Email.** To send the offer email to the campaign recipients.
- **Create Campaign Contact Activity.** To record activity associated with the contact.
- **Assign to Campaign.** To assign non-respondents to a new campaign.

First, define a send campaign email action.

To define a Send Campaign Email Action

1. In the Siebel client, navigate to Administration-Business Process > Actions.
2. In the Actions applet, create a new record using values described in the following table:

Field	Value
Name	Send First Campaign Contact
Program	Send Campaign Email
Workflow Object	Campaign Contact
Comment	(Enter appropriate text, as necessary.)

3. In the Send Message Arguments applet, specify the argument using values described in the following table:

Field	Value
Subject	(Enter text and dynamic fields, as necessary.)
Message Template	(Enter text and dynamic fields for sending email to Contacts.)

4. In the Recipients applet, create a new record using values described in the following table:

Field	Value
Type	(Choose a predefined Recipient.)
Name	(Choose the appropriate recipient name.)

This action is now available for use in a workflow policy.

Next, define a Create Campaign Contact Activity action.

To define a Create Campaign Contact Activity Action

1 Navigate to Administration-Business Process > Actions.

2 In the Actions applet, create a new record using values described in the following table:

Field	Value
Name	First CD-ROM Campaign
Program	Create Campaign Contact Activity
Workflow Object	Campaign Contact
Comment	(Enter appropriate text, as necessary.)

3 In the Arguments applet, specify the Type argument using values described in the following table:

Field	Value
Argument	Type
Value	(Define the type of contact activity for this action, such as In Store Visit, or Demonstration.)

The Type argument is required. You can also specify additional optional arguments, such as Description or Status. For each additional argument you specify, create a new record in the Arguments applet, then define the field and value.

Next, define an Assign to Campaign Email action.

To define an Assign To Campaign email Action

1 Navigate to Administration-Business Process > Actions.

2 In the Actions applet, create a new record using values described in the following table:

Field	Value
Name	Assign to Campaign
Program	Assign to Campaign
Workflow Object	Campaign Contact
Comment	(Enter appropriate text, as necessary.)

3 In the Arguments applet, specify the Type argument using values described in the following table:

Field	Value
Argument	New Campaign
Value	(This cell is intentionally empty.)

Creating the Workflow Policy Group for the Marketing Campaign

Policies must be assigned to a workflow policy group. Therefore, in this example a workflow policy group is created specifically for the marketing campaign.

To define a workflow Policy Group for the marketing campaign

1 Navigate to Administration-Business Process > Policy Groups.

2 In the Policy Groups applet, add a new record using values described in the following table:

Field	Value
Name	CD-ROM Promotion
Comments	group of policies for CD-ROM marketing campaign

Creating Policies for the Marketing Campaign

Once the workflow policy actions and the workflow policy group are ready, the policies can be created. Policies required in this example include:

- **Email for CD-ROM campaign.** Triggers the sending of the offer email and the email activity record.
- **Assign Non-Respondents.** Triggers the reassignment of non-respondents to a new campaign.

When performing the procedures for creating the policies, below, pay careful attention to how the fields in the Conditions applet are set.

To create the email for the marketing campaign policy

1 Navigate to Administration-Business Process > Policies.

2 In the Policies List applet, create a new record using values described in the following table:

Field	Value
Name	Email for CD-ROM campaign
Workflow Object	Campaign Contact
Policy Group	CD-ROM Promotion
Duration	0

Note that the Policy Group you specify here is the group you crated in the procedure in "Creating the Workflow Policy Group for the Marketing Campaign" on page 408.

3 In the Conditions applet, to specify the name, create a new record using values described in the following table:

Field	Value
Condition Field	Name
Operation	=
Value	1st CD-ROM Promotion

4 In the Conditions applet, to specify the Start Date, create a new record using values described in the following table:

Field	Value
Condition Field	Start Date
Operation	=
Value	(Enter the date on which the campaign starts sending messages to the target audience.)

5 In the Conditions applet, to specify the Campaign Status, create a new record using values described in the following table:

Field	Value
Condition Field	Campaign Status
Operation	=
Value	Launched

The Launched value is the trigger that sets off the campaign.

Next, create the Assign Non-Respondents policy.

To create the assign non-respondents policy

1 Navigate to Administration-Business Process > Policies.

2 In the Policies List applet, create a new record using values described in the following table:

Field	Value
Name	Non-Respondents of CD-ROM campaign
Workflow Object	Campaign Contact
Policy Group	CD-ROM Promotion
Duration	14

3 In the Conditions applet, to specify the Name, create a new record using values described in the following table:

Field	Value
Condition Field	Name
Operation	=
Value	1st CD-ROM Promotion

4 In the Conditions applet, to specify the Campaign Status, create a new record using values described in the following table:

Field	Value
Condition Field	Campaign Status
Operation	=
Value	Launched

5 In the Conditions applet, to specify the Done Flag, create a new record using values described in the following table:

Field	Value
Condition Field	Done Flag
Operation	=
Value	N

Defining Logic That Triggers the Assign Non-Respondents Policy

Setting Done Flag to N flags the activity record for this recipient as requiring additional attention.

If the Policy duration is set to 14 days and Done Flag is equal to N, then the policy executes in 14 days. Members of the target audience who did not respond to the first campaign are assigned to a new campaign after 14 days.

Predefined Workflow Policies

You can used predefined workflow policies to implement commonly used functionality. These policies can be viewed as groups in the Siebel client by navigating to Administration - Business Process > Policy Groups.

Predefined Messaging Workflow Policies

You can use a predefined messaging policy to display a message in pop-up dialog box in the Siebel client. For example, the predefined Messaging Policy Send Screen Pop for Activity can be used to display a pop up dialog that informs the end-user of a new activity.

To view groups of predefined messaging policies, in the Siebel client navigate to Administration - Business Process > Policy Groups. Choose Siebel Messaging in the Name column. The messaging policies are displayed in the Policies applet.

to configure pop up messaging

1. In the Siebel client, navigate to Administration - Business Process > Policy Group, then choose Siebel Messaging in the Name column.
2. Locate the workflow policy you need to use, such as Messaging Policy Send Screen Pop for Activity, then set the Expiration field to NULL.
3. Run generate triggers with EXEC = True and Remove = True.

   For more information, see “Running Generate Triggers” on page 278.
4. Run generate triggers again, this time with Remove = False.
5. Navigate to Messages > All Messages, then click the New button.
6. In the Last Name field, enter the name of the contact who receives the message.
7. In the Assigned To field, enter the name of the employee this user is assigned to.
8. In the Message field, type the text message that appears in the pop up dialog.
9. Set the Alert Type field to Screen Alert.
10. Click the Message Alert Setup tab, then make sure there is a corresponding entry for the message recipient and that Alert Type is set to Screen Alert.

    If the message recipient has not already been defined, click the New button then set the Last Name field to the same name you set in Step 6, and set the Alert Type field to Screen Alert.

11 Start a Workflow Monitor Agent task for the workflow policy group you activated in Step 3.

For more information, see "Executing a Workflow Policy with Workflow Monitor Agent" on page 290.

12 Verify the configuration (optional).

a Specify a test user when assigning the contact in Step 6.

b In a second client session, log into the Siebel client as the test user.

c In the initial client session, insert a new activity for this user.

d In the second client session, verify the pop up dialog appears that contains the message you entered in Step 8.

Identifying the Source Table When Modifying an Existing Workflow Policy

When defining the types of workflow policies you must operate for your business, you might find that the predefined workflow policy objects do not contain the policy components you need. Use the procedures in this topic as a general guideline for identifying the source database table and column when modifying an existing workflow policy object.

Tasks you should perform before modifying a workflow policy object include:

- Identify the names of the workflow policy object's database table and column. If you are going to add or modify a component, you must know the relationship between the component and the primary workflow policy component.
- Make sure you do not have other records referencing this object that can be affected by your change. For example, before inactivating a component column, confirm that no policy conditions are referencing the component column.

The procedures in this topic are presented in the context of Account objects.

To identify the database table

1 In the Siebel client, navigate to the appropriate workflow policy object view. This is the view that contains the business data you need to monitor.

For example, if you must modify the workflow for an account activity, you navigate to Accounts > Accounts List > Activities.

