SERENA

SERENA®

BUSINESS MANAGER

SBM Composer Guide (On-Premise Version)

Serena Proprietary and Confidential Information

Trademarks

U.S. Government Rights

Part number: Product version: 2009 R4.02

Publication date: 2011-06-21

Table of Contents

Chapter 1: What's New in SBM Composer?

The following SBM Composer features and changes are new in SBM Composer.

- *Sub-Relational* fields can now be used to create rule expressions based on fields in different tables, applications, or process apps. The rule expressions are evaluated and used to determine how items are routed in application workflows.
- The **Design** tab of the Ribbon has a new application workflow **View Mode** section with the following three modes:
  - **Presentation** shows icons on states and transitions that represent design elements that have relationships with the states and transitions. You can click or double-click an icon to view or edit the properties of the design element, and hover over an icon for a form or orchestration workflow to see a thumbnail image of the form or workflow.
  - **Relationships** shows the Relationships bar, which contains visual cues that instantly demonstrate the relationship between the application workflow and the design elements it references. You can open the editor and Property Editor for a design element directly from the Relationships bar and make changes without having to navigate to them from App Explorer.
  - **Properties** shows icons on states and transitions to indicate that certain properties are set. You can click or double-click an icon to view or edit the corresponding properties in the Property Editor.
- The new Embedded Report widget allows you to embed an SBM report in a custom form. If you obtain a report reference name and URL from the SBM User Workspace and then use the URL to configure the widget in SBM Composer, the report will still work after you promote the process app containing the report to other environments.

  Note: Because the Embedded Report widget allows portability, it is recommended that you use it instead of the Web Page widget to embed reports in custom forms.
- You can now lock an event defininition so you can continue to use it in an asynchronous orchestration workflow that is called by an application in another process app. When an event definition is locked, the application, state, and transition names that comprise the event type and the names of the extension fields that are sent with the event do not change, even if these names are changed in the original process app.
- Outgoing transitions from a decision to an **Any** state are now supported. The decision must have one incoming transition from the **Any** state before it can have an outgoing transition to it.
- You can now locate specific text in the AppScript or JavaScript editor, or locate the text and replace it with something else.
- The RSS widget is no longer supported, and was removed from the **Form Palette**.

Part 1: Using SBM Composer

This section contains the following topics:

Chapter 2: Introducing SBM Composer

SBM Composer is the design component of SBM 2009. Use SBM Composer to design the structure of a process app and its constituent applications. This includes defining the tables, workflows, roles, *overrides* [page 690], and forms in the application. Multiple applications can be combined in each process app that you define in SBM Composer.

In addition, process apps defined in SBM Composer can include orchestrations, which you use to coordinate Web service-enabled systems. *Web services* [page 701] extend the standard behavior of applications to integrate with other systems in your environment. For example, you could define an orchestration that calls a Web service to update an in-house requirements management system, portal, or third-party component in response to the creation of an item in the SBM User Workspace. SBM Composer orchestration workflows coordinate Web services using a wide variety of steps that control flow and manipulate data. Such steps include **Decision**, **While**, **ForEach**, **Calculate**, **Throw**, **Scope**, and **Compensate**.

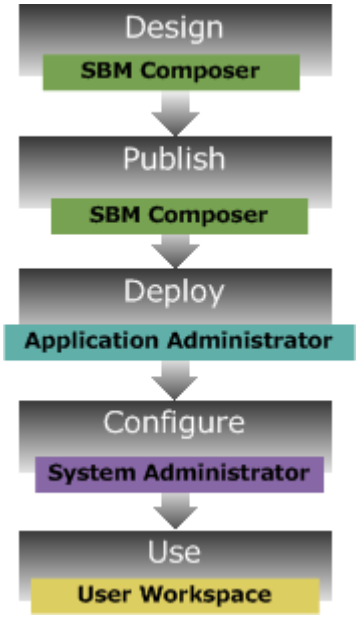
Multiple SBM Composer designers can collaborate to create or modify designs that are independent of the runtime environment. The designed process apps can then be deployed to the SBM User Workspace using either SBM Composer or Application Administrator.

This section contains the following topics:

About the SBM Components

Serena® Business Manager includes components from which you create and manage process apps.

The following figure illustrates the stages of creating and using a process app in SBM.



The following table describes the components of SBM.

Component	Description
SBM Composer	The component in which you design process apps, applications, orchestrations, and all of the design elements they comprise. In SBM Composer, you can work on a process app locally, without affecting the runtime environment. You can check process apps, applications, orchestrations, or individual design elements in and out of the SBM Application Repository, from which process apps can be deployed to a runtime environment. You can also deploy tested process apps directly from SBM Composer.
SBM Application Administrator	SBM Composer exports process app blueprints to Application Administrator through the SBM Application Repository. Versions of blueprints and individual design elements created in SBM Composer are stored in the SBM Application Repository. The SBM administrator uses Application Administrator or SBM Composer to deploy processes contained in the *application* [page 679] to the appropriate environments, such as the SBM Application Engine and the SBM Orchestration Engine.
SBM System Administrator	The component that SBM administrators use to create projects, assign users and groups to roles, set field *overrides* [page 690] for projects, and create *notifications* [page 689]. User and group accounts are also managed in this component.
SBM Application Engine	The component that processes applications created in SBM Composer.

Component	Description
SBM User Workspace	Users use the SBM User Workspace to create and track items (such as issues) that are related to projects. The projects are associated with workflows designed in SBM Composer. Forms, tables, fields, states, transitions, and other elements that users work with in the SBM User Workspace are also designed in SBM Composer.
SBM Orchestration Engine	The component that processes orchestrations that were created in SBM Composer. Orchestrations can specify that a Web service flow start in response to certain events in SBM.

Using SBM Composer and SBM System Administrator

SBM has two clients, SBM Composer and SBM System Administrator, that you use to create and manage applications and orchestrations.

In general, these two components are used as follows:

- **SBM Composer** is used to design process apps, applications, and orchestrations. This includes designing states, transitions, forms, actions, and other elements. You also use SBM Composer to create roles and tie the roles to particular states or transitions.
- **SBM System Administrator** is used to change projects, including adding, removing, or modifying projects and project overrides. You also use SBM System Administrator to add user and group accounts, assign users to specific roles, and create notifications.

The following sections describe the tasks you perform in each component.

Project Management

The following table describes the project management tasks that are performed from each client:

Task	Tool
Creating and editing projects	SBM System Administrator

Workflow, State, and Transition Management

The following table describes which workflow, state, and transition management tasks are performed from each client:

Task	Tool
Creating a workflow (application or orchestration)	SBM Composer
Adding states to a workflow	SBM Composer
Adding transitions to a workflow	SBM Composer

Task	Tool
Adding, deleting, or modifying actions for a state or transition (including scripts, triggers, events, Web services, and transitions)	SBM Composer
Restricting transitions by type	SBM Composer
Creating forms	SBM Composer
Associating privileges with a form (using roles)	SBM Composer

Table and Field Management

The following table describes which table and field management tasks are performed from each client:

Task	Tool
Adding a primary or auxiliary (non-system) table	SBM Composer
Adding system auxiliary tables (created when you create a database with Create Database wizard)	SBM System Administrator
Adding fields to a table	SBM Composer
Modifying field properties	SBM Composer
Modifying field overrides in a workflow, state, or transition	SBM Composer
Setting default values for a field	SBM Composer
Setting general field overrides	SBM Composer
Setting field overrides for specific projects or user fields	SBM System Administrator
Adding roles to a user field	SBM Composer
Adding groups or users to a user field	SBM System Administrator
Allowing data import for primary or auxiliary tables	SBM System Administrator

User, Group, and Role Management

The following table describes which user, group, and role management tasks are performed from each client:

Task	Tool
Creating roles	SBM Composer
Creating users and groups	SBM System Administrator
Assigning permissions to roles	SBM Composer
Assigning privileges to users	SBM System Administrator
Assigning privileges to groups	SBM System Administrator
Assigning users to groups	SBM System Administrator
Assigning users and groups to roles	SBM System Administrator
Modifying privileges for specific project or field	SBM System Administrator

About SBM Composer

SBM Composer is the SBM component that you use to create applications. Applications are composed of business processes that address specific business requirements. For example, applications can provide processes that define the transfer of responsibilities to various members of an organization as primary items are moved through a completion process. Applications can be implemented in various supported processing environments. Examples of supported processing environments are the SBM Application Engine, which is used to track items submitted to SBM projects, and the SBM Orchestration Engine, which is used to orchestrate *Web services* [page 701] between business applications.

Applications and orchestrations are composed of design elements that collectively define the processes that run in the runtime environments. SBM Composer is used to define each of the elements in the *application* [page 679] or *orchestration* [page 690]. For example, every custom form and every workflow are design elements. Each *design element* [page 683] is part of the greater process app and can be versioned independently from the rest of the process app. See Chapter 7: Introducing Applications [page 101] and Part 4: Working with Orchestrations [page 429] for more information.

SBM Composer provides an intuitive visual interface that lets you create and edit the design elements that make up an application or orchestration. App Explorer lets you select and open instances of design elements or create new ones. The quick access toolbar makes commonly used commands readily available, no matter what you are working on. The Ribbon displays design tabs with options and commands that vary according to your focus in the application design. Along with the main menu and context (right-click) menus, SBM Composer controls and commands are available as you need them, depending on the design task in which you are engaged.

SBM Composer lets you collaborate with other designers on the development of design elements by using its check-in, check-out, and limited versioning capabilities.

You *publish* [page 692] process apps from SBM Composer. Publishing transfers process app blueprints to Application Administrator, from which they can be deployed to such target environments as the SBM Application Engine and SBM Orchestration Engine.

In an example scenario, a business analyst of a company uses SBM Composer to describe a process app that defines a new employee processing business scenario. The business analyst uses SBM Composer to model the business process that will be used to track the completion of tasks associated with processing new employees. A process could be defined for Human Resources to track tasks for adding new employees to the payroll. A different team process could be defined for IT to track tasks for setting up phone and network access for new employees.

SBM Composer also supports the creation of *BPEL* [page 680]-based orchestrations. In a typical scenario, the business analyst of a company uses the graphical workflow designer in SBM Composer to define a purchase order business scenario. Web services needed for this scenario are included in the business process flow. After the business analyst defines the business process flow, SBM Composer generates an orchestration that defines the process logic. At runtime, the *BPEL engine* [page 680] executes the *orchestration workflow* [page 690].

About the SBM Application Repository

SBM includes a repository for the design elements that you create in SBM Composer. While it is not intended to be a full-featured source control manager, it does support basic features, such as limited versioning and patching, that enable collaborative development. After you create a process app in SBM Composer, you can check it in to the SBM Application Repository, where other designers can open it and check out design elements they need to modify.

The SBM Application Repository operations that are available when you right-click a *design element* [page 683] in App Explorer are **Check In**, **Get Latest**, **Check Out**, **Undo Check Out**, **Refresh Status**, **Version History**, and **Break Lock**.

Working Offline or Connected

Use the control near the bottom right corner of the SBM Composer window to toggle the connection to the configured SBM Application Repository. Your connection status shows you as either **Offline** or **Connected to *ServerName* (*ServerType*)** (where *ServerType* indicates whether you are connected to an on-demand or on-premise server).

The ability to toggle this setting makes it possible to work on your process app without a network connection. However, if you work offline, you cannot check design elements in or out, open process apps in the SBM Application Repository, or *publish* [page 692] and deploy process apps. (See Open Process App Dialog Box [page 638] for details.)

If you switch to working offline while a process app is open, SBM Composer gives you the opportunity to download any design elements that are not yet saved to the *Local Cache* [page 688].

Switching Repositories

Each process app you check in to a SBM Application Repository is associated with that repository. For example, the process app only appears in the **Open Process App** dialog box if that repository is specified in the SBM Composer **Options** dialog box. This is true even if you are working offline. For your convenience, SBM Composer retains the connection information you specify for multiple repositories, so you can switch among

them as needed. See Repository Tab of the SBM Composer Options Dialog Box [page 46] for details.

Local Cache

This special area on the file system of your computer is where SBM Composer stores design elements while you work on them, both before you check them in to the SBM Application Repository and any time they are checked out. Initially, SBM Composer creates the Local Cache in the directory allocated by Windows for the Windows login ID that was in effect when you installed SBM Composer. If you need to specify a different location, you can do so on the Repository Tab of the SBM Composer Options Dialog Box [page 46].

About Publishing, Deployment, and Promotion

The process app *blueprint file* [page 681] is the basic unit of development in SBM and represents a complete design of one or more applications, orchestrations, or both. Process app blueprints are created in SBM Composer, published from SBM Composer to Application Administrator, and deployed to the SBM Application Engine and the SBM Orchestration Engine. Process app snapshots contain both the blueprint information and configuration data, and are promoted from Application Administrator to different environments.

Publishing

Publishing is the act of packaging a process app that you created in SBM Composer and making it available in Application Administrator. Publishing makes a process app available for *deployment* [page 683]. You can *publish* [page 692] a process app in one of the following ways:

- In SBM Composer, publish the process app to Application Administrator.
- In SBM Composer, export the process app to a file. Then, in Application Administrator, load the file.

Deployment

Deployment is the act of taking a process app defined in SBM Composer and making it available on the SBM Server. Before you can deploy a process app, it must be published. The publish operation is performed automatically if you use the **Deploy** command, but if you do not have permission to deploy, but do have permission to publish, you must explicitly publish the process app.

You can deploy process apps from Application Administrator or, if an administrator set up an *environment* [page 684] in Application Administrator that supports it, you can deploy a process app from SBM Composer. For more information on deployment, see the *SBM Application Administrator Guide*.

After a process app that contains an *application* [page 679] is deployed, use SBM System Administrator to create projects, add project-level *overrides* [page 690], and assign users and groups to the roles defined in SBM Composer.

Note: For more information about deploying from SBM Composer, see Deploying a Process App [page 84].

Promotion

Promotion is the process by which administrators use Application Administrator to replicate a process app from one environment to another. After deployment and testing, for example, a process app in the test environment can be replicated to the production environment for real work. When you promote, you transfer both the process app design (created in SBM Composer, and known as a blueprint file), and the configuration data such as users, groups, projects, and reports (set up in SBM System Administrator or the Web Administrator) to another environment. As noted earlier, the combination of the design and configuration data is called a process app snapshot.

Note: For more information about promotion and process app snapshots, see the *SBM Application Administrator Guide*.

About Version Control

Version control, also known as revision control or source control, is the management of multiple revisions of the same file. The core purpose of version control is to enable the collaborative editing and sharing of data. Changes to a file are identified by an associated code, called the revision ID, typically a number that is incremented to track the changes. For example, a simple form of revision control might have the initial version of a file assigned the revision number 1. When the first change is made, the revision number is incremented to 2, and so on.

All revisions of all files are held in a central archive. When designers need to work on a file, they check it out of the archive, creating a new copy of the file in a local working location. They update the local copy of the file, and then check it back in to the archive, where it is available to other designers.

If designers decide not to check in their changes, they can undo the check-out. If they simply want to see the file, rather than change it, they can get a copy of the latest version.

Most version control systems include a mechanism by which you can view the history of files in the archive. The history shows who made the changes and when they made them.

An example scenario follows:

1. Dana and Andy both need to make changes to the same file, `index.html`. The current revision of the file in the archive is 1.

2. Dana checks out `index.html` from the archive. This creates a placeholder, revision 2, in the archive for a new revision of `index.html` and copies the file to her local working area. A lock is placed on the current revision of the file in the archive, so that no one else can check it out and change it. Dana modifies `index.html` and checks it back in to the archive. The lock is released. The current revision of `index.html` in the archive is 2.

3. Andy checks out `index.html` from the archive. This creates a placeholder (revision 3) in the archive, copies `index.html` to his local working area, and locks the current revision in the archive. Andy starts to edit `index.html` but then receives an e-mail message that informs him that the change is no longer required. Andy does an *undo check out* [page 700] operation that discards any changes he made to the file since he checked it out, leaving the file checked in, unlocked, and unchanged in the

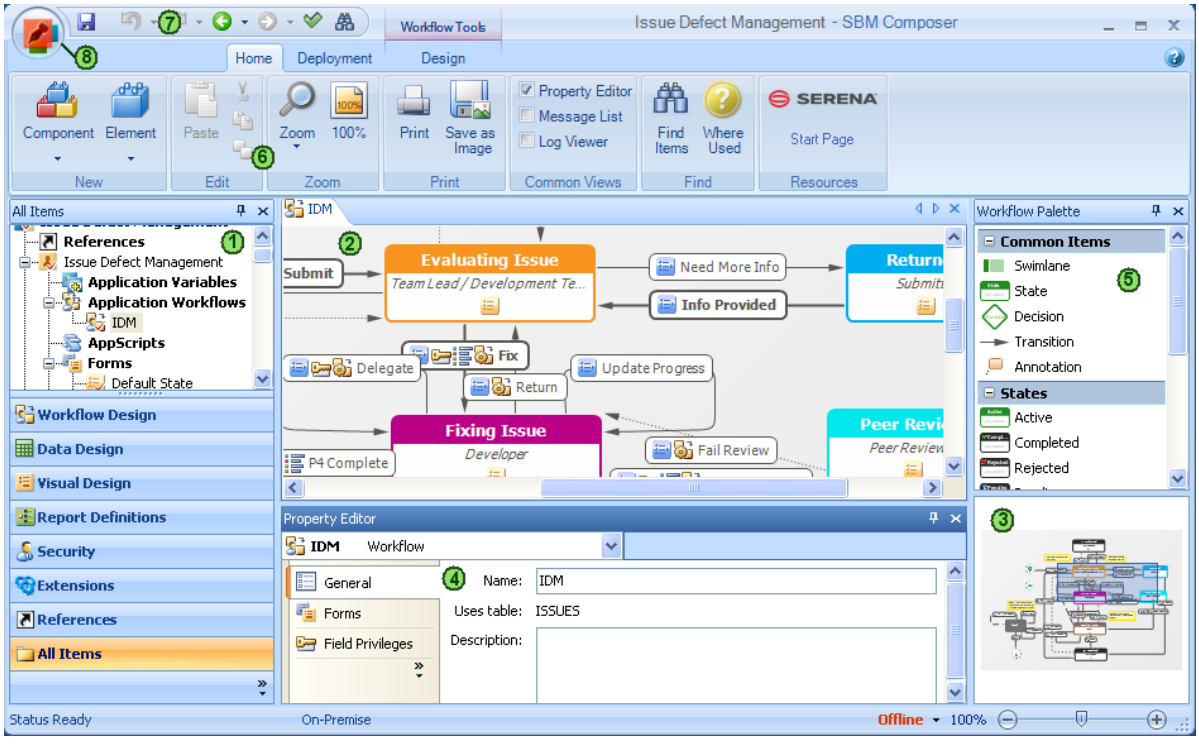
archive. The revision number that was created during the check-out is discarded, so the current revision of `index.html` in the archive is 2.

In Serena Business Manager, the archive is the SBM Application Repository. The archived data contains the process apps and the design elements they include. The **Get Latest of All**, **Check Out All**, **Check In All**, **Undo Check Out All**, and **Process App Version History** commands are all available from the Composer menu.

Chapter 3: SBM Composer Interface

This section introduces you to the main features of the SBM Composer user interface.

The following figure shows the main interface as it looks with a checked-out process app open. The following legend describes interface elements identified in the figure.



1. App Explorer [page 63]
2. Editor [page 63]
3. Zoom Preview [page 64]
4. Property Editor [page 65]
5. Palette [page 64]
6. Ribbon [page 53]
7. Quick Access Tool Bar [page 52]
8. Composer Button and Menu [page 39]

Composer Button and Menu

When you click the Composer button at the top left of the SBM Composer window, the commands in the following table are available from the Composer menu.

Command	Description
New	Opens the **Create New Process App** dialog box, in which you create a process app based on a process app *template* [page 699]. The templates are grouped into the following categories: • Process apps that are stored in AppCentral™ • Process apps with the basic elements of an *application* [page 679] or orchestration • Empty process apps • Templates that are stored in the file system of your computer. For detailed information about this dialog box, see Create New Process App Dialog Box [page 626].
Open	Opens the **Open Process App** dialog box, in which you can open a process app that is stored on your computer (the *Local Cache* [page 688]) or a process app that was checked into the SBM Application Repository. For detailed information about this dialog box, see Open Process App Dialog Box [page 638].
Import and Export | Import from File	Opens the **Import Process App Blueprint** dialog box, in which you can find a process app blueprint previously exported from SBM Composer or saved from Application Administrator. You can also find a process app template previously exported from SBM Composer. After you import a blueprint or template, use the **Open** command to work with the process app. For detailed information about this dialog box, see Import Process App Blueprint Dialog Box [page 635].
Import and Export | Export to File	Validates the process app that is open, and opens the **Save As** dialog box, in which you navigate to the directory where you want to store the process app. The export feature lets you send process app blueprints to others who do not have access to your SBM Application Repository. It also lets you export a process app and send the *blueprint file* [page 681] to someone else to load into Application Administrator, effectively publishing the process app. For detailed information about this feature, see Exporting a Process App [page 83].

Command	Description
Import and Export | Upgrade Snapshot	You cannot promote a process app *snapshot* [page 697] if it was created in an earlier version of SBM Composer. SBM Composer has a feature that upgrades older process app snapshots to the latest version. This command opens the **Upgrade Process App Snapshot** dialog box, which lets you do this. After the process app is upgraded and saved to a file, the upgraded snapshot file must be loaded into Application Administrator (using the **Load from File** command) and then promoted from Application Administrator into the SBM Server (using the **Promote** command). For detailed information about this feature, see Upgrading a Snapshot [page 87].
Compare | With Local File	Opens the **Compare Process App** dialog box, in which you select a process app stored on the file system of your computer to compare to the open process app. Use the comparison to determine the modifications that would be required to change from the open process app to the process app you selected for comparison. For detailed information about this feature, see Comparing Process Apps [page 81].
Compare | With Published Process App	Opens the **Select Published Process App** dialog box, in which you select a process app stored in the SBM Application Repository to compare to the open process app. Use the comparison to determine the modifications that would be required to change from the open process app to the process app you selected for comparison. For detailed information about this feature, see Comparing Process Apps [page 81].
Save	Saves a process app to the Local Cache. This lets you save your work without having to check the process app in to the SBM Application Repository. For detailed information about this feature, see Saving a Process App [page 82].
SBM Application Repository | Get Latest of All	Opens the **Get Latest Design Elements** dialog box, with all design elements selected. Use this dialog box to get the latest version of the design elements from the SBM Application Repository, overwriting any changes you made since checking in the design elements. For detailed information, see About Version Control [page 36] and Get Latest Design Elements Dialog Box [page 635].