2 Choose Help > About View.

About View identifies the Business object, Business components, and applets this view uses.

In the example of the Account Activities view, the dialog box identifies Action as the business component used by the Activities applet.

3 In Siebel Tools, navigate to the Business Components OBLE, then locate the business component object definition named Action.

Note the value in the Table property. In this example, the table name is S_EVT_ACT. You use this table name when you create a workflow policy component.

Next, determine the relationship between a business component and the primary workflow policy component.

To determine the relationship between the business component and the primary workflow policy component

1. Navigate to the Business Objects OBLE, then choose the object definition for the business object.

   In this example, choose the Account business object.

2. With the business object still chosen, navigate to the Business Object Components OBLE, choose the object definition with Action in the Bus Comp property, then click the hyperlink in the Link Property.

3. Note that the Source Field property is empty and the Destination Field property contains Account Id.

   The value in the Link property identifies the link that defines the relationship between the Account and the Action business components.

   This Link defines the relationship between the parent Business Component and the child Business Component through the Source field and Destination field.

   An empty Source Field property indicates that the join is using the ROW_ID column of the table defined in the Table property of the parent business component.

   The Destination Field property defines the field within the child business component that is a foreign key to the Business Component.

4. Click the hyperlink in the Child Business Component property.

   Clicking the hyperlink in the Child Business Component property navigates you to the object definition in the OBLE for the Action business component.

5. In the Object Explorer, expand the Business Component object type to expose the Field object type.

6. Choose the Account Id field, then note the value in the Column property.

   The Column property contains TARGET_OU_ID, which is the column in the table the Account Id field references. You use this information when you define the workflow policy component.

Manipulating and Processing Data

This topic provides reference information for manipulating and processing data when configuring a workflow process. It includes the following topics:

Accessing Run-Time Event Data

This topic explains how to access run-time event data from a workflow process.

To access run-time event data from a workflow process

1 Expose the WF Step Branch object type in the Object Explorer.

2 In the WF Step Branch OBLE, define the properties for a WF Step Branch object definition. Use the sample data described in the following table:

Property	Value
Name	Enter a name for the branch.
Type	Condition
Event Object Type	Applet
Event	InvokeMethod
Event Object	Choose the name of the event object that triggers the workflow.
Sub Event	NewRecord
Comments	Optional
Event Cancel Flag	The default is blank.

3 Set up process properties in the Process Designer.

For information, see "About Process Properties" on page 74.

4 Click the first step in your workflow process, then add a new output argument for each process property you need to populate that references a run-time event.

To add a new output argument, enter values into the argument fields displayed under the Output Arguments tab in the MVPW. Use the sample values described in the following table:

Argument Field	Value
Property Name	Enter a name for the property.
Type	Expression
Value	GetProfileAttr("RestrucOut") Note: As you are filling in the Profile Attribute field, note the name you give to each of the Profile Attributes property values. These are used in Step 9.
Output Argument	Leave default value.
Business Component Name	Leave default value.

Argument Field	Value
Business Component Field	Leave default value.
Comments	Optional

5 Activate the workflow process.

For more information, see "Process of Deploying a Workflow Process" on page 237.

6 Since the workflow process references run-time events, you must load the run-time events:

a Navigate to Administration-Runtime Events > Events.

b Click the Menu button on the Events list applet, then choose Reload Runtime Events.

7 Query the Event field for the run-time event you defined in Step 4.

8 Drill down on the Action Set Name field.

9 , Add a new Action for each process property that is populated by using a run-time event.

For example, you can create a new action to set the ACU Transaction ID and call it Set ACU Trans ID. Tasks you should perform for each new action include:

- Set the Action Type property to Attribute Set.
- Set the Profile Attribute to match the value you used in the GetProfileAttr call in the Process Designer inStep 4, such as TransType.
- Set the Set Operator value to Set.
- Use the expression builder to assign the Value field an appropriate value, such as the literal string FA-0001.
- Set the Sequence for the action to be less than the default sequence for the Workflow_XXXXXXX Action.
- Make sure the Sequence setting of the Workflow_XXXXXXX Action is the highest number so that this action happens after every other action.

NOTE: Whenever you modify the workflow process, you must repeat this step because modifying a workflow process resets the workflow action's sequence to 1.

10 From the applet menu, choose Reload Runtime Events.

Passing Data To and From a Workflow Process

This topic provides information about passing data to a workflow process and obtaining data from a workflow process. It includes the following topics:

The Workflow engine can be invoked by calling a business service. The Workflow Process Manager business service is a standard business service used for this purpose. When you run a workflow process by invoking the Workflow Process Manager business service, you can pass inputs to Workflow, and in some cases, obtain outputs from Workflow.

While this topic discusses passing data to and from a Business Service step, keep in mind that the Sub Process step uses the same conventions for passing property sets. For more information, see "About the In/Out Process Property" on page 80.

Passing Inputs to a Workflow Process

The input property set is required to contain a property named ProcessName, which specifies the name of the workflow process to be run. In addition to the ProcessName property, you can put other values, such as strings, numbers, and property sets, into the property set. These values are passed to the workflow process by the Workflow Process Manager business service.

Simple data type process properties, such as String, Number, and DateTime, that are marked In or In/Out are initialized if the input property set has a property in the top-level property set with a name matching the name of the workflow process property. The value of such a property in the input property set initializes the value of the matching workflow process property.

Hierarchical data type process properties that are marked In or In/Out are initialized if the input property set has a child whose property set Type field contains a string matching the name of the hierarchical workflow process property. If such a match is found, the matching child and everything below the child in the input property set is copied into the process property.

Note that multiple variables can be passed into SQL Program Arguments. For more information, see "Passing Multiple Variables into SQL Program Arguments" on page 453.

Passing Outputs from a Workflow Process

It is possible that a workflow process started programmatically will not return outputs. For example, an interactive workflow process can be started programmatically, but since it can pause, the output from the call to start the workflow process might reflect the state at an intermediate point. For this reason, only a workflow process that is guaranteed to run to completion in one call, that is, a service flows, can be expected to provide output in the output arguments of the call into the Workflow Process Manager business service.

Output arguments follow the same convention as input arguments. Simple workflow process properties, such as String, Number, and DateTime, that are marked Out or In/Out appear as properties on the top-level property set. Hierarchical process properties appear as children of the output property set. Hierarchical process properties can be located by examining the Type field of the child, which match the workflow process property name.

Passing Parameters from a Workflow Process to a Global Variable

You can use a business service to access and pass parameters from a workflow process to a global variable. Note that when an update is made to a global variable by using a Batch Manager task, each task is in essence a separate user session. Therefore, an update that is made to a global variable by using a Batch Manager task might not be visible to subsequent Workflow Process Batch Manager tasks. For more information, see "Example of Script That Accesses Workflow Parameters" on page 420.

Passing a Constant from a Workflow Policy Action into a Workflow Process

You can use the Run Workflow Process program to pass a constant from a Workflow Policy Action into a Workflow Process.

To configure a workflow to pass a constant from a workflow policy action into a workflow process

1. In Siebel Tools, navigate to the Workflow Policy Program OBLE and query for Run Workflow Process in the Name property.
2. With the Run Workflow Process object definition still chosen, navigate to the Workflow Policy Program Arguments OBLE.
3. Add a new argument, then choose File > Save from the application-level menu.

   For example, add a new argument with the Name property set to *IOName*. Make sure the Visible property has a check mark.
4. In the Workflow Policy Program OBLE, click the Run Workflow Process object definition.
5. From the application-level menu, choose Tools > Compile Selected Object, then click Compile in the Object Compiler dialog.

   In the Siebel client, this argument appears in the Arguments applet in the Workflow Policy Actions view.
6. Set the default value.

   The default value varies according to IOName.
7. Open the Process Designer for the workflow to which the constant is passed.
8. In the MVPW, create a new record with the Name argument set to the same name defined in the argument you created in Step 3.

   For example, Name must be the same as IOName. For more information, see "About Process Properties" on page 74.

9 Invoke the workflow policy.

For more information, see “Invoking a Workflow Process from a Workflow Policy” on page 137.

Examples of Script That Pass Data To and From a Workflow Process

This topic describes several examples of using script to invoke a workflow process and for passing parameters.