Command	Description
SBM Application Repository | Check Out All	Opens the **Check Out Design Elements** dialog box, with all design elements selected. Design elements must be checked out of the SBM Application Repository before you can edit them. For detailed information, see About Version Control [page 36] and Check Out Design Elements Dialog Box [page 623].
SBM Application Repository | Check In All	Opens the **Check In Design Elements** dialog box, with all design elements selected. All design elements must be checked in to the SBM Application Repository before you can *publish* [page 692] a process app. For detailed information, see About Version Control [page 36] and Check In Design Elements Dialog Box [page 622].
SBM Application Repository | Undo Check Out All	Opens the **Undo Check Out for Design Elements** dialog box, with all design elements selected. Use this dialog box to discard any changes you made to the selected design elements since you checked them out, and leave the elements checked in and unchanged in the SBM Application Repository. For detailed information, see About Version Control [page 36] and Undo Check Out Design Elements Dialog Box [page 644].
SBM Application Repository | Process App Version History	Opens the **Version History** dialog box. Use this dialog box to see the history of versions of the process app. For detailed information, see About Version Control [page 36] and Version History Dialog Box [page 644].
Validate	Validates the open process app. A process app must be validated before it can be published.
Publish	Opens the **Publish Process App** dialog box, in which you identify the version of the process app and specify whether the process app can be deployed by others. A process app must be published before it can be deployed.  **Note:** Process apps are automatically published as part of the *deployment* [page 683] process. Use the **Publish** command if you have privileges to publish a process app, but do not have privileges to deploy a process app. For detailed information, see Publishing a Process App [page 83] and Publish Process App Dialog Box [page 640].

Command	Description
Deploy | Deploy	Opens the **Deploy Process App** dialog box, in which you select the *environment* [page 684] to which you want to deploy, identify the version of the process app, and specify whether the process app can be deployed by others. You access the **Deploy Options** dialog box from the **Deploy Process App** dialog box. The **Deploy Options** dialog box lets you configure e-mail notifications that are sent when a process app is deployed, and map endpoints. For detailed information, see Deploying a Process App [page 84], Deploy Process App Dialog Box [page 630], and Deploy Options Dialog Box [page 630].
Deploy | Quick Deploy	Lets you deploy a process app without being prompted to provide information in the **Deploy Process App** dialog box. For detailed information, see Deploying a Process App [page 84].
Delete Process App | From Local Cache Only	Lets you delete the open process app from the Local Cache.
Delete Process App | From Local Cache and Remote Repository	Lets you delete the open process app from both the Local Cache and the SBM Application Repository. Deleting a process app from the SBM Application Repository does not affect any environments to which it has been deployed. For detailed information, see Deleting a Process App [page 87].
Close	Closes the open process app.
SBM Composer Options	Opens the **SBM Composer Options** dialog box. For detailed information, see Options Dialog Box [page 43].

Options Dialog Box

Use this dialog box to set SBM Composer options. Open the dialog box by clicking **SBM Composer Options** at the bottom of the Composer menu.

Click a link on the left side of the dialog box to view and change the corresponding settings. Click **Resources** to gain access to other users, get information about the software and your computer system, and view the SBM Composer log file.

Tip: You can use shortcut keys instead of the mouse to switch links in the dialog box. For details, see Options Dialog Box [page 43].

General Tab of the SBM Composer Options Dialog Box

Element	Description
Automatically open most recent process app	By default, SBM Composer opens the Start Page when you start a new session. The Start Page includes links to downloadable process apps, training, support, and the community of designers. If you select this control, the process app you were working on when you closed your last SBM Composer session opens instead of the Start Page.
Auto-save process app every *n* minutes	Specify whether and how often to create a record of your changes to the open process app so that it can be recovered after a power failure or other inadvertent termination of SBM Composer. When this check box is selected, SBM Composer saves your process app automatically at the specified interval.
Mode	Using SBM Composer in on-premise mode exposes certain features that are incompatible with *deployment* [page 683] in an on-demand environment ("the cloud"). Unless you need to work on a process app intended for deployment to the cloud, make sure that the **On-Premise** mode is selected.  **Tip:** Because the operating mode affects (among other things) how design elements are validated, you cannot change this setting while a process app is open. Close the process app, change the setting, and then open the process app again.
Include validation for On-Demand compatibility	When the **On-Premise** mode is selected, you can also choose whether SBM Composer should warn you when validation detects any features in your process app that are incompatible with on-demand *deployment* [page 683]. This helps you confirm that a process app you are designing (while in on-premise mode) can eventually be deployed to the cloud.
Synchronize App Explorer with current editor	Specify whether SBM Composer should update App Explorer to select the *design element* [page 683] that is open in the main editor pane.

Element	Description
Auto open item when switching filters in App Explorer	Specify whether SBM Composer should expand headings in App Explorer when you click on the corresponding filter buttons at the bottom of App Explorer.
Highlight validation failures in App Explorer	Select this check box and click the color picker to change the way validation failures are highlighted.
Open recently checked-out item in an editor	Specify whether, on check out of design elements from App Explorer, SBM Composer automatically opens one of them.
Provide introductory warnings	SBM Composer typically offers informational messages when you perform certain operations. For example, when you save a process app, SBM Composer reminds you that your changes were not checked into the SBM Application Repository. Clear this check box to avoid seeing these messages.  **Note:** You can dismiss an individual category of messages by selecting the **Do not show me this message again** check box in any message box where it appears.
Warn on data constraint violations	Specify whether SBM Composer warns you when you enter a default value (on the **Data Mapping** tab of the Property Editor) that does not conform to the requirements of the specified data type of the item (such as a default value of 0.5 for an integer field).
Select All Items filter on opening process app	Specify whether SBM Composer shows all design elements in App Explorer when you open an existing process app, or when you create a new process app from the **Create New Process App** dialog box. If you do not select this check box, the **Workflow Design** filter is displayed by default. For more information, see Create New Process App Dialog Box [page 626].
Warn about mode mismatches with the repository	Specify whether SBM Composer should warn you when you open an on-premise process app while connected to an on-demand repository, and provide the opportunity to switch to on-premise mode. If you switch to on-premise mode, and if the process app contains on-premise features, the process app cannot be exported or published. This is because the on-demand feature set is a subset of the on-premise feature set.

Repository Tab of the SBM Composer Options Dialog Box

Element	Description
Behavior	Set preferences for warnings about common actions: • Warn when deleting checked out design elements • Warn when closing a process app with new design elements • Warn when closing a process app with checked out design elements • Don't show **Check Out** dialog box for automatic check-out
Local Cache Path	Specify a different location for the *Local Cache* [page 688], or reset it to the default location. You could specify a different location if, for example, the drive containing the default location (typically the C: drive) is low on storage space.  **Tip:** You cannot change the location of the Local Cache while a process app is open.
Remote Repository Connection Settings	Specify whether you want to work online or offline. If you select **Work Online**, enter the computer name and port for the SBM Server, and the user name and password required to access it. The default installation port is 8085.  **Note:** Your current connection information, including whether you are connected or offline, is displayed at the top of the **Connection Information** element. Your current connection information, as well as the server mode (on-premise or on-demand), is also displayed at the bottom right of the window.  **Tip:** You cannot connect to a different SBM Application Repository while a process app is open.
Use Secure Connection	Specify whether a Secure Socket Layer (SSL) connection to the SBM Application Repository should be used.
Test Connection	Confirm that the SBM Application Repository can be reached with the information you provided.

Log Viewer Tab of the SBM Composer Options Dialog Box

Element	Description
Delete local copies... after *N* days	When you select the Details tab of the log viewer [page 419] (or click the **Download messages** button on that tab), SBM Composer copies log messages for the workflow you are viewing (or for the entire process app) to your computer, so you can still see them when you work offline. To keep your computer from filling up with these copied messages, SBM Composer automatically deletes any local copies that have been stored for longer than what you specify here.
Clear local message cache now	Delete the local copies of all the log messages currently stored on your computer. Message caching resumes immediately.
Maximum message count displayed in Log Viewer	The maximum number of messages to be displayed on the **Details** tab of the Log Viewer. The maximum is 20000.
Message retrieval batch size	Messages are retrieved from the server in batches of the size you specify until all messages are retrieved. The maximum is 500.
Defaults	Restore the values of the **Maximum message count displayed in Log Viewer** and **Message retrieval batch size** fields to 1000 and 50, respectively.

Application Tab of the SBM Composer Options Dialog Box

The **Application** tab itself has no content. It has four sub-tabs that are described below.

Form Tab

Element	Description
Editor	Specify preferences for behavior in the form editor: • Show insertion point during drag-and-drop operation even when invalid • Show cell handles when control is selected
New form Size Guide default settings	Specify default settings for the size of new forms. The values you specify are displayed in the Property Editor for the new form.

HTML and Script Tab

Element	Description
Font	Specify the name and size of the font used in the Javascript editor and Script editor, and in the **Content** tab of some form widgets (for example, the HTML/JavaScript and Google Gadget widgets).
Syntax Highlighting	The current color and font properties of the syntax elements are shown in the **Preview** box. If these values have never been changed, or if default values were restored, the default color and font properties are shown. • To change the syntax highlighting, select a syntax element and adjust the color and font properties. The changes you make are immediately reflected in the **Preview** box. • If you change your mind, click **Restore selected to default** to change the selected syntax element back to the default color and font properties, or click **Restore all to default** to change all syntax elements back to the default color and font properties.

Table Tab

Element	Description
Editor	Specify preferences for behavior of the table editor: • Immediately rename new fields
Warnings	Specify preferences for warnings related to tables: • Warn when allowing selection by group for *Multi-User* fields

Workflow Tab

Element	Description
Editor	Specify preferences for behavior of the workflow editor: • Immediately rename new states • Immediately rename new transitions

Event Tab

Element	Description
Warn when exporting unlocked event	Specify whether a warning should be displayed when you export an event definition that is not locked. An event definition must be locked if it is exported and then used by an asynchronous orchestration workflow in another process app.

Resources Tab of the SBM Composer Options Dialog Box

Element	Description
About	Displays the software version, and provides access to computer system information. This information could be requested by Serena Support.
SerenaCommunity	Provides access to Serena software users.
Log File	Displays the log of system messages, which could be requested by Serena Support.

Import and Export Menu

The following import and export commands are available from the Composer menu.

Import from File

The **Import from File** command opens the **Import Process App Blueprint** dialog box, in which you can find a process app blueprint that was previously exported from SBM Composer or saved from Application Administrator. (The process app blueprint could also be downloaded from the pre-built process app area of http://www.serena.com.) You can also find a process app *template* [page 699] that was previously exported from SBM Composer. After you import a blueprint or template, use the **Open** command to work with the process app. For more information about blueprints, see Import Process App Blueprint Dialog Box [page 635]. For more information about templates, see About Templates [page 77].

Export to File

The **Export to File** command validates the process app that is open, and opens the **Save As** dialog box, in which you navigate to the directory where you want to store the process app.

The export feature lets you send process app blueprints to others who do not have access to your SBM Application Repository, and lets you export a process app and send the *blueprint file* [page 681] to someone else to load into Application Administrator, effectively publishing the process app.

The export feature also lets you export a process app template. You can then browse for it in the **Create New Process App** dialog box. This makes it available as a template that

can be used as a starting point for creating other process apps. For more information about templates, see About Templates [page 77].

For more information about exporting a process app, see Exporting a Process App [page 83].

Upgrade Snapshot

If you attempt to load or promote a *snapshot* [page 697] created in an earlier version of SBM into Application Administrator, you receive a warning that the snapshot needs to be loaded and upgraded in SBM Composer. In SBM Composer, the **Upgrade Snapshot** command opens the **Upgrade Process App Snapshot** dialog box, which lets you upgrade the snapshot to the latest version.

For more information about upgrading snapshots, see Upgrading a Snapshot [page 87].

Compare Menu

The following comparison commands are available from the Composer menu.

With Local File

The **With Local File** command opens the **Compare Process App** dialog box, in which you select a process app stored in the file system of your computer to compare to the open process app. Use the comparison to determine the modifications that would be required to change from the open process app to the process app you selected for comparison.

With Published Process App

The **With Published Process App** command opens the **Select Published Process App** dialog box, in which you select a process app stored in the SBM Application Repository to compare to the open process app. Use the comparison to determine the modifications that would be required to change from the open process app to the process app you selected for comparison.

For more information about the comparison commands, see the following topics:

Comparing Process Apps [page 81]

Compare Process Apps Dialog Box [page 623]

Select Published Process App Dialog Box [page 643]

SBM Application Repository Menu

The following repository commands are available from the Composer menu.

Get Latest of All

The **Get Latest of All** command opens the **Get Latest Design Elements** dialog box, with all design elements selected. Use this dialog box to get the latest version of the design elements from the SBM Application Repository, overwriting any changes you made since checking in the design elements.

Check Out All

The **Check Out All** command opens the **Check Out Design Elements** dialog box, with all design elements selected. Design elements must be checked out of the SBM Application Repository before you can edit them.

Check In All

The **Check In All** command opens the **Check In Design Elements** dialog box, with all design elements selected. All design elements must be checked into the SBM Application Repository before a process app can be published.

Undo Check Out All

The **Undo Check Out All** command opens the **Undo Check Out for Design Elements** dialog box, with all design elements selected. Use this dialog box to discard any changes you made to the selected design elements since you checked them out, and leave the design elements checked in and unchanged in the SBM Application Repository.

Process App Version History

The **Process App Version History** command opens the **Version History** dialog box. Use this dialog box to see the history of versions of the process app.

For more information about the repository commands, see the following topics:

- About Version Control [page 36]
- Get Latest Design Elements Dialog Box [page 635]
- Check Out Design Elements Dialog Box [page 623]
- Check In Design Elements Dialog Box [page 622]
- Undo Check Out Design Elements Dialog Box [page 644]
- Version History Dialog Box [page 644]

Deploy Menu

The following *deployment* [page 683] commands are available from the Composer menu.

Deploy

The **Deploy** command opens the **Deploy Process App** dialog box, in which you select the *environment* [page 684] to which you want to deploy, identify the version of the process app, and specify whether the process app can be deployed by others.

You access the **Deploy Options** dialog box from the **Deploy Process App** dialog box. The **Deploy Options** dialog box lets you configure e-mail notifications that are sent when a process app is deployed, and map endpoints.

Quick Deploy

The **Quick Deploy** command lets you deploy a process app without being prompted to provide the information that is specified in the **Deploy Process App** dialog box. If this is the first time you are deploying a process app, it is recommended that you use the **Deploy** command, not the **Quick Deploy** command. The **Deploy** command lets you see the default settings and change them, if necessary.

The **Quick Deploy** command also lets you decide whether you want to create a new version of the process app and whether you want to leave design parts checked in after the deployment completes.

For more information about the deployment commands, see the following topics:

- Deploying a Process App [page 84]
- Using Quick Deploy [page 85]
- About Publishing, Deployment, and Promotion [page 35]
- Deploy Process App Dialog Box [page 630]
- Deploy Options Dialog Box [page 630]

Delete Menu

The following deletion commands are available from the Composer menu.

From Local Cache Only

The **From Local Cache Only** command lets you delete the open process app from the *Local Cache* [page 688].

From Local Cache and Remote Repository

The **From Local Cache and Remote Repository** command lets you delete the open process app from both the Local Cache and the SBM Application Repository. Deleting a process app from the SBM Application Repository does not affect any environments to which it has been deployed.

For more information about the deletion commands, see Deleting a Process App [page 87].

Quick Access Toolbar

The quick access toolbar, at the top of the SBM Composer window, contains commonly used commands.

Command	Description
Save locally	Saves the process app to the *Local Cache* [page 688] (on a local or network drive).
Undo	Returns the process app to its condition before the last operation (or multiple operations if you click the down arrow and select them).
Redo	Returns the process app to its condition after the last operation (or multiple operations if you click the down arrow and select them) that was undone. Each redo undoes the last undo.
Navigate Backward	Displays a previous view in the current editing session. You can browse the history of previous views by clicking the down arrow.
Navigate Forward	Displays a subsequent view in the current editing session. You can browse the history of subsequent views by clicking the down arrow. (This command is only available if you are viewing a previous view through the **Navigate Backward** command.)

Command	Description
Validate open Process App	Validates the open process app to determine whether it is ready to *publish* [page 692] to Application Administrator.
Publish open Process App	Publishes the open process app to Application Administrator.
Deploy open Process App	Deploys the open process app (if you have an *environment* [page 684] set up in Application Administrator to allow *deployment* [page 683] from SBM Composer).
Deploy open Process App using previous settings	Deploys the open process app (if you have an *environment* [page 684] set up in Application Administrator to allow deployment from SBM Composer) using settings previously specified in the **Deploy Process App** dialog box.
Find in open process app	Opens the **Find Items** dialog box to locate specific design elements in the open process app.

Ribbon

The Ribbon, near the top of the SBM Composer window, provides a central location for the commands you use to perform design tasks. The Ribbon contains commands appropriate to the *design element* [page 683] you are editing.

The commands in the Ribbon are grouped in tabs:

- Home [page 53]
- Design [page 58]
- Annotations [page 61]
- Script [page 62]
- Deployment Tab of the Ribbon [page 56]

Home Tab of the Ribbon

The **Home** tab is in the Ribbon when SBM Composer is first opened. It contains common commands that are available in all of the editors.

The following tables describe the commands available in the various areas on the **Home** tab of the Ribbon.

New

Command	Description
Component	Add a new *application* [page 679], *orchestration* [page 690], or application reference to the open process app.
Element	Add one of the following elements to the open process app: • State Form • Transition Form • Image • Icon • JavaScript • Listing • Role • Rule • AppScript • Table • Web Service • Workflow • Sub-workflow • Event with Reply

Edit

Command	Description
Paste	Place the content of the Windows Clipboard.
Cut	Move the selected item to the Clipboard.
Copy	Move a copy of the selected item to the Clipboard.
Duplicate	Move a copy of the selected item to the Clipboard, and place the content of the Clipboard.

Repository

Command	Description
Get Latest	Get a read-only copy of the selected design elements from the repository. This does not create a new version, and leaves the design elements available for check out.
Check Out	Retrieve an editable version of the selected design elements from the repository. This creates a new version, and locks the design elements so they cannot be checked out and changed by another user.
Check In	Store the selected design elements and any changes you made in the repository. This makes the new version available for check out.
Refresh Status	Update the repository status of the open process app. For example, checked out by you, checked out by another, and so on.
Undo Check Out	Cancel the check out of the selected design elements. This leaves them checked in, unchanged, and available for check out from the repository.
Version History	View a list of the versions of the selected design elements that were checked in to the repository.

Zoom

Command	Description
Zoom	Select **Fit to Window** or a percentage of the normal size to see more or less of the form or workflow in the editor pane.
100%	Display the form or workflow in the editor pane at its normal size.

Print

Command	Description
Print	Print the *design element* [page 683] in the editor pane.
Save As Image	Save an image of the form or workflow in the editor pane as PNG, JPEG, or other common image format.

Common Views

Command	Description
Property Editor	Toggle the display of the Property Editor, in the same space as the Message List and Log Viewer (at the center of the SBM Composer window, below the editor pane). When more than one of these is open at the same time, they appear as tabs.
Message List	Toggle the display of information generated during the design phase of a process app. The Message List occupies the same space as the Property Editor and Log Viewer. When more than one of these is open at the same time, they appear as tabs.
Log Viewer	Toggle display of information generated in a log file while a process app is running in the SBM User Workspace. The Log Viewer occupies the same space as the Property Editor and Message List. When more than one of these is open at the same time, they appear as tabs.

Find

Command	Description
Find Items	Opens the **Find Items** dialog box. For information about this dialog box, see Find Items Dialog Box [page 632].
Where Used	Opens the **Where Used?** dialog box. For more information about this dialog box, see Where Used Dialog Box [page 645].

Resources

Command	Description
Serena Logo	Opens the http://www.serena.com Web site.
Start Page	Opens the SBM Composer Start Page, from which you can see procedures and videos to help you get started with SBM Composer. The Start Page also contains other useful resources that you can view or download.

Deployment Tab of the Ribbon

The **Deployment** tab is in the Ribbon when SBM Composer is first opened. It contains common commands that are available in all of the editors.

The following tables describe the commands available in the various areas on the **Deployment** tab of the Ribbon.

Deployment

Command	Description
Validate	Validate the open process app to determine whether it is ready to *publish* [page 692] to Application Administrator.
Publish	Package a process app and make it available in Application Administrator. Publishing makes a process app available for *deployment* [page 683].
Deploy	Deploy the open process app (if you have an *environment* [page 684] set up in Application Administrator to allow deployment from SBM Composer). This command opens the **Deploy Process App** dialog box, in which you specify deployment settings.

Quick Deploy

Command or Option	Description
Quick Deploy [page 693]	Deploy a process app with the settings already specified in the **Deploy Process App** dialog box.
Perform Check In During Deploy	If this option is selected, check in design elements during the deployment and create process app versions in Application Administrator.
Keep Design Elements Checked Out	If this option is selected, check out design elements after the deployment completes. This only applies to design elements that were checked out before the deployment began.

Launch

Command or Option	Description
Launch SBM User Workspace When Deployed	If this option is selected, open the SBM User Workspace after the deployment completes.
Open Application	Open the SBM User Workspace to the selected *application* [page 679] in the process app.
SBM User Workspace	Open the SBM User Workspace without deploying a process app.
Application Administrator	Open Application Administrator.

Note: Internet Explorer presents a security warning when you click **SBM User Workspace** or **Application Administrator**. You can safely click the message and trust the blocked content.

Design Tab of the Ribbon

The **Design** tab is in the Ribbon when you open a workflow or form. Its contents vary depending on the *design element* [page 683] being edited.

Note: To modify the style definitions, click the **Styles** heading in App Explorer.

The following tables describe the commands available in the various areas on the **Design** tab of the Ribbon.

Form Tools

Preview

Command	Description
Preview	View a mockup of the form as would appear to someone using your process app. Use the controls at the top of the preview window to view the form as it would appear in a different workflow, state, or transition, or *role* [page 695].

Form Layout

Command	Description
Expander Layout	Uses a single page with expandable and collapsible headings for privilege sections.
Tab Layout	Uses a separate tab for each privilege section.
Mixed Layout	Uses a both expanders and tabs.
Portal Layout	Uses a "best fit" of privilege sections into a compact area. This layout is suitable to use as a Web page.
Help Layout	Provides an area in which you can type descriptive text to help the user with this form.

Note: The form layouts provide starting points for form designs that you can modify as you want.

Style

Command	Description
Transition Form Style	Set the form to look like an editable *transition form* [page 700].

Command	Description
State Form Style	Set the form to look like a read-only state form.
Control Style	Set the selected form component to look like a control (typically black).
HyperLink Style	Set the selected form component to look like a hyperlink (typically blue with underline).
Label Style	Set the selected form component to look like a label (typically blue).
Expander Style	Set the selected form component to be expandable and collapsible.
Tab Style	Set the selected form component to be a tab that users can click to bring it into focus.

Font

Command	Description
Font name, size, attributes	Set the text attributes of the selected form component.

Background

Command	Description
Color, image name	Use a single background color or to select an imported image to use as the background for the form or the selected form component.
(Image Icons)	Select the left image to have the imported image (if any) repeat horizontally across the background on the form or container. Select the right image to have the imported image repeat vertically along the background on the form or container.

Alignment

Command	Description
(Tool Icons)	Align rows and columns vertically (top, middle, bottom, full-height) and horizontally (left, center, right, full-width). You also use these tools to modify the alignment of a selected form *widget* [page 701]. See Form Widgets [page 223] for related information.

Size

Command	Description
Width, Height	Modify the size of rows and columns on the form you are editing. You also use these controls to modify the dimensions and characteristics of a selected form widget. See Form Widgets [page 223] for related information.

Row & Column

Command	Description
(Tool Icons)	Add a row above or below the selected row on the form being edited, add a column to the left or right of the selected column, or delete the selected row or column.

Workflow Tools

Layout

Command	Description
Auto Arrange	Let SBM Composer create a compact, linear arrangement of the states and transitions in the workflow.
Re-inherit from Parent	*(Sub-workflows only)* Restore the arrangement of states and transitions defined for the parent workflow, as closely as possible.

View Mode

Command	Description
Presentation	Shows icons on states and transitions that represent all design elements that have relationships with the states and transitions. Hover over the icon for a custom form or orchestration workflow to see a thumbnail image of the form or workflow. Click or double-click the icon to view or edit the properties of the state or transition related to the design element.
Relationships	Shows the Relationships bar. For more information, see About the Relationships Bar [page 269].
Properties	Shows icons on states and transitions to indicate that certain properties are set. Click or double-click the icons to view or edit the corresponding properties in the Property Editor.

Show/Hide

Command	Description
Unreachable Paths	Toggle the display of states and transitions that are not connected to the **Submit** state on the workflow being edited.
Disabled Transitions	Toggle the display of transitions with **Disabled** selected in the transition Property Editor.
Deleted States	Toggle the display of states you deleted from the workflow after the workflow was published at least once.  **Note:** States you added and deleted since the last publication are not affected by this setting.
Inheritance	Toggle the display of double lines in a sub-workflow to indicate states and transitions that were inherited from the parent workflow.
Annotations	Toggle the display of text annotations you can add to states, transitions, and workflows.
Labels	Toggle the display of the text labels on transitions.
Grid	Toggle the display of the blue "graph paper" background.

Transition Style

Command	Description
Preferred, Normal, Optional, Exception	Apply a predefined color or line style to provide a visual cue about the nature of the selected transition. Your choice does not affect the appearance or behavior of the transition on the associated form in the SBM User Workspace.

Swimlanes

Command	Description
Horizontal, Vertical	Specify whether you want horizontal or vertical swimlanes. See About Swimlanes [page 331] for information about swimlanes.

Appearance Tab of the Ribbon

The **Appearance** tab is in the Ribbon when you customize the appearance of an annotation or a swimlane.

The following tables describe the commands available in the various areas on the **Appearance** tab.

Style

Command	Description
Style	Set the style of the annotation or swimlane.

Line

Command	Description
Style, width	Set the style and width of the line around the annotation or swimlane.

Font

Command	Description
Font name, size, attributes	Set the text attributes of the selected annotation or swimlane.

Background

Command	Description
Color, image, image orientation	Use a single background color or an imported image to use as the background for the annotation or swimlane. Specify whether you want the image to be repeated vertically, horizontally, or both.

Alignment

Command	Description
(Tool icons)	Align the annotation text or the swimlane label vertically (top, middle, bottom, full-height) and horizontally (left, center, right, full-width).

Script Tab of the Ribbon

The **Script** tab appears in the Ribbon when you add or edit a script.

The following tables describe the commands available in the various areas on the **Script** tab of the Ribbon.

Validation

Command	Description
Validate Script	Click this icon to validate the selected SBM AppScript. Results appear in the Validation Output pane below the script editor.

Font

Command	Description
Font name, size	Use these controls to specify how you want to view the script in the editor.

About App Explorer

App Explorer represents the open process app as a hierarchical tree of items. The process app itself is the uppermost level of the tree.

At the next level are *application* [page 679] references, and the applications and orchestrations you created or added. Each application and *orchestration* [page 690] is further divided into forms, tables, roles, and other design elements they include. Expand and collapse these headings by clicking **+** and **–**.

As your process app grows in complexity, you can use the filter buttons at the bottom of App Explorer—**Workflow Design**, **Data Design**, **Security**, and so on—to view only a few of the headings at a time. Click the **All Items** button to view all the headings again. Click the arrows at the bottom right of App Explorer to show more or fewer buttons.

To change the alphabetical sorting of the design elements within a heading, right-click the header, select **Sort**, and then select **Ascending** or **Descending**.

To reorder the design elements within a heading, right-click the design element and then click **Move Up** or **Move Down**. This lets you move the most important items to the top of their heading, leaving the less important ones at the bottom.

An asterisk (*) marks each design element that you changed since the element was checked in to the SBM Application Repository.

When you click a heading in the App Explorer, its elements are listed in the editor pane to the right. When you click the name of an element in App Explorer, or double-click it in the list to the right, it is displayed in the Editor Pane [page 63].

Editor Pane

The editor pane appears in the center of the SBM Composer window. The content of this area changes, depending on the element that you are editing.

For example, when you select a *role* [page 695] in App Explorer, the role editor shows the privileges for that role. If you select a form, the form editor opens in the editor pane. When you select a *design element* [page 683] in App Explorer, its editor replaces the editor that is currently open. To edit multiple design elements, right-click each element in App Explorer and select **Open in New Tab**.

To rearrange tabs, drag a tab across the row of other tabs. When multiple editors are open, you can tile them to view multiple editors at the same time by clicking on a tab that is not in front, and dragging it down in the editor. As you move the cross-hairs cursor over the diagram that opens, shading indicates where the tab will be placed—above or below, to the left or right, or on top of the tab that is on top. Release the mouse button to drop the tab in that location, or move the cross-hairs cursor back over the dragged tab to cancel. With a little experimentation, you can create as complex an arrangement as you want.

Change the size of the editor pane or the tiled tabs by dragging their borders.

See also Using the List [page 64].

Palette

For many entities in App Explorer, a palette of related objects is displayed to the right of the editor pane.

For example, when you create or edit a form, the palette provides a list of controls, including fields controls that correspond to fields in a table, detail controls for standard detail sections, container controls for organizing the visual presentation of fields, and other controls such as buttons, text, and hyperlinks. If you edit a workflow, the palette contains states, steps, and the various types of transitions.

You can drag these objects onto the entity in the editor pane. You can also double-click some objects to add them to the editor pane.

Zoom Preview

For the form and workflow editors, a zoom preview appears in the lower right corner of the SBM Composer window.

The shaded rectangle represents the portion of the total form or workflow that is visible in the editor window. Drag the rectangle to move to a different part of the form or workflow. Use the slider bar below the zoom preview to change the size of the rectangle, so that more or less of the form or workflow is visible in the editor pane.

Using the List

A list of entities such as references and application workflows opens in the editor pane when you click one of those headings in App Explorer.

Select an item to view or edit in the appropriate Property Editor. You can also double-click an item in the list to open it in the appropriate editor.

Note: You can limit the number of entities that are listed in an editor. To do so, right-click somewhere in the editor and select **Filter**. In the **Look for** box that opens at the top of the editor pane, enter the word or phrase that must be contained in the name of an entity before it can be included in the list. Click **Find** to create the filtered list.

Tip: You can search for items in the list with the Find Dialog Box [page 631].

Customizing Lists and Tables

In an editor that displays a list or table, right-click column headings to use the commands described in the following table.

Command	Description
Hide *Column Name* Column	Hide the selected column. Use the **Columns** command to show hidden columns.

Command	Description
Columns	Show or hide columns you select from the submenu that opens.
Fit Available Width	Extend the columns to the width of the editor pane.
Best Fit	Set the width of this column to accommodate the longest entry in this column.
Best Fit (All Columns)	Optimize the width of all columns to accommodate all entries.
Add Sort Ascending	Sort the list in ascending order.
Add Sort Descending	Sort the list in descending order.
Reset Sort	Restore the default sort order of design elements in App Explorer.
Sort By	Sort by up to three columns.

Property Editors

The Property Editor appears at the bottom of the SBM Composer window, beneath the *design element* [page 683] you are editing. Forms, roles, tables, and the other design elements have different properties, and the corresponding Property Editor groups those properties in tabs that are at the left edge of the Property Editor.

For example, if you click a transition in a workflow, the Property Editor displays the properties for that transition, grouped in **General**, **Options**, **Form**, **Field Privileges**, **Field Overrides**, **Actions**, and other tabs. If you have the workflow checked out, you can edit the properties displayed on the tabs.

From the list at the top of the Property Editor, you can select a different component of the form, workflow, or other design element you are editing. For example, on a long form, this could be easier than scrolling the form editor pane and selecting the field for which you want to edit properties.

Shortcut Keys

You can use shortcut keys instead of the mouse when you work in SBM Composer. These shortcut keys work on standard keyboards only.

- Ribbon Operations [page 66]
- Global Shortcut Keys [page 66]
- Local Shortcut Keys [page 67]
- Auto-Hide and Focus Shortcut Keys [page 68]
- Options Dialog Box Shortcut Keys [page 68]

- Selection Field Shortcut Keys [page 69]
- Form Shortcut Keys [page 69]
- Palette Shortcut Keys [page 70]

Ribbon Operations

You do not have to remember shortcut keys that perform Ribbon operations, because when you press the Alt key, they are highlighted. For example, to open the Composer menu, press the Alt key to highlight the keys, release the Alt key, and then press the M key to open the menu.

Note: If you already know a shortcut key, you can simply press and release Alt, and then press *key* to perform its associated operation. It is important to release the Alt key before pressing *key*, because there are shortcut keys in the editor pane and Property Editors that use the same letter as those in the Ribbon, but require you to press Alt and *key* at the same time. The editor pane and Property Editor shortcut keys take precedence over the Ribbon shortcut keys, so you could get unexpected results if you do not press the Ribbon keys sequentially.

Global Shortcut Keys

The following table lists shortcut keys you can use from anywhere in SBM Composer.

Operation	Shortcut
Open the Composer menu.	Alt, M
Open the **Create New Process App** dialog box.	Ctrl+N
Open the **Open Process App** dialog box.	Ctrl+O
Open the **Import Process App Blueprint** dialog box.	Ctrl+Shift+I
Open a dialog box to export a process app to a file.	Ctrl+Shift+E
Save the open process app.	Alt, S *or* Ctrl+S
Undo the last operation.	Ctrl+Z
Redo the last operation you performed with the "Undo" operation.	Ctrl+Y
Navigate backward.	Alt, 8
Navigate forward.	Alt, 9
Validate process app.	Alt, V *or* Ctrl+Shift+V

Operation	Shortcut
Start a Publish operation.	Alt, P *or* Ctrl+Shift+P
Start a Deploy operation.	Alt, D *or* Ctrl+Shift+D
Start a *Quick Deploy* [page 693] operation.	Alt, Q *or* Ctrl+Shift+Q
Open the **Find Items** dialog box. This dialog box lets you search for all matching items in an editor pane, application, or process app.	Alt, F *or* Ctrl+Shift+F
Zoom in.	Ctrl +
Zoom out.	Ctrl -
Open the SBM User Workspace.	Ctrl+Shift+W
Open Application Administrator.	Ctrl+Shift+M
Open or close Log Viewer.	Ctrl+Shift+L
Open or close Message List.	Ctrl+Shift+G
Open or close Property Editor.	Ctrl+Shift+R

Local Shortcut Keys

The following table lists shortcut keys that you can use only when you have certain parts of SBM Composer in focus. For example, the shortcut key for the "find" operation is available from App Explorer or an editor.

Operation	Shortcut
Open the **Find** dialog box. This dialog box lets you search for matching items one at a time in App Explorer or in a list in the editor pane.	Ctrl+F
Find the next occurance of an item you specified in the **Find** dialog box.	F3
Toggle the presence of the **Look for** area in a list in the editor pane. This area lets you filter the results of a "find" operation.	Ctrl+I
Reset waypoints (small circles on a transition) so the transition follows the line from the center of one state to the center of another state. This gives the workflow a cleaner look.	Shift+Delete

Operation	Shortcut
Show the **Actions** tab on the Property Editor for a transition or state.	Alt+3, A *or* Ctrl+Shift+3, A
Show the **Forms** tab on the Property Editor for a workflow; show the Form tab on the Property Editor of a transition or state.	Alt+3, F *or* Ctrl+Shift+3, F
Show the **Field Privileges** tab on the Property Editor for a workflow, transition, or state.	Alt+3, R *or* Ctrl+Shift+3, R
Show the **Field Overrides** tab on the Property Editor for a workflow, transition, or state.	Alt+3, V *or* Ctrl+Shift+3, V

Auto-Hide and Focus Shortcut Keys

The following table lists shortcut keys you can use to bring focus to major areas of the SBM Composer user interface. If the area in focus is in the "Auto-Hide" mode (that is, the pushpin icon at the top right of the area is unpinned or pointing horizontally), you can use these shortcut keys to hide or show the areas.

Operation	Shortcut
Activate and bring focus to hidden App Explorer.	Ctrl+1
Bring focus to editor pane.	Ctrl+2
Activate and bring focus to hidden Property Editor.	Ctrl+3
Activate and bring focus to hidden palette.	Ctrl+4
Activate and bring focus to hidden Message List.	Ctrl+5
Hide App Explorer, Property Editor, or palette.	Esc

Options Dialog Box Shortcut Keys

The following table lists shortcut keys you can use to switch tabs in the **Options** dialog box. When you hover over the tab name, the shortcut key for the associated tab is shown in a tooltip.

Operation	Shortcut
Switch to **General** tab.	Ctrl+G

Operation	Shortcut
Switch to **Repository** tab.	Ctrl+R
Switch to **Log Viewer** tab.	Ctrl+L
Switch to **Form** tab.	Ctrl+F
Switch to **HTML and Script** tab.	Ctrl+H
Switch to **Table** tab.	Ctrl+T
Switch to **Workflow** tab.	Ctrl+W
Switch to **Resources** tab.	Ctrl+E

Selection Field Shortcut Keys

The following table lists shortcut keys you can use to edit and navigate selections in *Single Selection* and *Multi-Selection* fields.

Operation	Shortcut
Edit a selected value in the **Value** column.	Enter *or* F2
Apply changes to a selected value.	Enter
Discard changes to a selected value.	Esc
Add a new value.	Insert
Delete a selected value.	Delete
Open the drop-down list in the **Status** column.	Alt+Down
Select a value in the **Status** column.	Up, Down, Left, Right
Close the drop-down list in the **Status** column.	Esc
Change the value of the **Weight** column.	Ctrl+Up, Ctrl+Down

Form Shortcut Keys

The following table lists shortcut keys you can use in the form preview and form editor.

Operation	Shortcut
Open form preview mode.	F5

Operation	Shortcut
Navigate between cells in the form editor (the first cell must be selected).	Up, Down, Left, Right
Select tab on Tab container control.	Left, Right
Select top left cell of a focused form or container control.	Enter
Select parent control, tab, or cell.	Esc
Expand a container control.	+ (in Num Lock mode)
Collapse a container control.	- (in Num Lock mode)

Palette Shortcut Keys

The following table lists shortcut keys you can use from any palette (**Workflow Palette**, **Form Palette**, and so on).

Operation	Shortcut
Display shortcut keys for the selected palette.	Alt+4 *or* Shift+Ctrl+4
Select an item in the palette.	Select the palette section and then press Up or Down. (To select the palette, press Ctrl+4.)
Add item from a palette to the editor pane.	Select an appropriate cell in the editor (for forms), navigate to the item in the palette, and then press Enter.
Expand a palette section.	+ (in Num Lock mode)
Collapse a palette section.	- (in Num Lock mode)
Move to the top of the palette.	Home
Move to the bottom of the palette.	End
Move to a specific item in the palette.	Select the palette section and then press the first letter of the item.
Move to the top of the palette.	PgUp (repeat as needed)

Operation	Shortcut
Move to the bottom of the palette.	PgDn (repeat as needed)

Part 2: Working with Process Apps

This section contains information about creating and managing process apps in SBM Composer:

Chapter 4: About Process Apps

A process app is a set of business processes whose purpose is to address specific business requirements. Most business organizations have unique business process flows. In SBM Composer those processes are represented by applications and orchestrations.

A process app includes one or more applications that will be used to track primary items in the SBM SBM User Workspace. A process app can also include one or more orchestrations that provide the logic for the invocation of *Web services* [page 701].

In SBM Composer, you create process apps independently from the *environment* [page 684] on which they are intended to run. When process apps are deployed to runtime environments, they interact with and support data in the environment, such as user and group data and tracked items that are maintained in the SBM database. Likewise, process apps that contain orchestrations, which are deployed to components in the system processing environment, are initiated based on SBM events. This separation of the process app and the "runtime" data lets you design process apps that are reusable and adaptable to diverse supported processing environments. You can distribute process apps created in SBM Composer to different users to use "as is" or to update to suit specific business requirements. For example, a process app created for the purpose of tracking incidents in a technical support organization could also, with a few modifications, be used to track issues in an IT organization. Another benefit of the separation between the process app and the runtime data is that the continued development of aspects of a process app can occur without disruption of an organization's normal business operations.

You can make process apps available for use by publishing them to Application Administrator, where the process apps are prepared for *deployment* [page 683] to various environments. This includes defining the environments and supplying *endpoint* [page 684] information for the process apps prior to deploying them to runtime environments.

With the "export" feature in SBM Composer, process apps can be also be distributed for use in other supported environments or to developers who do not have access to the repository.

After a process app is deployed to the SBM Application Engine, most design elements, such as workflows, tables, fields, and other elements that support the tracking of items in the SBM Application Engine are viewable (read-only) in SBM System Administrator. This means that these design elements cannot be modified directly in SBM System Administrator but must be modified in SBM Composer. To implement any changes in the SBM Application Engine, the process app containing the modified elements must be either republished to the Application Administrator and redeployed (if endpoint information has changed) or redeployed directly from SBM Composer (if the environment has been set up to allow that).

About Applications and Orchestrations

Applications and orchestrations are components of process apps that define processes you intend to implement in supported environments. Applications contain definitions of tables, fields, workflows, states, transitions, actions, triggers, scripts, forms, roles, *Web services* [page 701], and other design elements that support processes associated with tracking primary items in SBM projects.

Applications and orchestrations within a given process app must be uniquely named. Applications and orchestrations in separate process apps must be uniquely named if they are to be deployed to the same *environment* [page 684].

Orchestrations contain information about Web services that can be initiated at specified points during the tracking of a *primary item* [page 691].

About the Global Process App and Global Application

Every SBM database includes the *Global Process App* [page 687], a special process app that contains only the *Global Application* [page 687]. Creating a new *environment* [page 684] or upgrading an existing environment typically includes getting the Global Process App into the SBM Application Repository (as described in the Application Administrator documentation) so that the contents of the Global Application are available for use in other process apps.

The Global Application initially includes the system auxiliary tables—`Companies`, `Contacts`, `Problems`, `Resolutions`, and others—and their associated default icons. In a system upgrade from TeamTrack, the Global Application initially also contains any other existing auxiliary tables and scripts that are not associated with a single specific *application* [page 679] and any existing triggers that are used in transition actions.

You gain access to the contents of the Global Application by creating a reference. (See About References [page 361] for details.) You can also check out the Global Process App, modify its contents, check it in, and *publish* [page 692] and deploy it like any other process app.

When you open the Global Process App, the SBM Application Repository could list several Global Process Apps, identifiable by the names of the environment sets from which they were gotten, as in **Global Process App (*environment-set-name*)**. For example, an *environment set* [page 684] might include Development, Testing, and Production environments for a single SBM system; your company might have multiple environment sets.

CAUTION:

The Global Process App should be deployed *only* to an environment in the same *environment set* as the environment from which the Global Process App was originally gotten. If you deploy the Global Process App to a different environment set, the Global Process App in the deployed-to (target) environment set will contain everything in the deployed Global Process App, and some system-provided elements in the target Global Process App will be overwritten.

The name of the Global Application initially includes the environment set name in parentheses. This helps you remember from which environment set the Global Process App that contained it was obtained. If you decide to rename the Global Application for some reason, you may want to consider leaving the environment set name unchanged in the new name for the Global Application. This helps you avoid the potential problems associated with deploying the Global Application to a different environment set.