Example of Script That Invokes a Workflow Process and Constructs an Input Property Set

This topic gives one example of using script to invoke a workflow process and construct an input property set. You might use this feature differently, depending on your business model.

The following is a sample script that invokes the Workflow Process Manager business service which constructs an input property set, psInputs, for the business service. This script defines strings that are put into the input property set as properties:

```
var msgName = "Siebel Agent Authorization Retrieval";

var reqSubType = "CICS Services Request";

var reqType = "AgentAuthorizationReq";

var CICSServiceName = "Consumer Auto Agent Authorization Retrieval";

var processName = "Consumer Auto VBC VBC Template";

var reqFileName = "C:\\sea752\\XMLMessages\\AgentAuthorizationVBCReq-final.xml"

var resFileName = "C:\\sea752\\XMLMessages\\AgentAuthorizationVBCResponse-
final.xml"
```

Example of Script That Defines Property Sets for the Input Property Set

This topic gives one example of using script to define property sets for the input property set. You might use this feature differently, depending on your business model.

The following is a sample script that defines property sets, which are put into the input property set as child property sets:

```
//Request PS

var psRequest = app.NewPropertySet();

var psAgentNumTag = app.NewPropertySet();

var psType = app.NewPropertySet();

var sAgentID;
```

Example of Script That Constructs Property Sets

This topic gives one example of using script to construct property sets. You might use this feature differently, depending on your business model.

The following is a sample script that constructs property sets:

```
//Build property set hierarchy

sAgentID = app.LoginName();

psRequest.SetType("XMLHierarchy");

psAgentNumTag.SetType("DataAgentNumber");

psAgentNumTag.SetValue(sAgentID);

psRequest.AddChild(psAgentNumTag);
```

Example of Script that Assembles Properties and Child Property Sets into the Input Property Set

This topic gives one example of using script to assemble properties and child property sets in the input property set. You might use this feature differently, depending on your business model.

The following is a sample script that assembles properties and child property sets into the input property set:

```
psInputs.AddChild(psRequest);//Pass in Property Set

psInputs.SetProperty("RequestURLTemplate", requestURLTemplate);//Pass in string

psInputs.SetProperty("RequestSubType", reqSubType);

psInputs.SetProperty("ReqType", reqType);

psInputs.SetProperty("MessageName", msgName);

psInputs.SetProperty("CICSServiceName", CICSServiceName);

psInputs.SetProperty("ProcessName", processName);

psInputs.SetProperty("Request File Name", reqFileName);

psInputs.SetProperty("Response File Name", resFileName);
```

Example of Script That Invokes the Workflow Process Manager Business Service and Passes the Input Property Set

This topic gives one example of using script to invoke the workflow process manager business service and pass the input property set. You might use this feature differently, depending on your business model.

The following is a sample script that invokes the Workflow Process Manager business service and passes the input property set to the Workflow Process Manager business service:

```
var svc = TheApplication(). GetService(“Workflow Process Manager”);
```

```
svc.InvokeMethod("RunProcess", psInputs, psOutputs);//Call the Workflow

var sErr = psOutputs.GetProperty("sErr");//Check the Workflow status
```

Example of Script That Accesses Workflow Parameters

This topic describes how to use script to access workflow parameters for a running workflow process.

To access workflow parameters for a running workflow process

1. Define a business service with relevant methods and parameters.
2. Access the business service from the workflow process.
3. In the business service step in the workflow process, pass the workflow process properties to the business service method arguments.

   For more information, see "Passing a Process Property In and Out of Workflow Steps" on page 81.
4. Use the following script to take the business service argument values and assign them to Profile Attributes:

```
function Service_PreInvokeMethod (MethodName, Inputs, Outputs)

{

if( MethodName == "XXX" ) {

var isWorkflowRunning, viewValidCurrent, viewValidNext;

// read the input arguments into profile attributes

isWorkflowRunning = Inputs.GetProperty("Workflow Running");

viewValidCurrent = Inputs.GetProperty("Valid View Current");

viewValidNext = Inputs.GetProperty("Valid View Next");

TheApplication().SetProfileAttr("WFRunning", isWorkflowRunning);

TheApplication().SetProfileAttr("WFViewCurrent", viewValidCurrent);

TheApplication().SetProfileAttr("WFViewNext", viewValidNext);

}
```

5. Use the profile attributes for more processing.

   The necessary information is put into the profile attributes of the application. You can use the standard procedures for accessing the profile attributes to extract this information. For more information, see *Siebel Personalization Administration Guide*.

Using the Timestamp

You can use the timestamp to get the current system time and to do time arithmetic based on the current time.

The arithmetic involving time information for a workflow process is different from that for a workflow policy program. The second operand of the 'Timestamp () ' function must be provided in a scale of minutes, that is, as a fraction of the whole day.

For example, if the intended result of an arithmetic operation is 30 minutes, then the argument appears as described in the following syntax:

```
Timestamp()+0.021
```

The operation is explained as follows:

- 0.021 = 30/(24*60).
- (24*60) represents a whole day in minutes.
- 30 represents the required minutes.

For more information date calculations with a workflow policy, see "Entering Date Calculations in the Conditions Applet" on page 211.

Manipulating Data Validation Rules

For information on manipulating data validation rules, see *Siebel Order Management Infrastructure Guide*.

Reference of Workflow Process Object Properties

This topic provides reference information for various objects used with a workflow process. It includes the following topics:

Reference of Workflow Process Object Definition Properties

Table 100 describes properties for the overall workflow process object definition.

Table 100. Description of Properties for a Workflow Process Object Definition

Property	Description	Possible Value
Auto Persist	Sets workflow persistence.	(Optional) YES or NO
Business Object	The name of the associated business object.	(Optional) This value is chosen from a picklist of business objects. Only business objects with a defined primary component appear in this picklist. For more information on defining the primary business component for a business object, see "Defining a Primary Business Component for a Business Object" on page 105 .
Error Process Name	The name of the workflow process to call as the error process.	(Optional) Choose from the picklist of predefined workflow processes.
Group	This property applies to workflows defined prior to version 7.7.	(Optional) **NOTE:** Do not use this property for new workflows you create. Use the Project property instead.
Pass By Ref Hierarchy	Applicable for hierarchal data passed between a sub process and the parent workflow process. If "pass by reference" is checked, then changes made in the child workflow process propagate back to the parent workflow process. When implemented, this feature reduces memory footprint required for a workflow process as data is passed by reference instead of by value.	(Optional) TRUE or FALSE
Process Name	The name of the process.	(Required) A descriptive name that is: ■ Consistent with your overall naming strategy ■ Meaningful to the designer of the process ■ Unique

Table 100. Description of Properties for a Workflow Process Object Definition

Property	Description	Possible Value
Project	The name of the project to which the workflow process belongs.	(Required)
Replication Level	How workflows synchronize to Mobile Web Clients. The All value synchronizes to the server's regional nodes as well as mobile users, whereas the Regional value synchronizes just to the server's regional nodes.	None, All, or Regional
State Management Type	The workflow process state management type. Describes the type of business service requests made while the workflow process is being executed. **NOTE:** If every service invoked by this workflow process is a stateless business service, then it is recommended the state management type be set to Stateless. Otherwise, it is recommended the state management type be set to Stateful. The default setting is Stateful. For more information, see "State Management Type and Workflow Processes" on page 424.	(Required). Possible values include: ■ **Stateless.** Business service method(s) invoked by this workflow process have no dependency on a service-specific state. Choose this setting if the stateful/cached services invoked by this workflow can be released when the workflow pauses or finishes. ■ **Stateful.** Business service method(s) invoked by this workflow process depend on a service-specific state. Choose this setting if the stateful/cached services invoked by this workflow must not be released during and after the execution of the workflow.
Status	The current status of the process.	(Required). Possible values include: ■ Not In Use ■ In Progress ■ Completed
Version	The version number of the process definition.	Read-only. The default version is 0. The version number increments by one when you use the Revise button to modify an existing process definition.