For example, if the Global Process App from one environment set is deployed to an environment in a different environment set, and that second Global Process App is gotten (in Application Administrator) into the repository, SBM Composer may display two triggers with the same name, and you have no way to tell which of the Global Process Apps they came from.

In SBM Composer, you can add to the Global Application any other auxiliary tables (and their supporting design elements: table forms, images, JavaScripts, and styles) and scripts you create that you want to make available for use in multiple applications. The Global Application is also where you create triggers to be used in transition actions.

Important: The Global Process App does not have to be deployed unless you modify it.

About Templates

A *template* [page 699] is a type of process app that is a starting point for creating other process apps. For example, if you have some database fields that you want to include in every process app you create, you could create a template that includes those fields.

A template has the following characteristics:

- A template cannot be referenced by another process app. (For information about references, see About References [page 361].)
- A template can reference other process apps (for example, the *Global Process App* [page 687]).
- A template cannot be published or deployed.
- A template can be exported.
- A template has an `.mst` file extension. (A blueprint has an `.msd` file extension.)
- A template can be imported, with all settings preserved.
- A template is added to the **Available Templates** pane of the **Create New Process App** dialog box.

- All process apps created from a template have unique design numbers. Templates do not have design numbers. (For information about design numbers, see About Design Numbers [page 362].)
- A blueprint can be converted into a template, but a template cannot be converted to a blueprint.

Designing a Process App

The following steps describe the general process for creating a process app from the beginning:

1. Determine what types of users (such as Developer, Tester, and Manager) need to use the process app, and create roles for those user types.
2. Determine what data needs to be collected from and viewed by users, and edit the *primary table* [page 692] fields to contain that data.
3. Determine whether custom forms are needed with which users view, supply, and modify the data. If they are needed, create those forms.
4. Determine the flow of data through the processes represented by the process app, and create application workflows and orchestration workflows for those processes.

Chapter 5: Managing Process Apps

This section contains the following information:

Creating a Process App

You can create a new process app based on a process app in AppCentral™ or by using a process app *template* [page 699].

To create a process app:

1. Select the **New** command from the Composer menu.

   The **Create New Process App** dialog box opens.

2. Complete the dialog box as described in Create New Process App Dialog Box [page 626].

Creating a Template

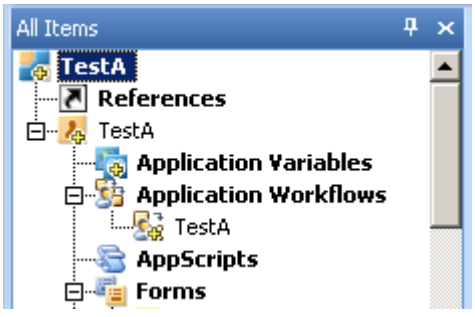
A *template* [page 699] is a type of process app that is a starting point for creating other process apps. For example, if you have some database fields that you want to appear in every process app, you could create a template that includes those fields. You cannot

publish [page 692] or deploy a template. See About Templates [page 77] for more information.

CAUTION: The operation of converting a process app into a template cannot be undone. You cannot convert a template back into a process app. As a precaution, you could back up your process app before you convert it. To do so, from the Composer menu, point to **Import and Export** and then select **Export to File**.

To create a template:

1. Create a process app that you want to use a template or open an existing process app that could be used as a template.
2. Click the process app name in the **All Items** filter of App Explorer.



3. Click **Convert to template** in the process app editor. You are warned that the applications in the process app you are converting will have no design number and that the operation cannot be undone. Click **Yes** if you want to continue with the operation. If you do so, the color of the process app icon changes from yellow and blue to green.

   Note: See About Design Numbers [page 362] for information about design numbers.

4. Export the process app to a file by pointing to **Import and Export** and then selecting **Export to File** from the Composer menu. The **Open** dialog box opens.
5. Save the process app with an `.mst` file extension.
6. Click **New** on the Composer menu. The **Create New Process App** dialog box opens.
7. Click **Browse**.
8. In the **Open** dialog that opens, select the `.mst` file for the template you just created. The template is displayed in the **Available Template** pane of the **Create New Process App** dialog box.

Opening an Existing Process App

To open a process app, select **Open** from the Composer menu.

You can open a process app that you saved to the *Local Cache* [page 688] or a process app that is in the SBM Application Repository (including earlier labeled versions).

Process App Editor

This editor appears when you select the process app name in the App Explorer.

Element	Description
Name	The name given to the process app when it was created. Note that process app names should be unique within the SBM Application Repository.
Category	*(Display only)* The category to which the process app was assigned when it was created.
Description	Optional comments or notes about the purpose of the selected process app.
Revision	Optional comments or notes about this revision of the process app.
Convert to template	Converts the process app into a *template* [page 699], which is a starting point for creating other process apps. For more information, see Creating a Template [page 79].
Publish history	The record of the process app blueprint as published to Application Administrator, including (where applicable) the existing process app as "gotten" from a deployment environment.
Version history	The record of the process app *design element* [page 683] as checked in to the SBM Application Repository. You can use the labels to match design elements to process app blueprints.
Design numbers	The *application* [page 679] name and its associated design number. Applications in some process apps that you download from AppCentral™ have the same design number, because they serve a similar purpose and can reference each other. If the process apps have diverged enough from their original purpose, and you want them to have separate design numbers, click **Reset design numbers**.  **Note:** For more information about design numbers, see About Design Numbers [page 362]. For more information about references, see About References [page 361].

Comparing Process Apps

Comparing the process app that is currently open to another process app gives you an idea of the additions, deletions, and modifications you would need to make if you wanted to change the open process app to match the process app selected for comparison. You could use this comparison to replicate any reported differences in the currently open process app. For example, a table reported as existing only in the compared process app might need to be added to the existing process app.

SBM Composer presents a two-column report. The left column shows a hierarchical tree of the blended content of the two process apps, along with an indication of which process

app contains each *design element* [page 683]: the currently open process app or the process app selected for comparison. (The word "Differs" designates an element that exists in both process apps but that is different in some way.) The right column provides details for every element and links to view images, forms, workflow images, SBM AppScripts, and JavaScripts.

See Compare Process Apps Dialog Box [page 623] for details.

To compare a process app with the currently open process app:

Point to **Compare** from the Composer menu, and then choose one of the following commands:

- **With Local File**: In the standard Windows dialog box that opens, locate the *blueprint file* [page 681] (typically with an `.msd` extension) stored on your computer, and then click **Open**.
- **With Published Process App**: In the Select Published Process App Dialog Box [page 643], select the version of the published process app.

Checking Out a Process App

To check a process app out of the SBM Application Repository, point to **Process App Repository** in the Composer menu and select **Check Out All**.

Leave all design elements selected.

See Check Out Design Elements Dialog Box [page 623] for details.

Checking In a Process App

To check a process app in to the SBM Application Repository, point to **Process App Repository** in the Composer menu and select **Check In All**.

Leave all design elements selected.

See Check In Design Elements Dialog Box [page 622] for details.

Saving a Process App

You can save a process app to the *Local Cache* [page 688] to preserve your work without checking the process app in to the SBM Application Repository.

The Local Cache contains only the design elements that were downloaded from the SBM Application Repository. If you know you will work on the process app while disconnected from the SBM Application Repository, click the **Connected** indicator (in the bottom right corner of the SBM Composer window) and enable the **Work Offline** option on the menu that opens. You are prompted to download any design elements not already saved to the Local Cache.

Note: Unlike the **Export to File** command, the **Save** command does not create a file that you can import into SBM Composer or load into Application Administrator.

To save an open process app:

On the Composer menu, click **Save** (or right-click the process app name in App Explorer, and select **Save**). You can also press Alt and then S, or press Ctrl+S to save the open process app.

Exporting a Process App

The export feature lets users exchange process apps without access to the same SBM Application Repository, and create new process apps based on an exported *template* [page 699] file.

For example, in the first situation, you could export a process app to send the *blueprint file* [page 681] to someone who needs to approve it, but who is not authorized to check in the process app or *publish* [page 692] it. The recipient could import the blueprint using the **Import from File** command on the Composer menu, make any needed changes, and export it again to send the blueprint file back to you.

You could also export a process app and send the blueprint file to someone else to load into Application Administrator, effectively publishing the process app.

In the second situation, you could convert a process app to a template, export the template to file, and then browse for the file in the **Create New Process App** dialog box. After you open the file, the template is displayed as an icon in the **Available Templates** pane of the dialog box. Other designers can then use it to create a new process app.

To export a process app:

1. Open the process app and, from the Composer menu, point to **Import and Export** and then **Export to File**. You can also press Ctrl+Shift+E to export the process app.

   SBM Composer validates the process app. (You cannot export a process app that fails validation.)

2. If the process app is valid, in the **Save As** dialog box, navigate to the directory where you want to save the exported process app.

3. Click **Save**.

Importing a Process App

Importing a *blueprint file* [page 681] that was previously exported from SBM Composer or saved from Application Administrator. You can also import a blueprint file that was downloaded as a prebuilt process app. This creates a local copy of the process app that includes all of the information necessary to associate it with the version in the SBM Application Repository and with any deployed versions.

To import a process app, from the Composer menu, point to **Import and Export** and then select **Import from File**. You can also press Ctrl+Shift+I to import a process app. See Import Process App Blueprint Dialog Box [page 635] for details.

Publishing a Process App

Prerequisites:

Publish privilege (set in Application Administrator)

You *publish* [page 692] a process app to make it available in Application Administrator for *deployment* [page 683] to the SBM Server. If the process app has not been validated or checked in, you are guided through these tasks first.

Important: The process app is automatically published when you use the **Deploy** command to deploy a process app. Use the **Publish** command if you want to publish the process app as a separate operation. You need to do this if you do not have privileges to deploy, but do have privileges to publish.

Note: After a process app is published, you must confirm any request to change the database names of its tables or the internal names of its applications and orchestrations, as such changes may cause incompatibility with other process apps. (You can freely change the logical names of its tables, applications, and workflows.)

Note: When you publish a process app, SBM Composer checks in the design elements you select and then checks them out again. The version number of the process app is incremented accordingly. (This is only the case if you use the **Deploy** command to deploy a process app. If you use the **Quick Deploy** command, you can specify whether the design elements are checked out or remain checked in. For more information, see Using Quick Deploy [page 85].)

To publish a process app:

1. On the Composer menu, click **Publish**. You can alternatively press Alt and then P, or press Ctrl+Shift+P. The **Publish Process App** dialog box opens.
2. Complete the **Publish Process App** dialog box.

   Note: See Publish Process App Dialog Box [page 640] for details.

The Message List indicates the success or failure of publishing the process app.

Deploying a Process App

Prerequisites:

Deploy Process Apps To This Host privilege (set in SBM System Administrator) and **Deploy** privilege (set in Application Administrator)

You deploy a process app when you want to make it available in the SBM User Workspace for users to access. You can deploy a process app using the **Deploy** or **Quick Deploy** command. See Using Quick Deploy [page 85] for information about the second command.

Before you deploy:

Before you can successfully deploy a process app from SBM Composer, the environment to which you want to deploy must have the **Enable Deployment** option selected in the **Composer** list in the **Edit Environment** dialog box in Application Administrator.

To deploy a process app using the Deploy command:

1. From the Composer menu, point to **Deploy** and then select **Deploy** , click the **Deployment** tab on the Ribbon and then click **Deploy**, press Alt and then D, or press Ctrl+Shift+D.

The **Deploy Process App** dialog box opens.

2. Perform the following steps:

   a. Select the environment to which you want to deploy.

   b. Type a version name, if you do not want to use the default value.

   c. Specify whether you want others to deploy the process app.

   See Deploy Process App Dialog Box [page 630] for details.

3. Set deployment options, such as a delayed start time, an e-mail notification, and whether to stop the deployment on warnings or only on errors.

4. If necessary, map the endpoints in your process app to external endpoints. (Endpoints are typically created and mapped automatically, but you can change the endpoints as needed. Some process apps do not require endpoints.)

   See Deploy Process App Dialog Box [page 630] and New Endpoint Dialog Box [page 637] for details.

5. Click **OK**.

Note: After you deploy a process app using the **Deploy** command, SBM Composer checks in the design elements that were checked out. The **Quick Deploy** command provides an option for leaving the design elements checked out so you can continue to design the process app. For more information, see Using Quick Deploy [page 85].)

Note: If the process app you are deploying contains a report with the same name as an existing report in the target *environment* [page 684], and the two reports are in the same project, are stored in the same table, and have the same access level (for example, public or private), *deployment* [page 683] generates a warning about duplicate reports.

Using Quick Deploy

The **Quick Deploy** command lets you deploy a process app without being prompted to provide the information in the **Deploy Process App** dialog box.

Note: If this is the first time you are deploying this process app, it is recommended that you use the **Deploy** command, not the **Quick Deploy** command. The **Deploy** command lets you see the default settings and change them, if necessary.

To deploy a process app using the **Quick Deploy** command:

1. From the Composer menu, point to **Deploy** and then select **Quick Deploy** , click the **Deployment** tab on the Ribbon and then click **Quick Deploy**, press Alt and then Q, or press Ctrl+Shift+Q.

2. In the **Quick Deploy** area of the **Deployment** tab, clear the **Create Versions of Process App Elements** check box if you do not want to create process app versions in Application Administrator.

   It is a good practice clear this check box when you want to deploy and test a process app you are designing, but are not ready to commit your changes. This also prevents extraneous versions of the process app in the database.

Select this check box if you are ready to commit your changes.

Note: This check box cannot be cleared if the environment to which you are deploying does not have the **Enable Deployment** option selected in the **Composer** list in the **Edit Environment** dialog box in Application Administrator.

3. Select the **Keep Design Elements Checked Out** check box if you want design elements that were already checked out to remain checked out after the deployment. This eliminates the step of checking out design elements after you deploy, so you can continue to design.

   Clear this check box if you have finished your design work and want to use the **Quick Deploy** command to deploy the process app.

   Note: This check box is unavailable if the **Create Versions of Process App Elements** check box is cleared.

4. In the **Deploy To** list, select the environment to which you want to deploy. You are prompted to map any unmapped endpoints that cannot be automatically mapped.
5. Click **Quick Deploy**.

To determine the status of the deployment, look at the Message List, or right-click the process app name at the top of App Explorer, and select **Get Deployment Status**.

Using Other Deployment Features

After you perform a deployment, you might want to have the SBM User Workspace open automatically so you can see how the process app works, or open the SBM User Workspace without doing a deployment. You also might want to open Application Administrator to give other users permission to deploy the process app.

To use other deployment features:

1. Click the **Deployment** tab on the Ribbon.
2. In the **Launch** area of the tab, clear or select the **Launch User Workspace When Deployed** check box to specify whether you want to open the SBM User Workspace after the deployment completes.
3. An *application* [page 679] is selected in the **Open Application** list. If you want to change it, select another one.
4. Click SBM User Workspace to open the SBM User Workspace manually, without performing a deployment. The SBM User Workspace opens to the application specified in the **Open Application** list, from the environment to which the process app was last deployed.
5. Click **Application Administrator** to open Application Administrator.

Deleting a Process App

To delete a process app, right-click the process app name in the App Explorer and select **Delete Process App** (or click **Delete** on the Composer menu), and then select whether you want to delete the process app from the SBM Application Repository as well as the copy in the *Local Cache* [page 688].

Note: Deleting a process app from the SBM Application Repository does not affect any environments to which it has been deployed.

If the deleted process app is already deployed, a SBM System Administrator user has to eliminate application-name and table-name conflicts by deleting the *primary table* [page 692] and associated auxiliary tables for applications in that process app. The new process app cannot be deployed until the user does this.

Upgrading a Process App

A *blueprint* is created in SBM Composer and contains process app design elements such as workflows, orchestrations, roles, fields, and custom forms.

When you upgrade to a later version of SBM Composer, process apps created in the earlier release might need to be upgraded. To do so, upgrade the blueprint that contains the process app.

To upgrade a process app:

1. From the Composer menu, point to **Import and Export** and then select **Import from File**. The **Import Process App Blueprint** dialog box opens.
2. Select the *blueprint file* [page 681] for the process app, and click **Open**. A message tells you if you need to upgrade the blueprint to work with the current version of SBM Composer.
3. Click **OK**. The **Import Process App** box shows the status of the upgrade. After the process app is upgraded, it opens in SBM Composer.
4. Deploy or redeploy the process app.

Upgrading a Snapshot

A *snapshot* [page 697] is created by Application Administrator and contains specific database instance information. It contains everything that a blueprint contains, plus user data such as folders, projects, and users.

Snapshots created in an earlier version of SBM Composer might need to be upgraded when a new version is installed. If you attempt to load or promote a snapshot created in an earlier version of SBM into Application Administrator, you receive a warning that the snapshot needs to be upgraded.

Important: After you upgrade a snapshot, you cannot edit the process app that includes it in an earlier version of SBM Composer.

To upgrade a snapshot:

1. From the Composer menu, point to **Import and Export** and then select **Upgrade Process App Snapshot**. The **Upgrade Process App Snapshot** dialog box opens.
2. Select the snapshot file (either an `.mss` or a `.zip` file) and then click **Open**.

After the snapshot is upgraded and saved to a file, the upgraded snapshot file must be loaded into Application Administrator (using the **Load from File** command) and then promoted from Application Administrator into the SBM Application Engine Web Server (using the **Promote** command).

Working in a Patch Context

A *patch context* [page 691] allows parallel and concurrent development to occur on the same process app, so that patches can be applied to production process apps as ongoing development of the main process app continues. When performing maintenance work, a *designer* [page 683] can create a patch context by opening a published process app by its label.

The patch context serves as the baseline for the maintenance work. Any number of process app designers can do concurrent development in a patch context.

Restriction: Only one process app designer can work on a single *design element* [page 683] at the same time.

Changes a designer makes when working in a patch context apply only to the patch context. They do not affect ongoing development of the head (tip) version of the process app.

Note: A patch context is not the same thing as a branch in a version control application.

Some changes can be made directly in a patch context. If you want these changes to also apply to the head version, make them manually in the head version. Other changes cannot be made directly in a patch context, and are described in the following section.

Adding Design Elements to a Patch Context

To protect against data loss, the following design elements cannot be added directly to a patch context:

- Applications, including these design elements:
  - Tables
  - Fields
  - Selections for *Single Selection* and *Multi-Selection* fields
  - Application Workflows
  - States
  - Report Definitions
  - Roles
- Orchestrations and orchestration workflows

You can, however, copy these design elements from the head version to the patch version. If a design element does not exist in the head version, create it there first.

Restriction: Applications and orchestrations are exceptions to this. You cannot add applications or orchestrations to a patch context, and cannot copy them from the head version.

When you right-click the design element heading in App Explorer in the patch context, and then select **Add Existing <Design Element>**, the **Add Existing <Design Element> from the Head (Tip) Version** dialog box opens. In this dialog box, select the design element that you want to add and then click the **Add** button.

Note: Design elements containing fields, states, tables, and so on in the head version must be checked in before you can copy these entities to the patch context.

When adding items from the head version, the parent design element in the patch context must be checked out. For example, to add a workflow from the head version, make sure the *application* [page 679] in the patch context is checked out.

After a table, *application workflow* [page 680], report definition, or *role* [page 695] is added from the head version, that version of the design element is immediately labeled with the patch label, and the parent design element in the patch context is checked in (and checked out again if it was previously checked out).

After a field or state is added from the head version, the field or state is simply added to the parent design element and the parent design element in the patch context is checked in (and checked out again if it was previously checked out).

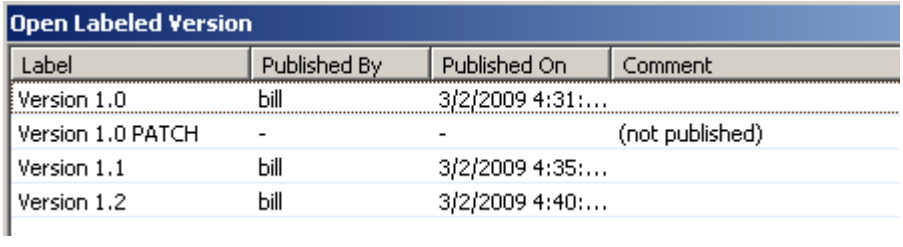
Example of Working in the Patch Context

Version 1.0 of a process app is in production. Susan checks out Version 1.0 (the head version) to continue main development on the process app.

At the same time, a patch needs to be applied to Version 1.0 for customers using the process app in production. The owner of a state needs to change. Bill opens Version 1.0 by its label in the **Open Labeled Version** dialog box. The *Version 1.0* label becomes *Version 1.0 PATCH*. Bill changes the owner of the state in the patch context (Version 1.0 PATCH).

Bill deploys Version 1.0 PATCH, and its label becomes *Version 1.1*. He manually changes the owner of the state in the head version so it matches the patch context. When he deploys the head version, its label becomes *Version 1.2*.

The **Open Labeled Version** dialog box resulting from this example is shown in the following illustration.



Chapter 6: Process App Tutorial

The exercises in this tutorial demonstrate how to use SBM Composer to create and deploy a simple process app. The process app will allow you to create an item, assign the item to a user, and then have the user close the item.

The process app that you create will be used as the basis for the Chapter 29: Orchestration Tutorials [page 563].

Note: The complete process app tutorial will take around thirty minutes to complete.

This tutorial covers the following topics:

Taking a Break During the Tutorials

If you need to stop and then return to this tutorial, be sure to save your process app. Otherwise, you will have to start from the beginning of the tutorial to complete all the exercises.

If you need to take a break:

1. Save your process app:
   a. On the Quick Access toolbar, click the **Save locally** icon.

      A message box opens reminding you that the design elements have been saved to the *Local Cache* [page 688] only.

   b. Click **OK**.
2. If you closed SBM Composer after saving the process app:
   a. Start SBM Composer.
   b. On the Composer menu, click **Open**.

c. In the **Open Process App** dialog box, select **MyProcessApp**, and then click **Open.**

d. In App Explorer, select the **All Items** filter.