Table 100. Description of Properties for a Workflow Process Object Definition

Property	Description	Possible Value
Web Service Enabled	This property indicates that a business service or workflow is exposable as a Web service and can be called independently. **NOTE:** Setting this property to TRUE does not automatically implement the workflow Web service, nor does configuring the workflow process for Web service cause the flag to be checked.	(Optional) TRUE or FALSE
Workflow Mode	The workflow process type.	(Required). Possible values include: ■ 7.0 Flow ■ Long Running Flow ■ Interactive Flow ■ Service Flow For more information, see "About the Workflow Mode Property" on page 121.

Note that some of these properties, but not all of them, can be defined by using the Workflow Processes OBLE, or the Properties window in the Process Designer.

The Properties window in the Process Designer displays properties based on context. To display properties for the workflow process, right-click in the design canvas, make sure no step or connector is chosen, then choose View > Windows > Properties Window from the Tools application-level menu:

State Management Type and Workflow Processes

The OM-session state management framework requires business services to have a defined state management type. A business service must be defined as Stateless or Stateful, or as a Server managed business service. The stateless Workflow Process Manager business service is the workflow engine.

The State Management Type property in the Workflow Process Properties window of a workflow process defines whether a workflow process is designed to be stateless or stateful. The Workflow engine can execute a *stateful* workflow process or a *stateless* workflow process:

- When a workflow process is defined as Stateless, the OM-session state management framework can release stateful or cached business services invoked by this workflow process when the workflow instance is paused or finished.

- When a workflow process is defined as Stateful, the OM-session state management framework does not release stateful or cached business services invoked by this workflow instance even after the instance is finished. Instead, the framework waits for the caller of the Workflow engine to decide when the cached services must be released. The caller of the Workflow engine can release stateful or cached business services by calling methods from Web Channel Dedicated Block Service.

Defining the Workflow Process State Management Type

By default, the workflow process State Management Type is set to Stateful to preserve the pre-8.1 behavior. However, if a workflow process is designed to be executed through Web Channel, it might be necessary to change the workflow's State Management Type to Stateless to take advantage of the session pooling feature in the OM-session state management framework.

Table 101 describes how the Workflow Mode property for a workflow process determines the workflow's State Management Type.

Table 101. Description of How to Set the State Manage Type Property

Workflow Mode	How Typically Defined	Description
Service Flow	Stateless	Since a service workflow cannot have a User Interact step or Wait step, it is always executed in the workflow's entirety before the object manager session state management framework is given a chance to release the session back to the pool.
Interactive Flow	Stateless	Similar to a long-running workflow, an interactive workflow instance can move between sessions through the Universal Inbox. However, since the State Management Type setting is only relevant when the workflow process is executed through Web Channel, and since an interactive workflow always executes in a user-interactive session, the State Management Type has no effect on the behavior of an interactive workflow.
Long Running Flow	Stateless	The Workflow engine is already designed to allow a long-running workflow to deploy between sessions when a long-running workflow is being paused. **NOTE:** It is recommended you not configure a long-running workflow to invoke a stateful business service since the service state is lost when a long-running workflow instance moves from one user session to another.
7.0 Flow	Stateful	Do not change this setting unless you are sure none of the business services the workflow invokes directly or indirectly is a stateful business service.

For more information on the Web Channel OM-session state management framework, see *Siebel Web UI Dynamic Developer Kit Guide*.

Reference of Workflow Step and Connector Properties

This topic provides property descriptions of the various step types used in Siebel Workflow.

Table 102 describes properties of a workflow process step.

Table 102. Description of Properties of a Workflow Step

Property	Type of Step	Description	Possible Value
Name	All	The name of the step.	A descriptive name that is: ■ Consistent with your overall naming strategy ■ Meaningful to the designer of the process ■ Unique When you create a new step, the step is automatically assigned a name, based on the step's type, and a sequence number. You can change this name or leave it as is. For more information, see "About Naming a Process Step or Process Property" on page 88.
Type	All	The type of step.	This value is automatically entered when you create the step in the Process Designer view. This is a read-only property.
Business Component	Siebel Operation	Required. The business component that performs the action you specify.	Choose from a list of business components that have been defined for the chosen business object.
Business Service Name	Business Service	The name of the service to invoke.	The picklist displays business services existing in Siebel Tools with the Hidden flag set to FALSE. For more information, see "Making a Business Service Visible to a Workflow Process" on page 93.
Business Service Method	Business Service	The name of the method to invoke on the service.	The picklist displays methods defined for the chosen business service.

Table 102. Description of Properties of a Workflow Step

Property	Type of Step	Description	Possible Value
Subprocess Name	Sub Process	The name of the Sub Process step.	A descriptive name that is: ■ Consistent with your overall naming strategy ■ Meaningful to the designer of the process
Error Code	Stop	A number associated with a string in the database that comprises the error message.	Numeric value.
Error Message	Stop	A string in the database that comprises the error message.	Text string.
User Interact View	User Interact	The name of the view for the user interact step.	Choose from a picklist containing predefined view names.
Operation	Siebel Operation	The type of operation.	The possible values are: ■ Delete ■ Insert ■ NextRecord ■ PrevRecord ■ Query ■ QueryBiDirectional ■ Update ■ Upsert
Maximum Iterations	Wait	The maximum number of times you can execute this step within a process instance.	When the maximum number of iterations is reached, an Object Manager error is generated and the workflow process returns an In Error status. If you need the process to run to completion, you must use a Workflow exception mechanism, such as an error process or exception branch, to catch and handle the error. For more information, see "Using an Error Exception to Handle Errors" on page 157.

Table 102. Description of Properties of a Workflow Step

Property	Type of Step	Description	Possible Value
Description	All	A text narrative describing the purpose of the step.	Free form text.
Comments	All	A text narrative describing the purpose of the step.	Free form text.
Update Snapshot	All	This parameter is used for recovery. Update Snapshot indicates that when the process reaches this step, Workflow takes a snapshot of the process state, so that if there is a system failure, you can recover the state.	Check mark.
Processing Mode	Wait	The mode in which the process is run when triggered by a run-time event. For more information, see "About the Wait Step Processing Mode Property" on page 89.	(Optional) **Local Synchronous.** Executes the process in the application object manager. This is the default. **Remote Synchronous.** Submits a synchronous request to the Workflow Process Manager server component to execute the process. **Remote Asynchronous.** Submits an asynchronous request to the Workflow Process Manager server component to execute the process.

Table 103 describes properties for the Start step.

Table 103. Description of Properties of the Start Step

Start Properties	Expected Value
Comments	(Descriptive text.)
Description	(Descriptive text.)
Inactive	FALSE
Name	Start
Parent Name	Workflow name:version
Processing Mode	Local Synchronous (or can be empty)

Table 104 describes properties for the Business Service step.

Table 104. Description of Properties of the Business Service Step

Business Service Property	Expected Value
Allow Retry Flag	FALSE
Business Service Name	(Choose the business service name from the picklist.)
Business Service Method	(Choose the business service method from the picklist.)
Comments	(Descriptive text.)
Description	(Descriptive text.)
Inactive	FALSE
Name	Business service 0 (You can modify this value, as necessary.)
Parent Name	Workflow name:version

Table 105 describes properties for the Decision Point step.

Table 105. Description of Properties of the Decision Point

Decision Point Properties	Expected Value
Comments	(Descriptive text.)
Description	(Descriptive text.)
Inactive	FALSE
Name	Decision Point 0 (You can modify this value, as necessary.)
Parent Name	Workflow name:version

Table 106 describes properties for the Sub Process step.

Table 106. Description of Properties of the Sub Process Step

Subprocess Properties	Expected Value
Comments	(Descriptive text.)
Description	(Descriptive text.)
Inactive	FALSE

Table 106. Description of Properties of the Sub Process Step

Subprocess Properties	Expected Value
Name	Sub Process 0 (You can modify this value, as necessary.)
Sub Process Name	(Choose the subprocess name from the picklist.) For a subprocess to appear in this picklist of defined workflow processes, the subprocess must have a status of Complete.

Table 107 describes properties for the Siebel Operation step.

Table 107. Description of Properties of the Siebel Operation Step

Siebel Operation Properties	Expected Value
Allow Retry Flag	FALSE
Business Component	(Choose the business component name from the picklist.)
Comments	(Descriptive text.)
Description	(Descriptive text.)
Inactive	FALSE
Name	Siebel Operation 0 (You can modify this value, as necessary.)
Operation	(Choose the operation from the list.)
Parent Name	Workflow name:version

Table 108 describes properties for the Task step.