Step 1: Create an Application Workflow

In this exercise, you create a new process app with an *application workflow* [page 680]. You will use this process app throughout the rest of this tutorial.

To create an application workflow:

1. Start SBM Composer.
2. Click the Composer button, and then select **New**.
3. In the **Create New Process App** dialog box, select **Application Process App** and then click **Create**.
4. In the **Configure Process App** dialog box, in the **Process App Name** box, type `MyProcessApp`.
5. In the **Category** box, type `Examples`. Categories help organize your process apps.
6. In the **Application Name** box, type `MyApp`. This causes the other fields to be populated with variations of MyApp. For information on what each field refers to, press **F1**.
7. Change the **Workflow Name** to `MyAppWorkflow`.
8. Click **OK**. The new process app, the **MyApp** *application* [page 679], and the **MyAppWorkflow** workflow appear in App Explorer.
9. Continue to Step 2: Add States and Transitions [page 92].

Step 2: Add States and Transitions

In this exercise, you add an active state called *Assigned* and an inactive state called *Closed* to **MyAppWorkflow**. Then you create transitions between the **New** state and the **Assigned** state and between the **Assigned** state and the **Closed** state.

To add states and transitions in MyAppWorkflow:

1. In App Explorer, under the **Application Workflows** heading, select **MyAppWorkflow**.
2. Set the **Manager** *role* [page 695] as the owner of the **New** state. This determines which users will own items as they are submitted.

   a. Select the **New** state in the workflow.

   b. Launch the **Add Owner Field** wizard by selecting **<Add owner>** in the **Owner** field of the **General** tab on the Property Editor.

   c. On the **Select or Add Roles** panel, enter the following information for the fields then click **Add Role**.

   - **Name**—`Manager`

- **Template**—`Administrator`

d. Select **Create New Field** and then click **Finish**. A new role and new field have now been created. The Manager role has all permissions. These permissions will be reduced in Step 4: Define Security [page 94].

3. In the **States** section of the **Workflow Palette**, drag an active state onto the application workflow editor and drop it to the right of the **New** state.
4. In the Property Editor, change the **Name** to `Assigned` and for **Owner**, select **<Add Owner>**. This time in the **Add Owner Field** wizard, select the **User** role and choose to create a new field called `Employee`.
5. In the **States** section of the **Workflow Palette**, drag an inactive state onto the application workflow editor and drop it to the right of the **Assigned** state. Change the **Name** to `Closed`.
6. In the **Transitions** section of the **Workflow Palette**, drag a regular transition onto **New**, release the mouse button, and then click the **Assigned** state.

   This creates a transition from the **New** state to the **Assigned** state.
7. Change the **Name** to `Assign`, and then press the Enter key.
8. Add another regular transition from the **Assigned** state to the **Closed** state, and change the transition name to `Close`. The final workflow looks like this:
9. Continue to Step 3: Define Fields [page 93].

Step 3: Define Fields

In this exercise, you define the fields in the **MyApp** *primary table* [page 692]. You add selection values to the existing *Item Type* field, and add a new custom field to your table. The custom field tracks whether the item was submitted by a customer or not.

To define fields in MyApp primary table:

1. In App Explorer, select **Data Design**. The **MyApp** table is displayed in the table editor. If not, select the **MyApp** node.
2. Add selection values to the *Item Type* field:
   a. In the MyApp (Primary Table), select the *Item Type* field.
   b. In the Property Editor, add a selection by clicking **Add** on the **Options** tab.
   c. Change the selection's **Value** to `Bug` and set the **Item ID Prefix** to `BUG`.
   d. Add another selection with a value of `Enhancement` and set the **Item ID Prefix** to `ENH`.
3. Add a custom field to track customer submitted issues:
   a. In the **Field Types** section of the **Table Palette**, drag a *Binary/Trinary* field onto the table editor.
   b. On the **General** tab of the Property Editor, change the name of the field to `Customer Submitted`.

c. On the **Options** tab of the Property Editor, type `Yes` for the **First Value** and `No` for the **Second Value**.

d. On the **Attributes** tab of the Property Editor, select `No` as the **Default Value**.

4. Continue to Step 4: Define Security [page 94].

Step 4: Define Security

In this exercise, you assign privileges to the **User** *role* [page 695] and restrict privileges for the **Manager** role.

To add and define roles in the MyApp process app:

1. In App Explorer, select **Security**.
2. Select the **User** role.
3. Select the following privileges for the **User** role:
   - Submit New Items
   - Own Items
   - View All Items
   - View Attachments if Owner
   - Update Item if Owner
   - Transition Item if Owner
4. In App Explorer, select **Manager**.
5. In the Property Editor, clear the check boxes for the following privileges for the **Manager** role:
   - Delete Items
   - Delete Notes on any Item
   - Delete Guest-Level reports
   - Delete Manager-Level reports

   Tip: In a production process app, you might restrict privileges for the **Manager** role, such as removing the privileges to delete items and reports. These privileges should be reserved for administrators.
6. Continue to Step 5: Deploy the Process App [page 94].

Step 5: Deploy the Process App

You must *publish* [page 692] and deploy the process app to your SBM Server to use the process app.

To publish and deploy the process app:

1. Click the Composer button, and then select **Publish**.

- If the process app has not been saved yet, or if any part of the process app has changed since the last time it was saved, a message box opens reminding you to save the changes. Click **OK**.
- If any of the design elements in the process app are not checked in to the SBM Application Repository, a message box opens reminding you to check them in. Click **OK**.

2. When the **Check In Design Elements** dialog box opens, you can enter an optional comment in the **Comment** box, and then click **OK**.

   The **Publish Process App** dialog box opens.

3. Complete the **Publish Process App** dialog box:
   - Type an optional comment in the **Comment** box.
   - Type text to identify the version of the process app in the **Label** box.
   - Select the **Allow others to deploy this process app** check box under **Visibility**.
4. Click **Publish**.

   A message box opens indicating whether the process app published successfully or the operation failed.

5. On the **Deployment** tab of the Ribbon, click the **Deploy** button.

   The **Deploy Process App** dialog box opens.

6. From the **Environment** list, select a runtime environment, and then click **Deploy**.

   In the Message List, a message appears notifying you that the deployment process started successfully.

7. Wait for deployment to complete.
   - If the deployment operation is successful, the following message appears in the Message List: "Deployment of 'MyProcessApp' has completed."
   - If the deployment operation fails, the following message appears in the Message List: "Deployment of 'MyProcessApp' has aborted."

   If the Message List is not available, on the **Home** tab of the Ribbon, in the **Common Views** area, select the **Message List** check box. If this check box is already selected and the details are still not visible, select the **Message List** tab in the area under the editor pane.

8. Continue to Step 6: Associate Users with Roles [page 95].

Step 6: Associate Users with Roles

In this exercise, you associate users with roles using SBM System Administrator.

To associate users with roles:

1. Add users to the **Manager** and **User** roles that you created. For example, to add Joe Manager to the **Manager** role:

a. Log on to SBM System Administrator.

b. Click the **Users** tab.

c. Select **Joe Manager**.

d. Click **Edit**.

e. Click the **Roles** tab.

f. In the **For Project** panel, select **MyAppProject**. The roles are displayed in the panel on the right.

g. Click **Manager** and then click **Enable**.

h. Click **OK**.

2. Continue to Step 7: Run the Process App [page 96].

Step 7: Run the Process App

In this exercise, you test MyProcessApp by running it in the SBM User Workspace.

To run a process app in the SBM User Workspace:

1. Log on to the SBM User Workspace.
2. Select the **MyApp** tab.

   If the **MyApp** tab is not visible, click the **More** tab, and then select **MyApp** in the list.
3. On the **Task Page**, under **Submit**, click the **Submit a new MyApp** link.
4. In the **Submit Tree**, click the **MyAppProject** link.

   The first submit form opens.
5. Complete the fields, entering text in the **Title** box and selecting values for **Item Type**, **Managers**, **Employee**, and **Customer Submitted**, and then click **OK**.

   The item moves to the **New** state, and the owner becomes the manager selected in the **Managers** field.
6. Click the **Assign** button.

   The **Assign** transition form opens, where you can update the fields, such as selecting a different employee.
7. Click **OK.**

   The item is assigned to the user selected in the **Employee** field. The user becomes the owner of the item as the item moves to the **Assigned** state.
8. Close the item by clicking the **Close** transition on the item.
9. To locate the new item, click the **Search** filter.
10. In the **Projects** field, select **MyAppProject** .
11. Click **Search.**

12. In the **Search Results** list, find the new item.
13. To access the item, click the link in the **Item Id** column.

The item has now completed the workflow. It has moved through three states. In the **New** and **Assigned** states, the item was assigned to the selected manager or employee.

Part 3: Working with Applications

This section contains information on creating and managing applications in SBM Composer:

Chapter 7: Introducing Applications

Applications are definitions of team processes. Applications created in SBM Composer contain elements required to support the tracking of project items in SBM. For example, the *application* [page 679] defines the single *primary table* [page 692] required to store tracked primary items and, optionally, one or more auxiliary tables used for storing auxiliary data that can be used to support the tracking of primary items.

Publishing and deploying process apps transfers the definitions within them to the SBM Application Engine. Using SBM System Administrator, the SBM administrator performs various tasks to assimilate the process apps into the processing environment. For example, the administrator assigns the workflows defined in the process app applications to projects in SBM. The administrator exercises project-level *overrides* [page 690], including changing default field values. The administrator also performs configuration tasks, such as creating projects, assigning users to roles and groups, creating e-mail notifications, and defining system settings.

Applications are represented by named tabs in the SBM SBM User Workspace. *Application tabs* [page 680] provide access to applications implemented in the SBM Application Engine.

Applications that define the elements necessary for the tracking of primary project items automatically include a primary table. You can, however, create applications that do not contain a primary table, but contain one or more auxiliary tables only. Such applications are deployed to the SBM Application Engine solely for the purpose of collecting auxiliary data that can be referred to by multiple applications. The *Global Process App* [page 687] is a special example of such an application.

Application Names

Applications within a given deployment environment must be uniquely named, even if they are used in different process apps. Applications within a given process app must be uniquely named.

Applications created independently in different process apps can have the same name. Changes to one do not affect the other.

Creating an Application

See Chapter 6: Process App Tutorial [page 91] for a tutorial.

Applications deployed to a given deployment environment must be uniquely named, even if they are used in different process apps. Applications within a given process app must be uniquely named.

Applications created independently in different process apps can have the same name. Changes to one do not affect the other.

Application Editor

This editor appears when you select an *application* [page 679] in App Explorer.

Element	Description
Type	Indicates that you are working on an application.
Logical name	The name by which the application is known. Application names must be unique within a process app. The logical name is displayed as a tool tip when a user holds the mouse pointer over the application tab in the SBM User Workspace.
Version	The version of the application *design element* [page 683] as checked in to the SBM Application Repository.
Internal name	The unique name by which the application is identified in the SBM database.
Prefix	A required prefix of up to 3 characters, constrained as described next to **Prefix**.
Tab name	A label of up to 16 characters that is displayed to a user in the SBM User Workspace.
Icon	An application image that appears in the SBM User Workspace. In the area to the right, the image is displayed in three sizes. This lets you determine how to scale your image for the best results. To use the default image, select **(Default image)** is selected in the drop-down list. To add a new image, select **(New image...)** in the drop-down list and navigate to the image you want. You can use any HTML-compatible image format (for example, `.png`, `.gif`, `.jpg`). To use an existing image, select the image in the drop-down list. **Note:** The new and existing images appear under the **Images** heading in App Explorer.
Description	Optional comments or notes about the purpose of the application.
History	The history of versions of the application from the SBM Application Repository.

Element	Description
Design number	A unique design number assigned to this application. If this application is in a process app that is used as a *template* [page 699], "template" is displayed. **Note:** For information about design numbers, see About Design Numbers [page 362]. For information about templates, see About Templates [page 77].

Understanding Relationships Between Items

This topic describes terms and concepts that help you understand the relationships between items in an *application* [page 679].

Primary Items

An *application* [page 679] uses its *primary table* [page 692] to define a single type of *primary item* [page 691]. The primary table determines the set of data fields the primary item contains. At runtime, when a primary item is created, it occupies a row in the primary table. This means that all items created in an application have the same set of data fields, but the value of any particular field will be specific to the particular item or row in the primary table.

An application defines one or more workflows. All of the workflows in an application create and process the same type of items. However, each workflow can be designed to process the various data fields in the item in a different way.

Primary items are created by **Submit** transitions. These transitions can be initiated directly by a user, or indirectly from an existing item through a **Post**, **Subtask**, **Copy**, or **Publish** transition. After items are created, they can be connected through a link or a relational field reference.

Links

Links use a built-in attachment mechanism that is available in all workflows. Users can explicitly create links between existing items. Links can be created implicitly through **Post**, **Subtask**, **Copy**, or **Publish** transitions. With implicit link creation, a link can be created between the existing (original) item that is being transitioned and the new item.

Links can connect items in the following ways:

- The original item connects to the new item (one-way link).
- The new item connects to the original item (one-way link).
- The original item and the new item connect to each other (two-way link).

Note: The Subtask transition implicitly creates a two-way link that appears in the **Subtasks** section of the item in the SBM User Workspace. You can optionally configure the **Subtask** transition to create item links in addition to the **Subtask** section links. These links appear in the **Attachments** section of the item in the SBM User Workspace.

Relational Field References

Note: In this section, "relational field references" means relationships between items, established through *Relational* fields. These fields can be in applications in the same process app or in applications in other process apps. (In the latter case, you must first create a reference to the application in the other process app. This is a different type of reference. For more information, see About References [page 361].)

Relational field references are often used to model parent-child relationships between items where the parent item stores a reference to one or more child items in a *Relational* field. This creates a one-way, one-to-one or one-to-many relationship.

You can also use relational field references to create multiple relationships, two-way relationships, and chains of relationships between items. An additional *Relational* field must be added to the primary table of the referencing item for each additional relational field reference. For example, to create a two-way relationship between items from two different applications, each primary table must define a *Relational* field that points to the primary table of the other application.

Transitions

Links, subtasks, and references allow items that have a relationship to be transitioned automatically when one of the related items is transitioned. There are three mechanisms for this:

- **Subtask transitions** work with Subtask attachment links.
- **Transition actions** allow an item to automatically transition another item that it references or that references it. Transition actions work with references.
- **Trigger actions** work across applications that may be aware of each other, but do not have to be. Trigger actions can be configured to work with subtasks, references, and links.

Chapter 8: Managing Roles

This section contains the following information:

About Roles

Roles are created in SBM Composer as part of an process app, which can comprise multiple applications. Roles span the applications within the process app, and serve two functions:

- They are a named collection of privileges. The privileges secure user actions and data access. For example, a role named User could be a collection of privileges suitable for someone to whom items are assigned but who has no administrative tasks. Someone with that role could be unable to execute some transitions and view some fields in the SBM User Workspace.
- They are a means to populate selection lists in the SBM User Workspace for *User*, *Multi-User*, and *Multi-Group* fields. You associate roles with these fields in SBM Composer.

The two functions can be connected or disconnected. When the functions are connected, a role provides privileges and populates a selection list. When the functions are disconnected, a role either populates a selection list with no privileges, or only provides privileges.

Privileges fall into two broad categories:

- System privileges control a user's ability to deploy and promote process apps, and perform actions that affect application configuration.
- Application privileges define what a user can view and act on within an application. These privileges allow fine-grained control over items, attachments, notes, reports, workflows, and fields.

Roles are distinct from groups, which are named collections of users. Administrators can use groups to identify a set of users based on criteria other than job function. A group could be created for a particular project, for example, or for a division within the company. You can assign roles to a group.

Note: If you associate a user or group with a role, and the role contains privileges that conflict with the user or group level of product access, those privileges are not granted to the user or group.

Users and groups are associated with roles for particular projects in SBM System Administrator.

Note: When groups or users are copied, role assignments are copied with them. Also, when groups or users are imported through LDAP using a template group or user, the role assignments associated with the template are copied to the new group or user.

Use Cases for Roles

In SBM, the process design environment (SBM Composer is independent of the administration environment where you are concerned with specific projects, users, and groups. At design time, you might not know which specific users or groups will interact with the application that you are creating or modifying; however, you still need to be concerned with privileges from a design standpoint. Roles are the solution to this problem, because in the design environment, a role is associated with privileges but not with specific users or groups.

If you are administering an application in the runtime environment, you must assign actual users (individually or as members of groups) to roles for each project. A given user or group could have different roles in different projects. For example, two projects could have two different people assigned to the Build Manager role. The *role* is the same, but the *person filling the role* for a specific project (or for a specific deployment of the process app) is different.

A good place to start when deciding what roles to create is to look at the user-type fields in your primary items—particularly fields that determine ownership for items. For example, if you have an Engineer field that determines ownership of an item in a particular state, Engineer might be an appropriate role to create.

The following sections describe various ways you can use roles.

Creating Specialized Roles

Every application has two default roles: Administrator and User. These broad roles could be adequate in simple applications. However, most applications require more specialized roles.

In an employee time-off application, there could be Employee, Manager, and Payroll roles. The Manager and Payroll roles have privileges that are not part of the Employee *role* [page 695]. For example, the Manager role could have the Delete Items privilege. The Advanced Fields section of the SBM User Workspace could be for payroll purposes only, so the Payroll role would have the Advanced Fields privileges.

Employees requesting vacation time have the Employee role, the employee's manager who approves or rejects the request has the Employee and Manager roles, and the payroll clerk has the Employee and Payroll roles. Because privileges are cumulative, the manager has all of the privileges granted by the Employee role, as well as any additional privileges granted by the Manager role. Similarly, the payroll clerk has all of the privileges granted by the Employee role, as well as any additional privileges granted by the Payroll role.

Assigning Privileges to Groups

Administrators create groups to limit the number of privilege sets that they need to maintain. For example, there could be a group called Development Team. The members of the group include individual users associated with the Engineering, Quality Assurance, and Documentation roles. Suppose a system privilege needs to be added for these users. An administrator can use SBM System Administrator Web Administrator to quickly add the

system privilege to the group, and the privilege set for all of the individual users would be updated.

Tip: It is best to use groups to assign system privileges only, and to use roles to assign other privileges.

Associating Roles with a User Field

In an issue management application, after engineers finish fixing an issue, they transition the issue from the Fixing Issue state to the Peer Review state. In this state, another engineer reviews the code that the first engineer changed to fix the issue. The owner of the Peer Review state is Peer Reviewer, which is a *User* field associated with the Engineer role.

The engineer selects the peer reviewer from a list box that is populated with users with the Engineer role.

Associating Roles with a Multi-User Field

In a change request application, after a technical analyst determines that a change request should be considered by the Change Approval Board (CAB), he or she transitions the change request to the In Review state. The owner of this state is Reviewers, which is a *Multi-User* field associated with the CAB Members role. This role has privileges to own and view change requests, but not submit or delete them.

The technical analyst selects the appropriate board members from a list box that is populated with all users with the CAB Members role.

Associating Roles with a Multi-Group Field

Multi-Group fields are useful when you want to take advantage of queueing. Queueing takes place when users evaluate issues in a backlog and assign certain issues to themselves.

For example, in a help desk application for a company, employees submit issues for technical problems they are experiencing. The issues move into a Backlog state. The *secondary owner* [page 696] for this state is a *Multi-Group* field called *Employee Location*. This field is associated with two roles: Technician and Manager. The Manager role is associated with the field because managers often work part-time as managers and part-time as technicians.

In the SBM User Workspace, the drop-down list for this field is populated by groups that represent the various company sites. Each group is comprised of technicians and managers who are located at the site. All of the technicians and managers at the selected site own the issues in the Backlog state, and can assign the issues to themselves.

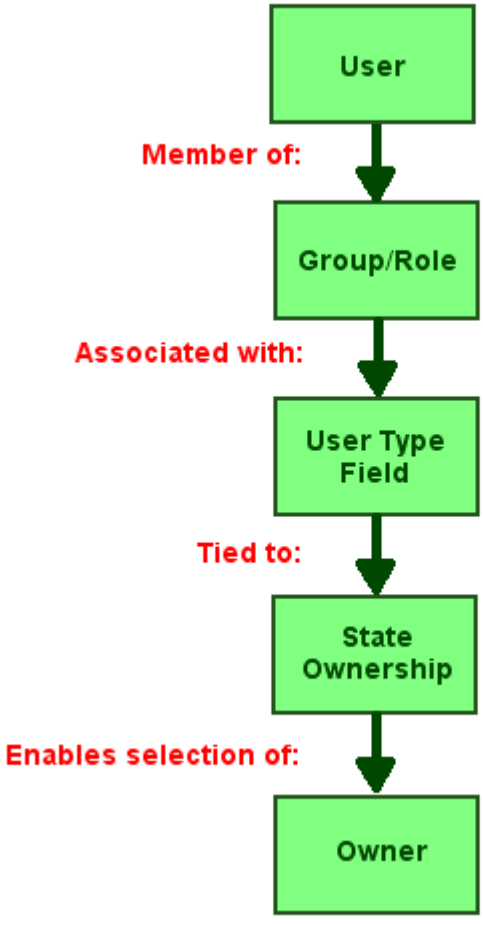
About Roles and Ownership

Workflows that you design in SBM Composer control how items move through states in the SBM User Workspace. Every state in a workflow must have one primary owner and can also have a secondary owner. The primary owner field must be a *User* field. The secondary owner field can be a *User*, *Multi-User*, or *Multi-Group* field. The fields can be populated with users who are assigned the role or roles associated with the field. You specify the roles that are associated with the fields in the Property Editor for the field in SBM Composer.

The selected owner is responsible for performing an action and then moving the item to the next state. For example, a new item in an issue tracking application starts in a Submit

state, and then moves to an Assigned state. The owner of the Assigned state could be someone with the Development Lead role. He or she assigns the item to someone with the Engineer role. The item moves through various states in the workflow until it is closed.

The following diagram illustrates the steps involved in the process of establishing the *ownership* [page 690] of items in an *application* [page 679].