Table 108. Description of Properties of the Task Step

Task Properties	Expected Value
Comments	(Descriptive text.)
Description	(Descriptive text.)
Inactive	FALSE
Name	Task 0 (You can modify this value, as necessary.)
Task Name	(Choose the task name from the picklist.)

Table 109 describes properties for the User Interact step.

Table 109. Description of Properties of the User Interact Step

User Interact Properties	Expected Value
Comments	(Descriptive text.)
Description	(Descriptive text.)
Inactive	FALSE
Name	User Interact 0 (You can modify this value, as necessary.)
Parent Name	Workflow name:version
User Interact View	(Choose the view name from the picklist.)

Table 110 describes properties for the Wait step.

Table 110. Description of Properties of the Wait Step

Wait Properties	Expected Value
Business Service Method	Sleep
Business Service Name	Workflow Utilities
Comments	(Descriptive text.)
Description	(Descriptive text.)
Inactive	FALSE
Maximum Iterations	
Name	Wait 0 (You can modify this value, as necessary.)
Parent Name	Workflow name:version

Table 111 describes properties for the Stop step.

Table 111. Description of Properties of the Stop Step

Stop Properties	Expected Value
Comments	(Descriptive text.)
Description	(Descriptive text.)
Error Code	(Choose the error code from the picklist.)
Error Message	(Once the Error code is chosen the Error message is automatically populated.)

Table 111. Description of Properties of the Stop Step

Stop Properties	Expected Value
Name	Stop 0 (You can modify this value, as necessary.)
Parent Name	Workflow name:version

Table 112 describes properties for the End step.

Table 112. Description of Properties of the End Step

Stop Properties	Expected Value
Comments	(Descriptive text.)
Description	(Descriptive text.)
Inactive	FALSE
Name	End 0 (You can modify this value, as necessary.)
Parent Name	Workflow name:version

Table 113 describes properties for the workflow Connector.

Table 113. Description of Properties of the Workflow Connector

Property	Description	Possible Value
Name	The name of the connector.	The connector name must be unique to the workflow process. If it is not unique, you cannot commit the record.
Type	The type of connector.	Type can have one of the following values: **Default.** Indicates that if no other conditions are satisfied, this connector is followed. Also, if Default is used, conditions defined for the connector are ignored. **Condition.** Indicates that a condition is defined for the connector. **Connector.** Indicates there is no condition branching involved. **Error Exception.** Indicates there is error exception handling. Error Exception captures system errors, such as an error noting that the Assignment Manager server component is not available. For more information, see "Using an Error Exception to Handle Errors" on page 157. **User Defined Exception.** Indicates there is user-defined exception handling. User Defined Exception captures user-defined errors, such as an error noting that an order being submitted is incomplete. For more information, see "Using an Error Exception to Handle Errors" on page 157.
Event Object Type	Describes the type of object on which a run-time event occurs. **NOTE:** Event Object must be chosen if Event Object Type is defined.	(Optional) Application. Applet. BusComp.
Event Object	The name of the object to which the event occurs.	Required if Event Object Type is specified. This is the name as defined in Tools. Event Object is context-sensitive, depending on the value in Event Object Type. For example, if Event Object Type is set to BusComp, then Event Object specifies the business component, such as Account.

Table 113. Description of Properties of the Workflow Connector

Property	Description	Possible Value
Event	The specific event that happens to the object.	Required if Event Object Type is specified. The set of available events is context-sensitive, depending on the value in Event Object Type.
Subevent	The name of the method or business component field to be monitored.	(Optional) Used when the object type is BusComp or Applet and the event is InvokeMethod or SetFieldValue. For InvokeMethod, the name of the method being invoked. For SetFieldValue, the name of the field being set.
Event Cancel Flag	Aborts the run-time event after executing the process. **NOTE:** If this flag is not checked, the following error results when the workflow process is run: *The specialized method [Method Name] is not supported on this business component*.	(Optional) This flag only applies to events that can be canceled. This flag works similar to CancelOperation in scripting.
Expression	User friendly text for the condition defined for the connector.	Read-only
Event Visibility	Controls whether the workflow process waits for a run-time event generated within the current session, or by other sessions.	If the workflow process is persistent, the visibility can be set to Enterprise or Local. **NOTE:** If the workflow process is not persistent, it is recommended that visibility be set to Local. In most cases where a workflow needs to wait for an event, the workflow must be persistent, and you set the value to Enterprise. However, setting Event Visibility to Enterprise can have a negative performance impact, because a run-time event searches for a matching instance. Therefore, it is recommended that if your workflows do not have to be persistent, that you set Event Visibility to Local.

Table 113. Description of Properties of the Workflow Connector

Property	Description	Possible Value
User Event Name	An arbitrary string that denotes the name of a user event.	This can be a string but it must be unique within the Siebel enterprise. Make sure you define a unique name on the User Event Name field. Also, make sure the name is sufficiently long so that it remains unique across the Siebel enterprise. For example: *Order Placed-Begin Processing Event for Service Request Automation-Version 2*.
User Event Storage	The process property that serves as the destination for the payload on the incoming user event.	This value can be a process property. The process property must be marked as the correlator.
User Event Timeout (Days)	The amount of time, in days, before the event times out.	Numeric value. If the user event is on a wait step, use this parameter. Do not specify a wait duration on the wait step.
Comments	More statements relative to the connector.	Free form text.

Table 114 describes properties for the Error Exception.

Table 114. Description of Properties of the Error Exception

Error Exception Properties	Expected Value
Comments	(Descriptive text.)
Inactive	FALSE
Name	Error Exception 0 (You can modify this value, as necessary.)
Parent Name	(From step name)
Type	Error Exception
User Event Name	(Leave this field empty.)
User Event Storage	(Leave this field empty.)
User Event Timeout	0

Reference of Process Properties and Arguments

This topic provides reference information for process properties and their arguments. It includes the following topics:

Argument Fields for a Process Property

This topic describes argument fields that can be defined for a process property in the MVPW. For configuration information, see "About Process Properties" on page 74.

Table 115 describes argument fields that can be defined for a process property.

Table 115. Description of Argument Fields for a Process Property

Argument Field	Description	Possible Value
Name	The name of the process property.	A descriptive name that is: ■ Consistent with your overall naming strategy ■ Meaningful to the designer of the process ■ Unique For more information, see "About Naming a Process Step or Process Property" on page 88.
Display Name	A user friendly version of the property name.	The Display Name can be the same as or different from the Name.

Table 115. Description of Argument Fields for a Process Property

Argument Field	Description	Possible Value
In/Out	Describes whether or not the process property is passed in or out of the process, passed into the process and returned, or used only within the process.	Possible values include: ■ **In.** The process property is passed into the workflow process. Binary types cannot be assigned this value. ■ **Out.** The process property is passed out of the workflow process. Binary types cannot be assigned this value. ■ **In/Out.** The process property is passed into the workflow process and returned. Binary types cannot be assigned this value. ■ **None.** The process property is used only within the workflow process.
Business Object	The name of the associated business object.	Read only. For more information, see "Defining a Primary Business Component for a Business Object" on page 105.
Business Component	The name of the business component containing the virtual field.	This value is chosen from a picklist of business components belonging to the workflow process business object.
Virtual Field	The name of the business component field mapped to the workflow process property.	This value is chosen from a picklist of fields belonging to the business component. Only calculated fields with no calculated values appear in this picklist.
Default String	Initial value if the process property is a string type.	Free form text. If you enter [Value], the process property is initialized with the value in the Value property of the workflow input property set.
Default Date	Initial value if the process property is a date type.	Calendar widget.

Table 115. Description of Argument Fields for a Process Property

Argument Field	Description	Possible Value
Data Type	The type of data that can be stored in the process property.	Possible values include: ■ **Alias.** Size limit for default value, not run-time value: 250 characters. ■ **Binary.** For variant or binary information, usually for XML data. Binary types must be assigned the None value in the In/Out property. ■ **Date.** For dates. ■ **Hierarchy.** Data type used by Siebel Enterprise Application Integration to store data from a property set. For more information, see *Overview: Siebel Enterprise Application Integration*. ■ **Integration Object.** For storing integration objects. Integration objects have the Hierarchy data type. ■ **Number.** For numeric data. Size limit for default value, not run-time value: 22 digits. ■ **String.** For alphanumeric data, usually for UTF-16 data. Size limit for default value, not run-time value: 250 characters. ■ **Strongly Typed Integration Obj.** A special data type for exposing a workflow process as a Siebel inbound Web service. Siebel business services use this property when creating WSDLs. **NOTE:** For both the workflow process and business service engines, values in the Strongly Typed Integration Obj and the Integration Object properties are treated as the same data type.
Default Number	Initial value if the process property is a numeric type.	Numeric widget.
Integration Object	Data type used by Siebel EAI to store data from an integration object.	For example: “Account-Get Oracle Customer (Oracle)”

Table 115. Description of Argument Fields for a Process Property

Argument Field	Description	Possible Value
Correlator Flag	Makes the process property ready for use as a correlator: a piece of business data that identifies the recipient of the incoming message. For more information, see "About the Workflow User Event Service Business Service" on page 381.	Check mark.
Access Mode	Controls whether a process property is read-only or read-write.	Read-only or Read-write.

Argument Fields for an Input Argument

Table 116 describes argument fields that can be defined on an input argument for the Business Service step, Sub Process step, and Wait step.

Table 116. Description of Argument Fields for an Input Argument

Argument Field	Description	Possible Value
Preferred Sequence (For business service steps only.)	(This cell is left intentionally empty.)	(This cell left is intentionally empty.)
Input Argument (For business service steps only.)	The name of the input argument.	(Required.) The picklist displays input arguments existing for the chosen business service method. A method argument appears in this picklist if it is defined as a business service method argument, the Hidden flag is set to FALSE, and the type is input or input/output.
Subprocess Input (For Sub Process steps only.)	The name of the input argument.	(Required.)
Type	The type of argument.	(Required.) Choices in the picklist include: ■ Literal ■ Process Property ■ Business Component ■ Expression

Table 116. Description of Argument Fields for an Input Argument

Argument Field	Description	Possible Value
Value	A string value.	For Literal and Expression type input arguments. This can be a picklist, depending on the argument chosen. String values can only be a maximum of 32,767 characters.
Property Name	The name of the process property.	For Process Property–type input arguments.
Business Component Name	The name of a business component within the business object of the business process.	For Business Component–type input arguments.
Business Component Field	The name of a field within the business component.	For Business Component Field–type input arguments.
Changed	(This cell is intentionally empty.)	Check mark.

Argument Fields for an Output Argument

Table 117 describes argument fields that can be defined on an output argument for a business service step, sub process step, and siebel operation step.

Table 117. Description of Argument Fields for an Output Argument

Argument Field	Description	Possible Value
Property Name	The name of the Process Property to store the results.	(Required.) When Property Name is clicked, a picklist of process properties that have been defined for the process appears. For more information, see "Passing a Process Property In and Out of Workflow Steps" on page 81.
Type	The type or argument.	(Required.) Choices in the picklist include: ■ Literal ■ Output Argument ■ Business Component ■ Expression
Value	A string value.	For Literal or Expression arguments. Note that string values can only be a maximum of 32,767 characters.

Table 117. Description of Argument Fields for an Output Argument

Argument Field	Description	Possible Value
Output Argument (For business service steps only.)	The name of the output argument.	For Output Arguments type. This is a picklist of output arguments for the chosen method. An argument appears in this picklist if it is defined as a business service method argument, the Hidden flag is set to FALSE, and the type is Output or Input/Output.
Subprocess Output (For Sub Process steps only.)	The name of the output argument.	For Output Arguments type.
Business Component Name	The name of the business component within the business object of the business process.	For Business Component type.
Business Component Field	The name of a field within the business component.	For Business Component Field type. **NOTE:** A business component field based on a multi-value group cannot be chosen as a value for an input or output argument. If you need to use a field based on a multi-value group, you must define a business component for the field and link it to the appropriate business object. For more information, see *Configuring Siebel Business Applications*.

Argument Fields for a Recipient Argument

Table 118 describes argument fields for a recipient argument. For configuration information, see "About the Sub Process Step" on page 95.

Table 118. Description of Argument Fields for a Recipient Argument

Argument Field	Description	Define This Property If. . .	Possible Value
Recipient Type Code	The type of recipient.	(This cell is intentionally empty.)	This value is fixed as "User" and cannot be changed.
Value Type Code	The source from which the recipient value comes.	(This cell is intentionally empty.)	Values in the picklist for this property include: ■ Name ■ Expression ■ Process Property ■ Business Component
Recipient Name	The name of the recipient. This property is a pick applet which displays the first name, last name, and login name of users in the database.	The Value Type Code is set to Name.	Choose one name from the list of users available in the database.
Business Component Name	The name of the business component.	The Value Type Code is set to Business Component.	Choose one business component.
Business Component Field	The name of the business component field.	The Value Type Code is set to Business Component.	Choose one business component field.
Process Property Name	The name of the process property.	The Value Type Code is set to Process Property.	Choose one process property.
Expression	If the recipient value is derived from an expression, the expression is entered in this property.	The Value Type Code is set to Expression.	The expression from which the recipient value is derived.

Argument Fields for a Search Specification Input Argument

Table 119 describes argument fields that can be defined for a search specification input argument on the Search Spec Input Arguments tab of the MVPW.

Table 119. Description of Argument Fields For a Search Specification Input Argument

Argument Field	Description
Expression Business Component	If you entered Expression in the Type argument field, enter the name of the business component that evaluates the expression. For example, in the Search Specification argument field, you can enter: "[Due Date] < '" + [Order Date] + "'" The Expression business component evaluates Order Date so that the search specification becomes: [Due Date] < '07/04/2001 18:51:26'
Filter Business Component	Enter the name of the business component that provides the group of records on which you perform your search.
Search Specification	If you entered Literal in the Type argument, enter a literal value in the form of an expression. For example, = 100. If you entered Expression in the Type argument, enter an expression such as [Status] LIKE '*Open*'. The expression is evaluated by the Expression business component you specify.
Type	Required. Choose the type of value on which to base your search: Literal or Expression.
Comments	Enter a text description of the purpose of the search.
Changed	Check mark indicates a changed search specification.

Reference of Workflow Policy Object Properties

This topic provides reference information for various objects used with the workflow policies module. It includes the following topics:

Properties of the Workflow Policy Column

The Workflow Policy Columns OBLE displays a list of the available workflow policy columns. Note that you must activate extension columns in Siebel Tools to make them available for use in workflow database operations.

Table 120 describes properties displayed in the Workflow Policy Columns OBLE.

Table 120. Description of Properties of the Workflow Policy Column

Property	Usage	Description
Name	Define the name of the workflow policy column. This is the default name that appears in the Conditions applet on the Workflow Policies view.	A name that is: ■ Consistent with your overall naming strategy. ■ Meaningful to the policy maker. ■ Descriptive of how the column is used.
Changed	The identifier for whether the record was added or edited.	A check mark or a blank value.
Project	Define the project the workflow policy column belongs to. The project must be locked by you before you can modify the column.	A project from the picklist of projects you currently have checked out.
Table Name	Define the name of the Siebel database table that contains the column.	A table name from the picklist of Siebel database tables.
Column Name	Define the name of the column in the Siebel table.	A database column on the database table specified in Table Name.

Table 120. Description of Properties of the Workflow Policy Column

Property	Usage	Description
Picklist	Define the picklist used when choosing a comparison value for the column in the Workflow Policies view.	A picklist defined in the repository. The column chosen has a corresponding Business Component field. If the corresponding Business Component field has a picklist defined, the picklist is entered here. For more information on picklists, see *Configuring Siebel Business Applications*.
Source Field	Define the field in the business component of the picklist that is the source of the comparison value.	A Business Component field name from the picklist specified in the Picklist field.
Applet	Define the applet used to display the picklist in the Workflow Policies view.	An applet chosen from the picklist. Only choose a Pick applet.
Inactive	Specify whether this column is active or inactive. If column is inactive, the column is not compiled when you compile your .srf and is not accessible by an object.	A check mark indicates this is inactive and is not compiled or accessible.
Comments	Add comments describing the purpose or use of column.	Enter comments text.