Comparing Role Privileges and User/Group Privileges

Roles let a designer (working in SBM Composer) work with privileges abstractly—that is, in relation to a function (such as QA Manager) rather than actual people. In contrast, an administrator (working in SBM System Administrator) assigns privileges to actual users and groups of users. The administrator can assign privileges directly to users or groups or by associating a user or group with one or more of the roles created in SBM Composer.

The privileges available to the designer when creating or editing roles are all available to the administrator as user and group privileges, and the names are all close to identical (for example, the "View Fields in the 'User' Section" role privilege maps to the "View User Fields" user/group privilege). However, not all privileges available in SBM System Administrator are available as role privileges. For example, administrative privileges for modifying a user profile, folder privileges, and various system privileges are user/group privileges and are available only in SBM System Administrator.

Generally, role privileges alone are not enough for complex process app implementation, and you will likely want to combine roles with additional user/group privileges applied in SBM System Administrator.

Note: Privileges are cumulative. That is, SBM identifies a privilege as granted to a user if the privilege is associated with any of the following: the user, a group to which the user belongs, or a role assigned to the user.

Role privileges for primary tables are based on projects. The privileges can be categorized as follows:

- **Advanced** - View
- **All** - Submit, Own, Delete, View
- **Archived Items** - View, Restore
- **Attachments** - View, Add, Edit, Set, Delete
- **Change History** - View
- **Hidden** - View
- **Item** - View, Update, Transition, Mass Update
- **Manager** - View
- **Notes** - View, Add, Edit, Set, Delete
- **Principal** - Link/Unlink
- **Principal and Subtasks** - View
- **Reports** - Manage, Create, Modify, Run, Delete, Access, Create/Modify Advanced SQL Queries
- **Section** - View, Edit
- **State Change** - View
- **Subtasks** - Link
- **System** - View
- **User** - View
- **Version Control History** - Manage
- **Workflow** - View

Role privileges for auxiliary tables are not based on projects. The privileges can be categorized as follows:

- **Advanced** - View
- **All** - Submit, Update, Delete, View
- **Attachments** - View, Add, Edit, Set, Delete

- **Change History** - View
- **Hidden** - View
- **Item** - Mass Update
- **Manager** - View
- **Notes** - View, Add, Edit, Set, Delete
- **Reports** - Manage, Create, Modify, Run, Delete, Access, Create/Modify Advanced SQL Queries.
- **Section** - View, Edit
- **User** - View

Roles Editor

In the roles editor, you create and edit roles, giving each *role* [page 695] a name (such as QA Manager) and a description, and associating privileges with the role. You can also specify whether the role should be used globally.

Note: To edit an existing role (for example, to change its name), you must have the role checked out.

To create a new role, right-click the **Roles** heading in App Explorer, and select **Add New Role**. After you click **OK** to check out the required design elements, use the Property Editor to give the role a name, and enter a description for the new role.

Note: The description is optional, but it can be useful for clarifying to others (or to yourself at a later time) what functions are to be performed by users assigned to this role.

Select the **Global** check box if you want to import the role into any other applications you add to the process app. The role in the other *application* [page 679] has the same name and privileges. If this check box is selected for any role in any application in the process app, after you add another application to the process app, you are prompted whether you want to add the global roles from existing applications to the new application.

Note: You can change the name of the role in the new application and change its privileges.

The most important task to perform when you create a role is associating privileges with the role. Your selection of privileges determines what users with this role can do in the categories (such as items and fields) displayed in the **Privileges** section of the roles editor.

Note: The set of available privileges from which to select is different depending on whether you are creating a role for a *primary table* [page 692] or for an *auxiliary table* [page 680]. This is because the activities that you can perform for the two types of table are different.

Tip: To select all listed privileges at once, click **Check all**. To clear all the check boxes, click **Uncheck all**.

When you finish assigning a name, description, and set of privileges to a role; and specify whether the role is a global role, save your changes by clicking the **Save locally** icon, or click **Check In** on the **Repository** tab of the Ribbon to both save your changes locally and check them in to the SBM Application Repository.

Chapter 9: Managing Tables

Tables in the SBM database contain fields that hold all the information related to SBM applications. Two types of tables support SBM applications: primary tables and auxiliary tables.

This section contains the following information:

Overview of Tables

The core of the SBM infrastructure is the SBM database. The database consists of two types of tables: primary tables and auxiliary tables. Both types hold records of information, called *items*. Primary tables store primary items, which are routed through workflows (sets of rules about how primary items progress through a tracking system). Auxiliary tables store auxiliary items, typically information related to primary items. Auxiliary items support the progression of tracked items, but they do not follow a workflow.

Examples of auxiliary items include company and contact records. The SBM database includes system auxiliary tables (`Companies`, `Contacts`, `Problems`, `Resolutions`, and others).

About Primary Tables

The *primary table* [page 692] is the foundation of a SBM *application* [page 679]. Each application has one primary table in the SBM database. The primary table is created automatically when you add an application to a process app.

A primary table stores a record for each item that follows the corresponding workflow. Primary tables contain a combination of required system fields, optional system fields, and *custom fields* [page 682]. These fields are used to collect data as the *primary item* [page 691] progresses through the workflow. Workflows depend on the primary table to hold the fields and actual records created by progress through the workflow.

When SBM User Workspace users with the proper privileges submit, transition, update, and perform other operations on primary items, they update the corresponding records in the primary table.

About Auxiliary Tables

Auxiliary tables store items that support, but do not follow, a workflow process. Auxiliary items typically contain information that is collected once and used repeatedly. Auxiliary

tables allow this information to be stored separately from primary items, making it easily shared and reusable across multiple applications.

Because auxiliary tables typically store static information, they are often used to create relationships between tables. *Relational fields* [page 694] can be added to any *auxiliary table* [page 680] or *primary table* [page 692] to reference information contained in auxiliary tables. This is the basis for field dependencies, which let you populate selection lists in a selection or relational field type, based on selections in another field. See Using Field Dependencies [page 168] for details.

About Creating Tables

You create primary and auxiliary tables for your applications using SBM Composer. You add fields to the *primary table* [page 692] to collect information about primary items. You add fields to auxiliary tables to collect auxiliary information. SBM Composer provides an extensive set of system and custom *field types* [page 686] for the purpose of collecting information for primary and auxiliary items.

Before configuring a primary table, carefully consider its purpose, what types of information you want to display in state forms and to whom, and what data you want to collect in transition forms and from whom.

New tables you add to the *application* [page 679] are defined as auxiliary tables and typically contain information that supports the items in the primary tables. Before configuring an *auxiliary table* [page 680], decide what type of auxiliary information is important to include in the application to support the workflow.

Creating a Primary Table

Each *primary table* [page 692] is associated with an *application* [page 679]. When you add an application to your *process app* [page 692] in SBM Composer, the associated primary table is created automatically.

Use the table editor to modify the table, defining its properties and adding fields as needed. To modify fields, select them in the table editor and then make changes in the Property Editor.

Creating an Auxiliary Table

Unlike primary tables, which are created automatically when you add an *application* [page 679] to your *process app* [page 692], you create auxiliary tables as you need them.

You typically add auxiliary tables to existing applications, which already contain one *primary table* [page 692]. Auxiliary tables are added to applications when a need for auxiliary information is anticipated.

In the table editor you can modify, delete, and create system fields and *custom fields* [page 682] for auxiliary tables.

To create an *auxiliary table* [page 680]:

1. In App Explorer, in the application to which you want to add the auxiliary table, right-click the **Tables** heading (or right-click an existing table), and select **Add New Table**. A new auxiliary table is added to the **Tables** heading, and it opens in the table editor. The auxiliary table automatically includes the required *Title* field, which you can rename but cannot delete.

2. Click in the heading area (above the list of fields) to select the entire table, or select a field in the list to display the appropriate Property Editor. See Modifying Table Properties [page 116] for details.

   Note: If the Property Editor is not visible, you can open it by selecting **Property Editor** in the **Common Views** area on the **Home** tab of the Ribbon.

3. Drag fields from the **Table Palette** to the table editor, or select and delete fields as needed.

Copying a Table

You can base a new table on the structure of an existing *auxiliary table* [page 680]. The new table will also retain certain table settings, such as record locking.

The following considerations apply:

- The new table does not include items (records) from the original table.
- Fields and field attributes cannot be modified during a copy.
- *Relational fields* [page 694] that refer to the original table will refer to the new table. For example, if you copy the `Issues` system auxiliary table, and it has an *Issues Single-Relational* field, the field in the new table will point to itself, not to the original table.
- Fields that are deleted in the original table are not created in the new table. All non-deleted original fields are available for editing in the new table.
- You cannot copy a table that is checked out to another *designer* [page 683].

To copy a table, right-click the table in App Explorer (or in the list that is displayed when you select the **Tables** heading), and select **Duplicate**.

Table Editor

Use the table editor to design primary and auxiliary tables. It is displayed when you select a table in App Explorer, or when you double-click a table in the list that is displayed when you select the **Tables** heading in App Explorer. Use the tabs in the Property Editor to view and modify the various aspects of the selected table.

- General Tab of the Table Property Editor [page 117]
- Options Tab of the Table Property Editor [page 119]
- Labels Tab of the Table Property Editor [page 120]
- Icons Tab of the Table Property Editor [page 120]
- Forms Tab of the Table Property Editor [page 121]
- Dependencies Tab of the Table Property Editor [page 121]

Palette

A palette of *field types* [page 686] is displayed to the right of the table editor. Add fields to the table by dragging them from the **Table Palette** to the table editor. Each of the

system fields (*Active/Inactive*, *Description*, and so on) can be used only once in a table. The other (custom) field types can be used as often as needed.

Tip: You can also add a field by right-clicking the table editor, selecting **Add New**, and selecting the field type.

Delete a field by right-clicking it in the table editor and selecting **Delete** or **Cut**. When you delete or cut a system field type, it is returned to the list to make it available for use.

Note: You cannot delete a required *system field* [page 699] (such as the *Title* field in an *auxiliary table* [page 680]).

Cutting a field also copies it to the Windows Clipboard, so you can paste it into the same or other tables as needed.

Note: You cannot copy any of the system fields.

Modifying Table Properties

To modify table properties:

1. Select the table in App Explorer (or click the **Tables** heading, and then select the table from the list).

   The Property Editor below the table editor displays information about the selected table.

   Tip: If the Property Editor is not visible, select **Property Editor** in the **Common Views** area on the **Home** tab of the Ribbon.

2. View and modify information about the selected table using the tabs of the table Property Editor.

   - General Tab of the Table Property Editor [page 117]
   - Options Tab of the Table Property Editor [page 119]
   - Labels Tab of the Table Property Editor [page 120]
   - Icons Tab of the Table Property Editor [page 120]
   - Forms Tab of the Table Property Editor [page 121]
   - Dependencies Tab of the Table Property Editor [page 121]

Note: If you select a field in the table, the field Property Editor replaces the table Property Editor.

General Tab of the Table Property Editor

Use this tab to specify general information about the table.

Element	Description
Logical name	The name that describes the contents of the table. The logical name is typically derived from the plural form of the items in the table. For example, if each item in the table represents a customer, the logical name for the table should be `Customers`. You can modify the logical name of an existing table.  **Tip:** The name that you type in this field automatically populates the **Database table name**, which must be unique within the SBM database and is limited to 24 Unicode characters. You should limit the logical name to 24 characters, as additional characters will be left off of the **Database table name**.
Singular item name	The name that describes a single item in the table. For example, if each item in the table represents a customer, the singular item name should be `Customer`. SBM Composer automatically fills this field as you type the logical table name. You can modify the singular item name. However, for best results, it should be unique with the SBM database.
Edit tab name link	Shows the *application* [page 679] editor, where you can change the string your users see in the SBM User Workspace. When you finish, click  (in the Quick Access tool bar) to return to the current view of the table editor and the **General** tab of the Property Editor.
Database table name	The unique internal database name for this table. SBM Composer automatically fills this field as you type the logical table name, ignoring characters after the first 24.  **Note:** The database name can contain only ASCII alphanumeric characters (A–Z, a–z, 0–9) and underscores. Any other characters are ignored as you type them. You cannot change the database table name after the table is created.

Element	Description
Value display format	The format that controls how items in the table appear as values to users in the SBM User Workspace. For example, you can select fields from the table to be displayed when an item is used in a list or as a link. You can determine the order in which the fields appear, add text to the format, and use HTML tags to format the content. For example, if you specify a value display format of `<Title> - <b><Owner></b>` then items from the table look something like this in search results: ITEM10098 This is the title of the item - **Item's owner**  **Note:** By default, any HTML tags included in the field are delivered for rendering by the Web browser in the SBM User Workspace. Guidelines for the value display format follow: • The value display format you specify is limited to 255 characters. • By default, the value display format for a new table is <Title>. • Relational field values are based on the value display format. For example, the value display format for the `Companies` table is <Customer Name>. You can add or modify supported fields to the display. • HTML tags are ignored in the selection lists presented to users. • Fields located in the **Hidden Fields** section can be used, and values are displayed to users in all areas where the value display format is used for the table.

Element	Description
	• The value display format determines the fields that are searched and the order in which values are searched for *relational fields* [page 694] for the *Value Find* [page 700], Lookup form, and *Relational Field Value Lookup* [page 694] form in the SBM User Workspace. For example, if you add the *Item ID* and *Description* fields as the value display format for a *primary table* [page 692], users can search for values by item ID or keyword for relational fields that reference the table.  **Note:** The display order of the fields affects how the Value Find feature operates. Value Find demarcates the entered values based on spaces, and then searches for the first value in the first column, the second value in the second column, and so on. For example, if the user searches for `manager doe` and the value display format is <FullName> - <Position>, the Value Find results will be **None**. In addition, if a user searches for `john doe` with the value display format of <FullName> - <Position>, the Value Find results will be **None**. This is because Value Find looks for a <FullName> containing *john* and a <Position> containing *doe*. Users must use quotation marks to treat the value as single entity when searching each column. For example, entering `"john doe"` in Value Find will yield the results *John Doe - Manager*. • *Relational* field values based on auxiliary tables are sorted by the table's value display format. To change the order in which values sort for the *Relational* field, modify the value display format for the *auxiliary table* [page 680]. • Fields specified in the value display format always appear in the Item List frame for search results in the SBM User Workspace. When users run Listing reports, fields specified in the value display format are displayed if users have not selected fields to display in the report.
Description	Optional comment or note about the purpose of this table.

Options Tab of the Table Property Editor

Use this tab to organize search results for the Advanced Lookup Tool in the SBM User Workspace.

For example, in an Incidents table, you could select the *Company* and *Contact* fields. When a user performs a search, the results are organized according to relationships among incidents, companies, and contacts.

If you do not need these settings for a specific purpose, simply select **(None)**.

Labels Tab of the Table Property Editor

Use this tab to rename the field section titles, item details labels, and report privilege category labels for the selected table. The changes made to these labels are displayed to your users in areas related to the table in the SBM User Workspace.

For example, if you rename the report privilege categories on this tab for an *auxiliary table* [page 680], the new names are displayed in the Privilege Category drop-down list on the Save Report forms for this table in the SBM User Workspace.

The **Default** column shows the text that will be displayed for each item. You can type a replacement in the **Override** column next to any label you want to change.

Element	Description
Sections	The labels to apply to *field sections* [page 686] for the table. These labels are displayed in all areas of the SBM User Workspace related to the table, and in the **Privilege section** drop-down list on the **General** tab of the Property Editor for fields.
Item details	The labels to apply to the listed sections for the table. These labels are displayed in all areas of the SBM User Workspace related to the table, and in the **Privilege section** drop-down list on the **General** tab of the Property Editor for fields.
Report levels	The labels to apply to the listed report privilege categories for the table. These labels are displayed in all areas of the SBM User Workspace related to the table, and in the **Privilege section** drop-down list on the **General** tab of the Property Editor for fields.

Icons Tab of the Table Property Editor

Use this tab to customize icons displayed in the SBM User Workspace for the various *field types* [page 686].

For each field type listed, you can override the default by selecting an icon already loaded into SBM Composer, or by selecting **Add** to load in and select a new icon. The new icon is also added to the **Images** heading in App Explorer, so it is available for selection as needed.

Element	Description
Relational and Multi-Relational Fields	These icons are displayed when the table is used as a *Single Relational* or *Multi-Relational* field in another table. For example, if you create a *Single Relational* field in an `Issues` table that is associated to an `Incidents` table, the icon specified here for the `Incidents` table is displayed when users work with issues in the SBM User Workspace.

Forms Tab of the Table Property Editor

For each form type, click **New** to create a custom form for the selected *auxiliary table* [page 680]. Click **Preview** to see a representative rendering of the form as it will appear to users in the SBM User Workspace. Click **Edit** to modify the existing form for the table.

Note: This tab is available for auxiliary tables only. Primary tables use the forms you set up for their corresponding workflows, states, and transitions. See Forms Properties of an Application Workflow [page 280], Form Properties for a State [page 297], and Form Properties for a Transition [page 319] for details.

Element	Description
View Form	The read-only form used to display items in the table.
Edit Form	The read-write form used to edit items in the table.

Field Privileges Tab of the Table Property Editor

Use this tab to modify the order and privileges of fields in an *auxiliary table* [page 680].

Element	Description
List of fields	The fields in this list are divided into sections, such as **Standard** and **Advanced**. • To change the order of fields within a section, drag and drop the fields, or right-click the field and then select **Move up** or **Move down**. • To change the privileges of a field, drag and drop the field between sections, or right-click the field and then select **Move up** or **Move down**.
Fields in the *Section Name* Section	When you select a field in the list of fields, this section displays the privileges for that field. For example, it could indicate that fields in the Advanced section can be changed by the **Administrator** and **Manager** roles, can be read by the **Developer** *role* [page 695], and are not visible to the **User** role.

Dependencies Tab of the Table Property Editor

This tab is available for auxiliary tables only.

Element	Description
Select independent field	Select the field for which you want to define a dependency.
Restrict dependent field... To these values	When a user selects a value in the specified independent field, the dependent field you specify here is automatically limited to the values you specify here.

Element	Description
Change value to	Define a specific dependent field default value for each independent field value.

Establishing Relationships Between Tables

You can establish relationships between tables in SBM Composer using *Single-Relational* and *Multi-Relational* fields. SBM Composer also provides *Sub-Relational* fields to retrieve supplementary information from a related table. All primary and auxiliary tables in your process app can be used to define these relationships.

To use tables from applications from another process app (such as the system auxiliary tables in the *Global Process App* [page 687]), define a dependency on that application in the current process app. See About References [page 361] for details.

Chapter 10: Working With Fields

Data collection is the cornerstone of many process apps, which depend on accurate and complete data about primary and secondary items. Compiling relevant historical data into management reports also depends on effective data collection. Fields are used to gather data as users submit, transition, and update primary items and work with auxiliary items. SBM Composer provides many types of fields, most of which have sets of properties that define the behavior of the fields. These fields can be customized to meet the data collection requirements of the applications deployed to SBM.

SBM Composer provides a set of required system fields in primary and auxiliary tables. These fields often satisfy data collection needs for primary and secondary items, but you can also add fields as needed for your process app.

SBM Composer also gives you the flexibility to set field properties to make it easy for your users to enter information about items. You can modify (override) field properties established in the *primary table* [page 692] at different levels of the *application* [page 679]—workflows, states, and transitions.

This section contains the following information:

How Fields are Used in the SBM User Workspace

As users submit, update, and transition primary items into projects in the SBM User Workspace, they fill in fields on forms, typically providing information for tracking and management purposes.

For example, when a *primary item* [page 691] is submitted, a manager could review the item and assign it to the appropriate owner. The manager fills in the appropriate fields on the form, such as to whom the item is assigned. When the item is resolved, the current owner fills in fields describing the resolution. The item is then transitioned to the next responsible party, who fills in the fields as needed.

Fields are also used to gather data for items in auxiliary tables, which support a workflow process. Typically, fields in an auxiliary table store static data, such as customer information. To help support your workflow process, you can use Relational fields to pull this static data into primary items.

Organizing Fields

Each field is assigned to a privilege section. Privilege sections determine which roles can view and modify fields on forms in the SBM User Workspace. For example, you could set up the *Owner*, *Project*, and *State* fields to be accessible only to users assigned to the Manager role.

On a *quick form* [page 693], fields are gathered automatically according to their privilege sections. On a custom form, privilege sections control accessibility of fields, but you can arrange them any way you want. See Quick Forms and Custom Forms [page 196] for details.

You can rename *field sections* [page 686] on the **Labels** tab and **Icons** tab of the table Property Editor. See Labels Tab of the Table Property Editor [page 120] and Icons Tab of the Table Property Editor [page 120] for details.

Each field type is associated with a default privilege section, to which it is assigned when you drag it onto the table editor. If you select the field, you can change its privilege section on the **General** tab of the field Property Editor.

By default, field definitions (including the privilege section) are inherited by associated workflows and passed on to sub-workflows. You can override field definitions in these workflows and in any state or transition associated with them.

SBM provides several privilege sections. The following list includes system fields for primary and auxiliary tables as examples of the typical use of the various privilege sections.

- **Standard**: This privilege section is the default assignment for the *Description*, *Item ID*, *Item Type* [page 688], and *Title* system fields.

  Tip: On a quick form, the Standard section is at the top of the form, so you could place required fields in this section to help ensure that users provide a value.

- **User**: This privilege section is the default assignment for the *Last State Changer*, *Resolution Description*, and *Resolution Summary* system fields.

- **Advanced**: This privilege section is the default assignment for the *Close Date*, *Last State Change Date*, *Submit Date*, and *Submitter* system fields.

- **Manager**: This privilege section is the default assignment for the *Active/Inactive*, *Owner*, *Project*, *Secondary Owner*, and *State* system fields.

  Tip: It is generally advisable to prevent access to the *Owner*, *Secondary Owner*, *Project*, and *State* fields by most users. These fields are generally handled automatically by the system. Therefore, most users should not have access to change the field values. (There could be situations in which a manager needs to change the values in these fields manually.)

- **System**: This privilege section is the default assignment for the *Last Modified Date*, *Last Modifier*, and *Last State Changer* system fields.