Properties of the Workflow Policy Object

The Workflow Policy Objects OBLE displays a list of the available workflow policy objects.

Table 121 describes properties displayed in the Workflow Policy Objects OBLE.

Table 121. Description of Properties of the Workflow Policy Object

Property	Usage	Description
Name	Define the name of the workflow policy object.	A descriptive name that is: ■ Consistent with your overall naming strategy. ■ Meaningful to the policy maker.
Changed	The identifier for whether the record has been added or edited.	A check mark indicates the record has been added or edited.

Table 121. Description of Properties of the Workflow Policy Object

Property	Usage	Description
Inactive	Specify if the object is active or inactive.	A check mark indicates this field is inactive and is not compiled or accessible. If object is inactive, the object is not compiled when you compile your .srf and is not accessible by another object or policy.
Comments	Add comments relating to the workflow policy object.	Descriptive text.
Project	Define the project name.	Defined in the project picklist.

Properties of the Workflow Policy Component

The Workflow Policy Components OBLE displays a list of workflow policy components for the parent workflow policy object currently chosen in the OBLE. The Workflow Policy Components OBLE displays both the primary policy component and nonprimary policy components and how each of the policy components are related.

Table 122 describes properties displayed in the Workflow Policy Components OBLE.

Table 122. Description of Properties of the Workflow Policy Component

Property	Usage	Description
Name	Define the Name of the workflow policy component.	A descriptive name that is: ■ Consistent with your overall naming strategy. ■ Meaningful to the policy maker.
Changed	Indicates whether the record has been added or edited.	A check mark indicates the record has been added or edited.
Primary	Specify whether this workflow policy component is primary for the workflow policy object chosen in the Workflow Policy Objects OBLE.	A check mark indicates this is the primary workflow component. Note: Each workflow policy object must have only one primary workflow policy component.
Source Table Name	Define the table that the workflow policy component is based on.	A table name from the picklist.
Source Column Name	Define the column in the source table that relates to another workflow policy component.	A picklist of columns from the table specified in the Source Table Name field. (Not required for the primary workflow policy component.)

Table 122. Description of Properties of the Workflow Policy Component

Property	Usage	Description
Target Component Name	Define the target workflow policy component that this workflow policy component is related to.	A table name from the picklist. (Not required for the primary workflow policy component.)
Target Column Name	Define the column in the target workflow policy component that the source column in this workflow policy component is joined to.	A picklist of columns from the workflow policy component specified in the Target Component Name field. (Not required for the primary workflow policy component.)
Inactive	Indicates if the component is active or inactive.	A check mark indicates this field is inactive and is not compiled or accessible. If the component is inactive, it is not compiled when you compile your .srf and is not accessible by a policy.
Comments	Add comments for the workflow policy component.	Descriptive text.

How the Primary Component and Non-Primary Components are Represented in the OBLE

The Primary property provides a visual indication of which component is the primary component and which components are the nonprimary components.

For example, in Figure 26 the Workflow Policy Component named Account has the primary property checked, which defines Account as the primary component. Other components listed in the Workflow Policy Components OBLE are nonprimary components of the Workflow Policy Component named Account.

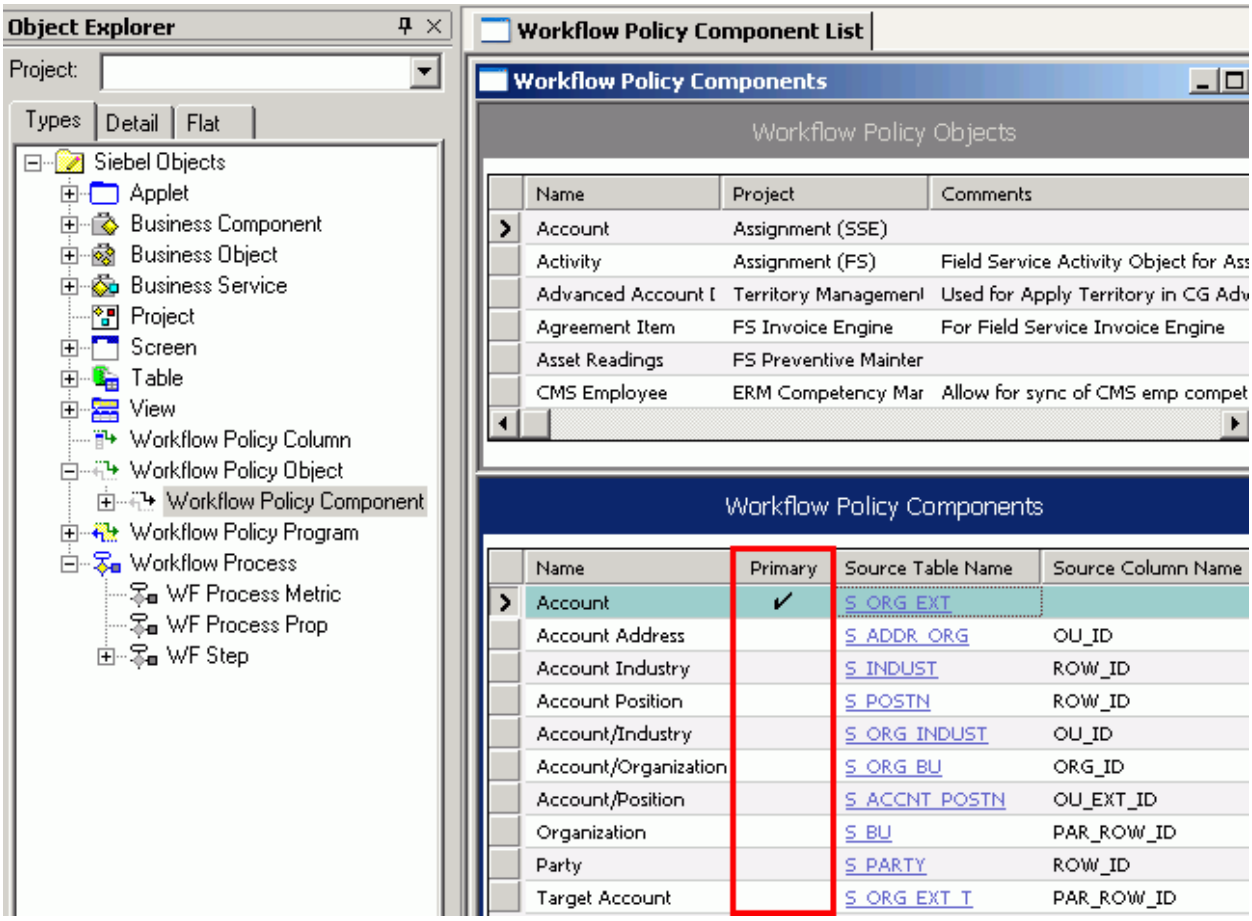


Figure 26. Representation of the Primary and Non-Primary Components in the OBLE

Properties of the Workflow Policy Component Column

The Workflow Policy Components Columns OBLE displays a list of the policy columns that can be monitored for the parent Workflow Policy Component that is currently chosen in the OBLE.

Table 123 describes properties displayed in the Workflow Policy Component Columns OBLE.

Table 123. Description of Properties of the Workflow Policy Component Column

Property	Usage	Comment
Workflow Column Name	Define the name of the column defined in the workflow Policy Component Column view.	A picklist of columns that were defined in the Workflow Policy Column view for the table that the workflow policy component is based on.
Alias	Define the name of the column as it appears in the Conditions Field picklist in the Workflow Policies view.	A descriptive name that is: ■ Consistent with your overall naming strategy. ■ Meaningful to the policy maker. ■ Descriptive of how the column is used. The default is the workflow policy column name.
Changed	Indicates whether the record was added or edited.	A check mark indicates that the record has changed.

Properties of the Workflow Policy Program

Table 124 describes properties displayed in the Workflow Policy Programs OBLE.

Table 124. Description of Properties of the Workflow Policy Program

Property	Required	Usage
Name	Yes	Define the name of the action to perform. This name is exposed in the Actions view in the Siebel client.
Changed	No	Indicates recent modifications.
Project	No	Define the Name of the project as defined in the project picklist.