- **Hidden**: This privilege section provides a place to store fields that you do not want users to view in a particular workflow, state, or transition.

System Fields

All primary and auxiliary tables contain system fields that help gather data needed by the tracking system. Some system fields are required, but others are optional.

- System Fields for Primary Tables [page 125]
- System Fields for Auxiliary Tables [page 128]

System Fields for Primary Tables

System fields for primary tables help ensure that the process you define is followed and that correct data is gathered as items move through that process. System fields also provide greater reporting and searching benefits than *custom fields* [page 682]. For example, only system fields are available on Multi-table reports, which let users search for information across multiple primary and auxiliary tables.

Primary table fields are created in SBM Composer. However, you can configure specific field settings for projects in SBM System Administrator.

Two types of system fields are provided for primary tables: those that are required and those that are optional.

For details, refer to:

- Required System Fields for All Primary Tables [page 125]
- Optional System Fields for All Primary Tables [page 127]

Required System Fields for All Primary Tables

Required system fields for primary tables gather necessary data for your tracking system and help ensure that the process you define is followed. Required system fields are automatically added to all primary tables. These fields cannot be deleted or moved into the Not Used section. You can, however, move them into the Hidden field section. You can edit these fields and change certain property settings and move them to different *field sections* [page 686].

Note: To ensure data integrity for fields that are auto-populated by the system, such as the *Owner* field, consider setting them as read-only to prevent users from changing the values. You can also move these fields into the Hidden field section.

The following table describes the system fields required for primary tables. The field icon is provided, along with information and special considerations for each *system field* [page 699].

Icon	Field Name	Notes
	Active/ Inactive	A *Binary/Trinary* field that indicates a primary item's status. States use the *Active/Inactive* field to automatically determine the status of items in each state. In SBM Composer, you can change the active and inactive labels. The *Active/Inactive* field is always displayed as a drop-down list in the SBM User Workspace.

Icon	Field Name	Notes
	Item ID	An auto-populated *Text* field that numbers primary items based on project settings defined in SBM System Administrator. The *Item ID* is used to identify and locate items in the ID Search feature, SourceBridge, reports, and more. If your *Item Type* [page 688] field uses prefixes, they are prefixed to the *Item ID*.
	Item Type	A *Single Selection* field populated with the types of items you wish to track, such as defects and enhancement requests. Specify a three-letter prefix that is prefixed to the Item ID. For example, a defect might have an Item ID of DEF00011 and an enhancement might have an Item ID of ENH00025. You can tailor your process to manage each item type differently, if needed. For example, you can modify a workflow by restricting the item types available for certain transitions. **Note** By default, the Item *Type* field is not set as required. For best results, set the field to be required as items are added to the system.
	Last Modified Date	An auto-populated *Date/Time* field that indicates the date and time an item was last updated. This information is used for auditing and establishing *change history* [page 681] for items.
	Owner	An auto-populated *User* field indicating the current *primary owner* [page 691] of an item. The value of the owner field is determined by the state in which an item resides and changes as items are transitioned through a workflow. In SBM Composer, you select a User field type as the value for the Owner field for each state in the General tab of the state Property Editor.
	Project	A system *Project* field populated with all projects in an *application* [page 679]. If users change the value of the *Project* field for a specific item, they are also moving that item to the specified project. Depending on your workflow and project relationships, the item could follow a different process than you intended. To avoid this problem, set the field to read-only or move it to the Hidden section.  **Note:** Projects are defined in SBM System Administrator.
	State	A system *State* field populated with all states for a specific workflow. If users change the value of the *State* field for a specific item, they are also moving that item to the specified state in the workflow and it could follow a different process than you intended. To avoid this problem, set the field to read-only or move it to the Hidden field section or restrict the section according to privileges.

Icon	Field Name	Notes
	Submitter	An auto-populated *User* field indicating the user who submitted an item. Users whose privileges include submitting items into the system are available as values for this field in the SBM User Workspace.
	Title	An 80-character fixed-length *Text* field. The system *Title* field is displayed by default in many areas, such as built-in report and search results. This is the optimal display length for displaying item titles in the SBM User Workspace. You can increase the character length of the system *Title* field. However, doing so could return unexpected results.

Optional System Fields for All Primary Tables

Optional system fields for primary tables lets you collect additional data to enhance your tracking system. Create these fields when setting up the *primary table* [page 692] for your *application* [page 679] in SBM Composer.

The following table describes the optional system fields that are available for primary tables.

Field Name	Notes
Close Date	An auto-populated *Date/Time* field that records the date and time items are transitioned to an inactive state. If the item is later moved to an active state, the *Close Date* value is cleared. If you do not add the *Close Date* field to your application, the close date and time are still captured in various areas, such as the *state change history* [page 697] and the timestamp for *Text/Journal* fields. Adding the *Close Date* field to your system lets users query for this information in search features and reports, particularly State Change reports.
Description	By default, the system *Description* field is a *Text Memo* field that is set to allow keyword searching.
Last Modifier	An auto-populated *User* field that records which user last changed an item. The value of the *Last Modifier* field changes as items are updated and transitioned through a workflow. All users who can update and transition items into the system are available as values for this field, but you can control who updates and transitions items through privileges. If you do not add the *Last Modifier* field to your primary table, the name of the user who last modified an item is still recorded in the *change history* [page 681]. Adding the *Last Modifier* field to your primary table lets users query for this information in search features and reports.

Field Name	Notes
Last State Change Date	An auto-populated *Date/Time* field that records the date and time an item last moved to another state. The value of the *Last State Change Date* field changes as items are transitioned to different states. The *Last State Change Date* field is useful for searching and reporting on items based on the last date and time they were last moved. To view a complete history of state changes for particular items, users can create State change reports.
Last State Changer	An auto-populated *User* field that records the user who last moved an item to a new state. The value of the *Last State Changer* field changes as items are transitioned through a workflow. All users who can transition items into the system are available as values for this field, but you can control who transitions items through privileges and transition restrictions. If you do not add the *Last State Changer* field to your application, the Change History section shows who moved items from state to state. Adding the *Last State Changer* field to your application lets users query for this information in search features and reports.
Secondary Owner	A *User* type field configured in that lets users select one or more users or groups as secondary owners of an item. After you add the *Secondary Owner* field to a primary table, you can then specify the *User*, *Multi-User*, or *Multi-Group* field that determines secondary *ownership* [page 690] for each state.
Submit Date	An auto-populated *Date/Time* field that records the date and time items were submitted. If you do not add the *Submit Date* field to your application, the submit date and time are still captured in various areas, such as the State Change History and the timestamp for *Text/Journal* fields. Adding the *Submit Date* field to your application lets users query for this information in search features and reports, particularly Trend, Backlog Trend, and State Change reports.

System Fields for Auxiliary Tables

System fields for auxiliary tables help ensure that correct data is gathered for items that may support a workflow process. System fields also provide greater reporting and searching benefits than *custom fields* [page 682]. For example, only system fields are available on Multi-Table reports, which let users search for information across applications and auxiliary tables.

At a minimum, auxiliary tables must contain the *Title system field* [page 699]. This field, which is automatically added to all auxiliary tables, typically serves as the value display format setting for the table, but you can use a different field for this setting as needed.

Auxiliary table fields are created in SBM Composer. You can configure specific field settings for auxiliary tables in SBM System Administrator, however.

The following table lists the optional system fields available for auxiliary tables.

Field Name	Description
Active/ Inactive	The *Active/Inactive Binary/Trinary* field for auxiliary items indicates the status of the item and lets users enable and disable items in auxiliary tables. You can change the active and inactive labels for this field, but it always appears as a drop-down list in the SBM User Workspace. You can use the *Active/Inactive* field in auxiliary tables to limit the selections available in *Relational field types* [page 686] without deleting items in auxiliary tables. For example, you can add the *Active/Inactive* field to the *Companies* table, letting users disable companies that are no longer needed without deleting the data associated with those companies. The inactive companies are not available as new values for the *Company* field, but they are available as selections made when they were active. In addition, users can still access them in reports and searches.
Description	A *Text/Memo* field that is set to allow keyword searching.
Item ID	An auto-populated *Text* field that numbers auxiliary items based on the order in which items are added to the table. The *Item ID* is used to identify and locate items.
Last Modified Date	An auto-populated *Date/Time* field that indicates the date and time an *auxiliary item* [page 680] was last updated. This information is used for auditing and establishing *change history* [page 681] for items.
Last Modifier	An auto-populated *User* field that records which user last changed an auxiliary item. The value of the *Last Modifier* field changes as items are updated. All users who can update items into the system are available as values for this field, but you can control who updates auxiliary items through privileges.
Submit Date	An auto-populated *Date/Time* field that records the date and time auxiliary items were submitted.
Submitter	An auto-populated *User* field indicating the user who submitted an auxiliary item. All users who can submit items into the system are available as values for this field, but you can control who submits items through privileges.
Title	A *Text* field that serves as the system title field and the default for the value display format for this table. By default, a *Title* field is automatically added to custom auxiliary tables. This field cannot be deleted.

System Fields for System Auxiliary Tables

Seven system auxiliary tables are provided with SBM, and system fields listed in the following tables are provided with each of these tables.

Companies Table

Field Name	Notes
Company Name	A *Text* field that serves as the system title field for this table and the default for the value display format for this table.
Customer ID	A *Text* field that can store additional identifying information for a company record.
Primary Contact and Secondary Contact	*Single Relational* fields that let users associate primary and secondary contacts for companies. These fields provide special system behavior.

Contacts Table

Field Name	Notes
Company	A *Single Relational* field based on the *Companies* table that lets users select a company to associate with a contact. This field provides special system behavior.
E-mail	A *Text* field that lets users provide an e-mail address for contacts. When you create *Contact* records from user accounts, the *E-mail* field in the *Contacts* table is populated with the e-mail address provided with the user account.
First Name	A *Text* field that by default is used as part of the value display format for this table. When you create *Contact* records from user accounts, the *First Name* field in the *Contacts* table is populated with the first word provided in the Name box for the user account.
Last Name	A *Text* field that serves as the system title field for this table and by default, is used as part of the value display format. When you create *Contact* records from user accounts, the *Last Name* field in the *Contacts* table is populated with the last word provided for the user account.
Middle Name	A *Text* field that by default is used as part of the value display format for this table. When you create *Contact* records from user accounts, the *Middle Name* field in the *Contacts* table is populated with the words between the first and last words provided for the user account.
Phone Number	A *Text* field that lets users provide phone numbers for contacts. When you create *Contact* records from user accounts, the *Phone Number* field in the *Contacts* table is populated with the phone number provided for the user account. The *Phone Number* field is optional.

Field Name	Notes
User Name	A system *User* field that is automatically populated with a SBM user name when a *Contact* record is created for that user. The option to create a *Contact* record for users is on the General tab of the Add/Edit User dialog box. This field is set to read only by the system and this property cannot be modified.

Languages Table

Field Name	Notes
Active/ Inactive	A *Binary/Trinary* field that indicates the active or inactive status of the language.
Language	A *Text* field that stores the names of languages the user interface has been translated into. The *Language* field is the system Title field and the default for the value display format for this table.
Locale	A *Text* field that stores the ISO 639-1 locale string for the language.

Problems Table

Field Name	Notes
Description	A *Text* field that lets users provide a description for problems. The *Description* field is provided with the *Problems* table by default, but is optional.
Folder	A *Folder* field that lets users create links to problems in folders. *Knowledge Base* [page 688] folders are designed to allow anonymous access to items, but users can also create Problem links in public or favorite folders. The *Folder* field is optional, but using it can help users better organize records in the *Problems* table.
Title	A *Text* field that serves as the system title field and the default for the value display format for this table.
Visibility	A *Binary/Trinary* field that lets users set the visibility of a problem to internal or public. Users who are granted the "View Public Problems and Resolutions" privilege can view items with public visibility. Anonymous users can also view items with public visibility if they are stored in Knowledge Base folders that allow anonymous access. By default, all new problems have internal visibility.

Field Name	Notes
Last Updated	An auto-populated *Date/Time* field that indicates the date and time a problem was last updated. This information is used for auditing and establishing *change history* [page 681] for problems. The *Last Updated* field is provided with the *Problems* table by default, but is optional.

Resolutions Table

Field Name	Notes
Description	A *Text* field that lets users provide a description for resolutions. The *Description* field is provided with the *Resolutions* table by default, but is optional.
Last Updated	An auto-populated *Date/Time* field that indicates the date and time a resolution was last updated. This information is used for auditing and establishing *change history* [page 681] for resolutions. The *Last Updated* field is provided with the *Resolutions* table by default, but is optional.
Problem	A *Single Relational* field based on the *Problems* table and lets users select a problem to associate with a resolution.
Title	A *Text* field that serves as the system title field and the default for the value display format for this table.
Visibility	A *Binary/Trinary* field that lets users set the visibility of a resolution to internal or public. Users who are granted the "View Public Problems and Resolutions" privilege can view items with public visibility. Anonymous users can also view resolutions with public visibility if they are associated with problems that are stored in Knowledge Base folders that allow anonymous access. By default, all new resolutions have internal visibility.

SharePoint Project Servers Table

This table contains relationship information between a SharePoint® site and an SBM project. It is populated automatically and must not be populated manually.

SharePoint Servers Table

Field Name	Notes
Title	Identifies what this SharePoint® server is used for, such as "Sales" or "Human Resources."

Field Name	Notes
Site Url	Contains the Site Collection root or high-level SharePoint folder under which SharePoint sites will be created. For example, `http://SharePointServerName:port/sites/ImageBuilder`
User Name	The account that will be used to create the folders in SharePoint. This account needs permissions in SharePoint to add folders and add, edit, and delete documents. This account is also used when attachment actions are performed in SBM by an anonymous user.
Password	The password used to access the account. This password is stored in the database, so it should be configured to not expire.

String IDs Table

Field Name	Notes
Name	A *Text* field that stores string representations of unique resource identifiers associated with strings. The *Name* field serves as the system title field and the default for the value display format for this table.
Root Value	Stores the user-visible value for the string. Root values should not be modified.

Strings Table

Field Name	Notes
Create Date	An auto-populated *Date/Time* field that records the date and time strings were submitted. The *Create Date* field is provided with the *Strings* table by default, but is optional.
Creator	An auto-populated *User* field indicating the user who submitted a string. All users who can submit items into the system are available as values for this field, but you can control who submits strings through privileges. To ensure data integrity for *Creator* field values, set this field to read-only or move it into the Hidden Fields section.
Description	A *Text* field that lets users provide a description for strings.
Language ID	A *Single Relational* field associated with the *Languages* table that lets you associate a translated string with a language.

Field Name	Notes
Last Modified Date	An auto-populated *Date/Time* field that indicates the date and time a string was last updated. This information is used for auditing and establishing change history for items. The *Last Modified Date* field is provided with the *Strings* table by default, but is optional.
Last Modifier	An auto-populated *User* field that records which user last changed a string. The value of the *Last Modifier* field changes as strings are updated. All users who can update items into the system are available as values for this field, but you can control who updates strings through privileges. To ensure data integrity for *Last Modifier* field values, set this field to read-only or move it into the **Hidden Fields** section. The *Last Modifier* field is provided with the *Strings* table by default, but is optional.
String	A *Text* field that stores strings that are displayed in SBM User Workspace elements, such as labels, button names, and error messages.
String ID	A *Single Relational* field associated with the *String IDs* table that lets you associate a unique resource identifier, referred to as an internal ID, with a string. The *String ID* field is the default for the value display format for this table.

System Behavior for Contacts and Companies Fields

The *Companies* and *Contacts* system tables are included in every environment. These system tables let you store information about companies you work with and contacts that are associated with those companies.

In addition, system *Contacts* and *Companies* fields can be defined in each application. Special behavior is provided with these system fields. For example, you can set up your system so that an existing user can grant External access to a contact. You can also grant privileges that let contacts view items submitted by other contacts within in the same company.

This behavior is only available for the system *Contacts* or *Companies* fields. For example, if you create a *Single Relational Contacts* field in an *application* [page 679], the system behavior does not apply.

Custom Fields

SBM Composer provides a diverse set of *field types* [page 686] that you can add to the primary and auxiliary tables in your process apps. See Custom Field Types [page 135] for a complete list.

Consider the following tips when you create custom fields:

- *Binary/Trinary*, *Date/Time*, *Numeric*, *Text*, and *Summation* fields represent simple values stored within database tables. These field types are the most efficient and provide the best performance when used in reports. In the SBM User Workspace, users can search against and sort most of these field types in reports. Report calculations can also be performed on *Date/Time* and *Numeric* fields.

- *Text* fields with the **Memo** or **Journal** option enabled cannot be sorted in reports.
- SBM supports non-ASCII characters in *Text* field values, allowing data in the SBM User Workspace to be displayed in a wide variety of international languages. For details, refer to the *SBM System Administrator Guide*.
- *Folder*, *Single Selection*, *Multi-Selection*, *Multi-User*, *Multi-Group*, and *User* fields are limited to the selections set by the *designer* [page 683] or administrator. *Single Relational*, *Multi-Relational*, and *Sub-Relational* fields are constrained by data provided by users.
- With a *Single Relational* field, users can select a value from a drop-down list or search for a value using the *value find* [page 700] or relational field value search feature. A *Multi-Relational* field adds the ability to select multiple values (rather than a single value) and the option to use a set of check boxes (rather than a drop-down list). In either case, users who have privileges to view the relational field item in the related table can click the pop-up icon that appears next to the relational field when they view items in the SBM User Workspace. The related item opens in a pop-up window, and users can work with the item as their privileges allow.
- The *Item ID* and value display format are displayed as values for *Single Relational* and *Multi-Relational* fields. If an *auxiliary table* [page 680] used for *relational fields* [page 694] does not contain an *Item ID* field, only the value display format is displayed. See General Tab of the Table Property Editor [page 117] for information about the value display format.
- Consider user privileges carefully when you create relational fields across tables. Users who do not have privileges to view items in a relational field table may be able to view those items as values in relational fields in other tables. To prevent users from viewing relational field values, move relational fields to a section for which users do not have view privileges.
- *Sub-Relational* fields are not added to the database, but define a relationship between fields. *Sub-Relational* fields can be useful in certain situations, but extensive use of them may slow the performance of your process app in the SBM User Workspace.

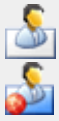
Custom Field Types

SBM provides a set of *field types* [page 686] that you can use to create your own *custom fields* [page 682] for data collection. Custom fields can be added to primary and auxiliary tables.

The following table lists available field types.

Icon	Field Type	Description
	*Binary/ Trinary*	*Binary/Trinary* fields can store either two values or three values of your choosing. *Binary* fields can be displayed on forms as a drop-down list containing two values, as radio buttons that let users select one value or the other, or as a check box that users select or clear as needed. *Trinary* fields can be displayed as a drop-down list containing three values or as radio buttons that let users select one of the three values when submitting, updating, or transitioning items.
	*Date/Time*	*Date/Time* fields store dates and time values. *Date/Time* fields can be displayed as the date only, date and time, time only, or elapsed time. If you select the Elapsed Time option, you can also select the Stopwatch, Calculate Days, and Show Seconds options.
	*Folder*	*Folder* fields enable users to link a *primary item* [page 691] or *auxiliary item* [page 680] to a specified Public or *Knowledge Base* [page 688] folder. If you enable searching for *Folder* fields, *Value Find* [page 700] is enabled.
	*Multi-Group*	*Multi-Group* fields let users select one or more groups to associate with an item. Users can select values from a drop-down list or set of check boxes. You can also use *Multi-Group* fields to establish secondary *ownership* [page 690] for primary items by selecting the field on the General tab in the Property Editor for states. When you enable searching for *Multi-Group* fields, Value Find is enabled.
	*Multi-Relational*	*Multi-Relational* fields let users associate one or more values with a particular *primary table* [page 692] or *auxiliary table* [page 680]. For example, in an *Incidents* table, you could create a relationship to the *Contacts* table, letting users select one or more contacts in an incident. If you enable searching for *Multi-Relational* fields, Value Find and Relational Field Value Search are enabled. For more information about *Multi-Relational* fields, see Types of Relational Fields [page 138].
	*Multi-Selection*	*Multi-Selection* fields let users select one or more values that you create for the field in SBM Composer. Users can select values from a drop-down list or set of check boxes. If you enable searching for *Multi-Selection* fields, Value Find is enabled.
	*Multi-User*	*Multi-User* fields let users select users or groups as values for the fields. The type of selections available depend on settings applied in SBM Composer. Users can select values from a drop-down list or set of check boxes. If you enable searching for *Multi-User* fields, Value Find is enabled.

Icon	Field Type	Description
	*Numeric*	*Numeric* fields store integers, floating point values, or fixed precision values. A prefix and suffix can also be set for *Numeric* fields. Select the Show thousands separator check box to show the values in *Numeric* fields with commas. For example, after selecting this check box, a *Numeric* field value is displayed as 1,000 rather than 1000.
	*Single Relational*	*Relational* fields allow users to set up relationships between the different tables. *Single Relational* fields let users select one value to associate with a particular primary or auxiliary table. For example, you could create a problem relationship to an *Issues* table. Users can select values from a drop-down list or search for values using the Value Find. If you enable searching for *Single Relational* fields, Value Find and Relational Field Value Search are enabled. For more information about *Single Relational* fields, see Types of Relational Fields [page 138].
	*Single Selection*	*Single Selection* fields let users select one value that you create for the field in SBM Composer. If you enable searching for *Single Selection* fields, Value Find is enabled.
	*Sub-Relational*	*Sub-Relational* fields allow access to additional fields in a related primary or auxiliary table. *Sub-Relational* fields are tied to a *Single-Relational* field already in an *application* [page 679]. For example, if an *Incidents* table contains a *Single-Relational* field to an *Issues* table, you can include a *Sub-Relational* field of *Owner*. This lets users see the owner of a related issue within the incident. *Sub-Relational* fields are only available if *Single Relational* fields already exist for a particular table. For more information about *Sub-Relational* fields, see Types of Relational Fields [page 138].
	*Summation*	*Summation* fields sum the values of weights of *Single Selection* fields or values for *Numeric* fields set as integers.
	*Text*	*Text* fields can be of fixed length up to 255 Unicode characters, Memo type (whose length varies by DBMS), or a Journal type. Custom *Text* fields of fixed length can be configured to replace text characters with asterisks (*), which is useful for password entry fields.