Table 124. Description of Properties of the Workflow Policy Program

Property	Required	Usage
Type	Yes	Types you can choose from the picklist include: ■ **DB Operation.** Insert or update a database table based on arguments. ■ **External Program.** Execute an external program in Windows. ■ **Send Message.** Compose and send an automatic email message. ■ **Send Page.** Send a page to a pager. ■ **Send Broadcast Message.** Send a broadcast message to a group of users.
Workflow object	No	Limits use of this program to policies associated with this workflow policy object.
Inactive	No	Checked if program is not active.
Comments	No	Add text to describe the program.

About the DB Operation Type

Often, when some data changes, other data must be changed in accordance. This data dependency is frequently implemented at the Object Manager layer through business component definitions and run-time events. Database operations by workflow policies provide an alternative that works at the database record level.

NOTE: It is recommended that DB Operation be used only when the data dependency involved centers around database records rather than around business components. For example, use DB operations when one database record must always be updated if another database record is inserted, regardless of which business components the two database records belong to, or whether the two database records belong to a business component at all.

Properties of the Workflow Policy Program Argument

A workflow policy program argument defines recipients, database actions, and available substitutions. Each workflow policy program typically has several program arguments. The argument fields that display in this view depend on the type of workflow policy program you choose. A workflow policy program argument is a child of a workflow policy program.

CAUTION: Certain text is not allowed in the Default Value workflow policy program argument. Do not use trailing spaces, [newline], or escape characters. Using this text can cause problems, such as failure of Workflow Monitor Agent during run time.

Table 125 describes properties displayed in the Workflow Policy Program Arguments OBLE.

Table 125. Description of Properties of the Workflow Policy Program Argument

Property	Required	Usage
Applet	Optional	Define the Picklist applet.
Default Value	Optional	Define the text value of a type that depends on the Name of the program argument-an SQL statement, the text of a message, the email address of a recipient, and so forth. Maximum length is 2000 characters. When setting this property, be sure to note the caution information described in this topic.
Name	Required	Define the parameter from a predefined set of values. For a description of values, see "Workflow Policy Program Argument Values for the Name Property" on page 452. Note that the value is entered manually. You do not choose from a picklist.
Picklist	Optional	Define the Picklist object.
Required	Boolean	Specify whether or not data entry is required. Value is TRUE or FALSE.
Source Field	Optional	Specify the Picklist Source field.
Visible	Boolean	Specify whether the data supplied by this argument is displayed. Value is TRUE or FALSE.
Inactive	(This cell is intentionally empty.)	Checked if program is not active.

Date Formats with Workflow Policy Programs

Formats to use when setting a Default Value for time/date fields include:

- Date Column format: 2001-03-16
- Time Column format: 19:26:26
- Date Time Column format: 2001-04-05 21:25:00

Note that a workflow policy program is executed by the Siebel Server that connects to Oracle through ODBC. As part of this process, the required data is retrieved from the database through this connection. The format of a date in an email is the format returned by the ODBC driver, which might be different to that used by Oracle.

Workflow Policy Program Argument Values for the Name Property

You can add functionality to a workflow policy program by creating a new workflow policy program argument. A *workflow policy program argument* determines how the workflow policy program behaves, including what substitutions are available for a workflow policy program and how the recipients are defined.

TIP: To familiarize yourself with how workflow Policy Program Arguments are used, examine some of the predefined object definitions. For example, in the Workflow Policy Programs OBLE, query the Name property for Run External Program. Note that this object definition references some of the common workflow policy program arguments described in Table 126. Next, query the Name property for Send Email. Note that this Workflow Policy Program references several Workflow Policy Program Arguments specific to sending a message, as described in Table 127.

Table 126 describes valid values for workflow policy program arguments common to workflow policy programs.

Table 126. Description of Arguments in the Name Property of the Workflow Policy Program Arguments OBLE

Common Argument	Usage	Allowable Default Value
Primary Id	The row ID of the violating row that the Workflow policy program is acting on.	Empty.
Primary Table	Define the base table to which the action is applied. The base table can be unrelated to the record of the primary ID. For example, the violating row is in a child table and you now need to insert or update a record in the parent table. Tables can also be updated that are not related to the primary ID table. For example, create a Broadcast Message record when a certain monitored condition in the Opportunity record is true.	Tables defined within the Siebel business object repository, as compared to the workflow business object. Workflow business objects are used for monitoring conditions but are not used in the coding of action programs.
Update Row ID	Define the row ID of a table other than the primary table of the workflow policy object. You can associate a workflow policy action with a workflow policy that updates a table. This value is used only when the Operation Type is set to update.	The row ID you need to update.
Operation Type	Define the operation to perform: update or insert.	Two possible values for DB Operation: Update or Insert.

Table 126. Description of Arguments in the Name Property of the Workflow Policy Program Arguments OBLE

Common Argument	Usage	Allowable Default Value
Field Name	Define the Name of the column in the base table to which the operation is performed. This is one of two field column pairs.	Allowable values: Text, Variable, Function.
New Row ID	For insert operations, this argument is automatically populated with the row ID of the row about to be inserted.	Empty.
Field Name (Column)	Define the Field Name, which must be identical to the Field Name of the first column pair and (Column) appended to the name. This is the second of two Column Pairs.	Actual field name in the base table. The value can not contain leading spaces.
Sql Statement	Used as a way to identify more data from the database to be used as substitutions when the action is performed.	Valid SQL query statement for the RDBMS used: Oracle, MS SQL, Informix, or Sybase.
Sql Statement Inputs	Define the Name of the column in the base table on which the operation is performed.	(This cell is intentionally empty.)
Sql Statement Outputs	Use as a placeholder for the values chosen in the SQL Statement argument.	Variable Name.

Considerations to weigh include:

- The value is entered manually into the Name property of the Workflow Policy Program Arguments OBLE.
- When you run a database operation with Insert as the Operation Type, you can choose a Default Value, New Row ID, as described previously, which provides the value for the ROW_Id field for the row being inserted.

Passing Multiple Variables into SQL Program Arguments

Multiple variables can be passed into SQL program arguments.

To pass multiple variables into SQL program arguments

1. In the Workflow Policy Program Arguments OBLE, choose Sql Statement Inputs.
2. Enter the variable names into the Default Value property.

   Use the following format: [variable1], [variable2], [variable n]

Send Message Program Arguments

Table 127 describes program arguments specific to the Send Message program.

Table 127. Description of Properties of the Send Message Program Argument

Value in Name Property	Usage	Value
Email Message	Define the body of the email message.	Text with available substitutions.
Email Message Repeated	Define the text that is repeated when the Consolidate feature is used.	Text with available substitutions.
Email Subject	Define the text in subject line of the email message.	Any text.
Send to Contact	Define the contacts available in Siebel.	(This cell is intentionally empty.)
Send to Position	Define the list of the positions available in Siebel.	(This cell is intentionally empty.)
Send to Employee	Define the list of employees available in Siebel.	(This cell is intentionally empty.)

Send Page Program Arguments

Table 128 describes program arguments specific to the Send Page program,

Table 128. Description of Properties of the Send Page Program Argument

Value in Name Property	Usage	Value
Send to Contact	Define the contacts available in Siebel.	Picklist of contacts.
Send to Employee	Define the list of employees available in Siebel.	Picklist of employees.
Send to Position	Define the list of the positions available in Siebel.	Picklist of positions.
Send to Relative	Specify to send to an individual or group of individuals related to the Workflow object.	(This cell is intentionally empty.)
Alpha Numeric Page Message	Define the body of the text message.	Text with available substitutions.
Numeric Page Message	Define the body of the numeric message.	Text with available substitutions.

Run External Program Arguments

Table 129 describes program arguments specific to the Run External Program.

Table 129. Description of Properties of the Run External Program Argument

Value in Name Property	Usage	Value
Command Line	Define the parameters to pass to the executable.	(This cell is intentionally empty.)
Executable Name	Define the full path to the executable to execute.	(This cell is intentionally empty.)
Executable Type	Define the mode in which the Workflow Action Agent executes the external program.	Wait. No wait.

Index

D

E

F

G

I

R

S