Icon	Field Type	Description
	*User*	*User* fields let users select a single user or group member to associate with an item. *User* fields are also used to establish primary *ownership* [page 690] for primary items and can also be used to establish secondary ownership. Users can select a value from a drop-down list or search for values using the Value Find. System *User* fields, such as *Owner* and *Submitter*, are indicated by a red circle in the bottom left corner of the *User* field icon. If you enable searching for *User* fields, Value Find is enabled.

Types of Relational Fields

Relational fields are used to establish relationships between data in primary and auxiliary tables. Relational fields are created in SBM Composer.

There are three types of Relational fields:

- **Multi-Relational Field** – *Multi-Relational* fields allow users to select one or more items from a primary or auxiliary table as values for the field. For example, in a primary table used to manage customer support incidents, you could create a relationship to the *Contacts* table, allowing users to select one or more contacts in an incident. In this case, each incident could have many contacts. You can set up a *Multi-Relational* field that enables users to select multiple values from a drop-down list or a set of check boxes or to search for values using Value Find or Relational Field Value Search. Users who have privileges to view the Relational field item in the related table can click the pop-up icon that appears next to the *Multi-Relational* field when they view items. The related item opens in a pop-up window, and users can work with the item as their privileges allow.

- **Single Relational Field** – *Single Relational* fields allow users to select a single item from a primary or auxiliary table as a value for the field. For example, you could create a relationship in a primary table used to track defects to the *Problems* table. In this case, each defect could be associated with a problem in the *Problems* table. You can set up a *Single Relational* field that enables users to select a value from a drop-down list or to search for a value using the Value Find or Relational Field Value Search. Users who have privileges to view the Relational field item in the related table can click the pop-up icon that appears next to the *Single-Relational* field when they view items. The related item opens in a pop-up window, and users can work with the item as their privileges allow.

- **Sub-Relational Field** - *Sub-Relational* fields provide access to additional field value in a related primary or auxiliary table. *Sub-Relational* fields are tied to a *Single-Relational* field already in a table or workflow. For example, if your *Incidents* workflow contains a *Single-Relational* field to the *Issues* table, you can include a *Sub-Relational* field of Owner. This enables users to view the owner of a related issue within the incident.

The following information applies to Relational fields:

- Consider user privileges carefully when you use Relational fields across tables. Users who do not have privileges to view items in a Relational field table may be able to view those items as values in Relational fields in other tables. To prevent users from

viewing Relational field values, move Relational fields to a section for which users do not have view privileges.

- The *Item ID* and value display format display as values for *Single Relational* and *Multi-Relational* fields. If an auxiliary table used for Relational fields does not contain an *Item ID* field, only the value display format appears as values. The Value Display Format is set in SBM Composer; you can view this setting on the **Advanced** tab of the **Edit Table** dialog box in SBM System Administrator.
- These rules apply to value sorting:
  - Relational field values based on primary tables cannot be sorted manually. Values are sorted by project and then Item ID.
  - Relational field values based on auxiliary tables are sorted by the table's value display format. To change the order in which values sort for the Relational field, modify the value display format for the auxiliary table.
- You can create relationships to fields in the Global Process App tables, but you cannot create relationships from tables in the Global Process App.

Modifying Fields

SBM Composer lets you modify all system fields and *custom fields* [page 682] in a workflow, state, and transition. The level at which you modify the field, such as workflow, state or transition, determines which field properties are available for modification.

At the table level you can modify general field information, options and attributes. For some fields, you can enter information about field dependencies and enter selection values. You can also set field permissions and field order. These fields become the default fields you use in your workflows and forms. You can override many of the default settings at the workflow, state, and transition level.

Field Property Editor

The field Property Editor lets you enter information and select options for the field selected in the table editor.

Tip: When you select a field in the table editor, the field type you are working on is shown in the **Fields** drop-down list in the field Property Editor. Selecting a different field from the drop-down list selects it in the table editor and displays field properties in the field Property Editor.

Field information and options you set affect the behavior and functionality of the fields as they are used collect information as users work with items in project or auxiliary tables in the SBM User Workspace. You can use the field Property Editor to specify information for fields in both primary and auxiliary tables.

Controls used for creating field settings are grouped in tabs located on the left side of the field Property Editor.

- General Tab of the Field Property Editor [page 140]
- Options Tab of the Field Property Editor [page 140]
- Attributes Tab of the Field Property Editor [page 165]

- Dependencies Tab of the Field Property Editor [page 167]

General Tab of the Field Property Editor

This tab is available when you select a field in a table in the table editor.

Element	Description
Name	This is the name as it will appear to users in the SBM User Workspace.
Database field name	The unique name by which the field is stored in the database.  **Note:** • You cannot change the database field name after the *application* [page 679] containing the table is published to the SBM Application Repository. • You cannot change the automatically specified database name of a *system field* [page 699]. • The database field name can contain only ASCII alphanumeric characters (A–Z, a–z, 0–9) and underscores. Any other characters are ignored as you type them.
Type	The field type, such as *Numeric* and *Text*. The field type cannot be changed.
Description	An optional description of the purpose and use of the field. The text you enter is displayed to users who hover the mouse pointer over the field name in the SBM User Workspace.
Privilege section	See Organizing Fields [page 124] for information about privilege sections in quick forms and custom forms.

Options Tab of the Field Property Editor

This tab is available when you select a field in a table in the table editor.

Use this tab to specify the appearance and behavior of fields in the SBM User Workspace. Each field type has different options, which are described in the topics in this section.

 **Note:** For system fields, some options are unavailable, and others cannot be changed. See System Fields [page 125] for more information.

- Binary/Trinary Field Options [page 141]
- Date/Time Field Options [page 142]
- Numeric Field Options [page 144]
- Text Field Options [page 146]
- Summation Field Options [page 148]

- Folder Field Options [page 149]
- Single Selection Field Options [page 151]
- Multi-Selection Field Options [page 153]
- User Field Options [page 155]
- Multi-User Field Options [page 156]
- Multi-Group Field Options [page 158]
- Single Relational Field Options [page 160]
- Multi-Relational Field Options [page 161]
- Sub-Relational Field Options [page 163]

Binary/Trinary Field Options

Binary/Trinary fields can store either two or three values that you specify.

They can also be used to determine *subtask status* [page 698]. The values specified in the **First value**, **Second value**, and **Third value** properties correspond to actions based on subtask status. See Tutorial: Defining Subtask-Driven Actions [page 383] for related information.

Style

- **Dropdown list**

  Select this option to let users select a single value from a drop-down list populated with the values from the **First Value**, **Second Value**, and (for trinary fields) **Third Value** properties.

- **Radio buttons**

  Users select one of radio buttons labeled with the values from the **First Value**, **Second Value**, and **Third Value** properties.

- **Check box**
  Users select or clear a check box labeled with the field name specified on the **General** tab. This option is available only for binary fields.

  Important: Internally, SBM treats the first value as 0, which corresponds to "checked," and treats the second value as 1, which corresponds to "not checked".

- **Third value**

  Select this option to create a trinary field, adding a third value to the drop-down list or defining a third radio button. This option is not available when the **Checkbox** option is selected.

- **First**, **Second**, and **Third value**

  If the field is used to determine subtask status, the values you specify must be related to the three applicable subtask statuses: **In Review/Progress**, **Accepted/Complete**, and **Rejected/Reverted**.

These properties are not available when the **Checkbox** option is selected.

Display

- **Span entire row on forms**

  Select this option to make this field the only field to appear on this row in the SBM User Workspace.

Search & Query

- **Appears in report field lists**

  Select this option to include the field in lists on report forms.

  Note: If this option is cleared after the field is used in a report (in the SBM User Workspace), the changed setting is ignored for that report; that is, the field will still appear until it is removed from the report definition.

  See General Tab of the Field Property Editor [page 140] for related information.

- **Appears on lookup form and relational field value lookup**

  Select this option to add the field to the Lookup form of the Advanced Lookup Tool and the *Relational Field Value Lookup* [page 694] form.

  Note: The field order for the Lookup and Relational Field Value Lookup forms is determined in SBM by the default field order of the table's first project in the project hierarchy. Projects are defined in SBM System Administrator.

- **Appears on searches for this table**

  Select this option to specify that the field will be available for selection on the Advanced Search page in the SBM User Workspace. This setting applies to primary and auxiliary tables. See Considerations for Advanced Search [page 176] for additional information.

Date/Time Field Options

Date/Time fields store dates and time values.

Style

Important: You cannot change the style of a *Date/Time* field after a process app is published. For example, you cannot change a date only field to a time of day field. However, there is a way to reset the field and change it if you intend to deploy the process app to a different *environment* [page 684]. For more information, see Modifying Locked Fields in a Published Process App [page 175].

Select one of the display options for the field. Note that date/time values are always stored in native date/time format in the database. Values for *Date/Time* fields set to record time only or elapsed time are stored as integers. Assuming a default value of "Now", the value stored in the database is equal to the number of seconds that have passed since 12:00:00 a.m. in the submitter's or modifier's time zone.

- **Date and time**

  Used for display of date-and-time values.

- **Date only**

Used for display of date values.

- **Time of day**

  Used for display of time values.

- **Elapsed time**

  Stores a value representing an elapsed time in hours, minutes, and seconds. You can choose whether to include seconds in the display value, however. Users can provide an elapsed time in the format specified on the form (`d hh:mm:ss`, `d hh:mm`, `hh:mm:ss`, or `hh:mm`) or type an integer to represent the number of hours that should appear in the field. When you select this option, the **Stopwatch**, **Calculate days**, and **Show seconds** options become available.

  - The **Stopwatch** option lets users record elapsed time in the SBM User Workspace. The stopwatch starts recording time when a *transition form* [page 700] opens in the SBM User Workspace for submitting, transitioning, or updating an item. Users can edit the elapsed time by typing a new value in the field when the time is paused on the transition form.

    Note: The timer does not record elapsed time if it is moved to the **Hidden Fields** section. The stopwatch records time only when a form is open, so hiding the *Stopwatch* field in a state does not affect elapsed time. You must hide the *Stopwatch* field in the specific transition if you do not want it to record time. If the *Stopwatch* field is read-only or is placed in a field section the user does not have privileges or preferences to view, the timer still records elapsed time while the form is open. If the user cancels the transition or resets the form, the elapsed time is not saved.

    To avoid incorrect stopwatch values, you should enable the **Show seconds** option when the **Stopwatch** option is enabled.

  - Select the **Calculate days** option to calculate and display elapsed time in the format `d hh:mm:ss`. For example, if a user enters the integer `50` in the field, it is interpreted as hours and converted to days, and the resulting value is displayed as 2 2:00:00 (if the **Show seconds** option is also enabled). Note that the elapsed days do not appear in the SBM User Workspace until at least one full day has elapsed.

  - Select the **Show seconds** option to display the elapsed time values in `00:00:00` format. (Disable the **Show seconds** option to display elapsed times in `00:00` format.) Note that seconds are retained in the database, even if they are not displayed.

Display

- **Span entire row on forms**

  Select this option to make this field the only field to appear on this row in the SBM User Workspace.

Search & Query

- **Appears in report field lists**

Select this option to include the field in lists on report forms.

Note: If this option is cleared after the field is used in a report (in the SBM User Workspace), the changed setting is ignored for that report; that is, the field will still appear until it is removed from the report definition.

See General Tab of the Field Property Editor [page 140] for related information.

- **Appears on lookup form and relational field value lookup**

  Select this option to add the field to the Lookup form of the Advanced Lookup Tool and the Relational Field Value Lookup form. For a *Date/Time* field, this option adds Start and End boxes next to the field on the lookup form, letting users specify a start and end date as part of their search criteria.

  Note: The field order for the Lookup and Relational Field Value Lookup forms is determined in SBM by the default field order of the table's first project in the project hierarchy.

- **Appears on searches for this table**

  Select this option to specify that the field will be available for selection on the Advanced Search page in the SBM User Workspace. This setting applies to primary and auxiliary tables. See Considerations for Advanced Search [page 176] for additional information.

Numeric Field Options

Numeric fields store signed integers, floating point values, or fixed precision values. A prefix and suffix can also be set for *Numeric* fields.

Tip: *Numeric* fields can be used as weights in Trend reports. When you use weights, be sure to set a default value, or set the field as required. (See Attributes Tab of the Field Property Editor [page 165] for details.)

Style

Important: You cannot change the style of a *Numeric* field after a process app is published. For example, you cannot change an integer field to a floating point field. However, there is a way to reset the field and change it if you intend to deploy the process app to a different *environment* [page 684]. For more information, see Modifying Locked Fields in a Published Process App [page 175].

- **Integer**

  The maximum positive integer accepted is 2147483643, and the minimum negative integer accepted is –999999999.

- **Floating point**
  The maximum and minimum accepted range of values are determined by your server hardware.

  Note: SBM Composer respects the regional option set for your computer for floating point and fixed precision values. For example, if your regional option is "Czech," a comma is used instead of a decimal point.

- **Fixed precision**

  The maximum and minimum accepted range of values are determined by your server hardware.

- **Digits displayed after the decimal point**

  Specify the number of digits (with a minimum of 0 and a maximum of 15) you want to appear after the decimal point in the SBM User Workspace. This option is available only when the **Fixed precision** option is enabled.

  Note: Some large integer and floating point values are reserved for internal use. Integer values larger than 2147483643 and floating point numbers between 4294967292.000 and 4294967295.99999 cannot be used in *Numeric* fields for this reason.

Display

- **Prefix** and **Suffix**
  Type any plain text or formatting HTML code you want to display before and after the field's values in the SBM User Workspace. For best results, formatting changes should be applied using styles on custom forms rather than by adding HTML tags to the field's prefix and suffix. For more advanced HTML formatting, create a custom form for your workflow, and then add an HTML/JavaScript *widget* [page 701] to the form.

  Note: Use the `<` and `>` character entities to display the less than (<) or greater than (>) signs, respectively. Otherwise, these signs will be interpreted as HTML code. Use the ` ` character entity for each extra space you want to display.

  Note: The rendered HTML is visible in preview mode and in the SBM User Workspace, but not in the form editor.

- **Show thousands separator**
  Select this option to display in the SBM User Workspace values with commas (such as "1,000" for "1000").

  Note: SBM Composer respects the regional option set for your computer. For example, if your regional option is "Czech," a space is used instead of a comma.

- **Span entire row on forms**

  Select this option to make this field the only field to appear on this row in the SBM User Workspace.

Search & Query

- **Appears in report field lists**

  Select this option to include the field in lists on report forms.

  Note: If this option is cleared after the field is used in a report (in the SBM User Workspace), the changed setting is ignored for that report; that is, the field will still appear until it is removed from the report definition.

  See General Tab of the Field Property Editor [page 140] for related information.

- **Appears on lookup form and relational field value lookup**

Select this option to add the field to the Lookup form of the Advanced Lookup Tool and the *Relational Field Value Lookup* [page 694] form.

Note: The field order for the Lookup and Relational Field Value Lookup forms is determined in SBM by the default field order of the table's first project in the project hierarchy. Projects are defined in SBM System Administrator.

- **Appears on searches for this table**

  Select this option to specify that the field will be available for selection on the Advanced Search page in the SBM User Workspace. This setting applies to primary and auxiliary tables. See Considerations for Advanced Search [page 176] for additional information.

Text Field Options

Text fields let users enter information about primary and auxiliary items.

Note: System fields could offer a subset of the options described here.

Style

Important: You cannot change the style of a *Text* field after a process app is deployed. For example, you cannot change a fixed-length field to a memo field. However, there is a way to reset the field and change it if you intend to deploy the process app to a different *environment* [page 684]. For more information, see Modifying Locked Fields in a Published Process App [page 175].

- **Memo**

  Select this option to let a user enter up to 65,535 Unicode characters in the field. (See **Span entire row on forms**, below.)

- **Journal**

  Select this option to automatically insert a date, time, and user ID whenever a user enters text in the field. A *journal field* [page 688] can accept up to 65,535 Unicode characters.

  Selecting this option enables these additional options:

  - **Append only**

    Forces the text in any new journal entries to be appended to the text in an existing journal entry, preventing users from modifying existing journal entries. Clear this option to let users edit existing journal entries.

  - **Insert newest first**

    Select this option to insert *new* journal entries from newest to oldest. Clear the option to insert new journal entries in chronological order. (Changing this option does not affect the order in which existing journal entries are listed.)

- **Fixed length**

  Select this option and specify maximum number (up to 255 Unicode characters) allowed in the field. (See **Span entire row on forms**, below.)

(Custom text fields only) Selecting this option displays an additional option:

- **Password**

  Forces the characters in this field to be replaced with asterisks (*) on state and transition forms and in the form preview.

  Note: This option does not appear on the **Options** tab for system text fields, like *Title*, or for fields for which the **Memo** or **Journal** option is selected.

Display

- **Prefix** and **Suffix**
  Type any plain text or formatting HTML code you want to display before and after the field's values in the SBM User Workspace. For best results, formatting changes should be applied using styles on custom forms rather than by adding HTML tags to the field's prefix and suffix. For more advanced HTML formatting, create a custom form for your workflow, and then add an HTML/JavaScript *widget* [page 701] to the form.

  Note: Use the `<` and `>` character entities to display the less than (<) or greater than (>) signs, respectively. Otherwise, these signs will be interpreted as HTML code. Use the ` ` character entity for each extra space you want to display.

  Note: The rendered HTML is visible in preview mode and in the SBM User Workspace, but not in the form editor.

- **Render HTML tags**

  Select this option to have the SBM User Workspace render any HTML tags that users add to this field. (Disable the option to have any HTML tags displayed as plain text, instead.) This setting only applies to memo fields, and applies to the field at all workflow and project levels.

  Note: Any obviously "suspicious" HTML is not rendered by default.

  Remember: Improper or invalid HTML could have a negative impact on the field and possibly on the entire form. Refer to the World Wide Web Consortium Web site at http://www.w3.org for information about HTML syntax.

- **Span entire row on forms**

  Select this option to make this field the only field to appear on this row in the SBM User Workspace.

  Note: This setting does not affect the display of *Text* fields with the **Memo** option or the **Fixed length** option if the specified length is greater than the Max Text Field Size (set in SBM System Administrator).

Search & Query

- **Appears in report field lists**

Select this option to include the field in lists on report forms.

Note: If this option is cleared after the field is used in a report (in the SBM User Workspace), the changed setting is ignored for that report; that is, the field will still appear until it is removed from the report definition.

See General Tab of the Field Property Editor [page 140] for related information.

- **Appears on lookup form and relational field value lookup**

  Select this option to add the field to the Lookup form of the Advanced Lookup Tool and the *Relational Field Value Lookup* [page 694] form.

  Note: The field order for the Lookup and Relational Field Value Lookup forms is determined in SBM by the default field order of the table's first project in the project hierarchy. Projects are defined in SBM System Administrator.

- **Appears on searches for this table**

  Select this option to specify that the field will be available for selection on the Advanced Search page in the SBM User Workspace. This setting applies to primary and auxiliary tables. See Considerations for Advanced Search [page 176] for additional information.

- **Automatically prepend wildcard to lookup text**

  Note: This option is available only when the **Appears on lookup form and relational field value lookup** option is enabled.

  Select this option to automatically use wildcard characters both *before* and *after* the search text for this field in the SBM User Workspace, unless there are wildcard characters *within* the search text. This helps users search for keywords more easily.

  When this option is enabled, a percent sign (%) is displayed next to the field on the Lookup form in the SBM User Workspace.

  When this option is disabled, a wildcard character is automatically appended to the search text, unless the search text itself includes wildcard characters.

- **Include field in keyword searches**

  Select this option to let users search for particular text from Knowledge Base Search, Basic Search, Advanced Search, and Global Search forms in the SBM User Workspace.

  Tip: You could improve the performance of your process app in the by disabling this option for any *Text* fields that you do not expect to be relevant in searches (especially those with the **Memo** option enabled). Note that disabling this option also removes the field from keyword searches for the Basic Search and Global Search feature.

Summation Field Options

Summation fields sum the values contained in *Single Selection* fields by the assigned weights or in *Numeric* fields set as integers.

Values

- **Fields to sum**

Lists *Single Selection* fields that have an assigned weight or *Numeric* fields set as integers. Select the fields to be summed.

Display

- **Span entire row on forms**

  Select this option to make this field the only field to appear on this row in the SBM User Workspace.

Search & Query

- **Appears in report field lists**

  Select this option to include the field in lists on report forms.

  Note: If this option is cleared after the field is used in a report (in the SBM User Workspace), the changed setting is ignored for that report; that is, the field will still appear until it is removed from the report definition.

  See General Tab of the Field Property Editor [page 140] for related information.

- **Appears on lookup form and relational field value lookup**

  Select this option to add the field to the Lookup form of the Advanced Lookup Tool and the *Relational Field Value Lookup* [page 694] form.

  Note: The field order for the Lookup and Relational Field Value Lookup forms is determined in SBM by the default field order of the table's first project in the project hierarchy. Projects are defined in SBM System Administrator.

- **Appears on searches for this table**

  Select this option to specify that the field will be available for selection on the Advanced Search page in the SBM User Workspace. This setting applies to primary and auxiliary tables. See Considerations for Advanced Search [page 176] for additional information.

Folder Field Options

Folder fields are used to link primary or auxiliary items to a specified Public or *Knowledge Base* [page 688] folder. (Folders are created and managed in SBM System Administrator.)

Note: To use a folder as a field selection, it must be configured to accept new items. In addition, users must be granted the **Add Items to Folder** privilege. For more information, see the *SBM System Administrator Guide*.

Style

- **Allow searching**

Select this option to enable the *value find* [page 700] feature for the field on submit, transition, and update forms. This enables users enter to search criteria (an entire word, a few letters, or an asterisk) and then click the **Find** button (or press Enter) to perform the search. Results appear in a drop-down list, from which the user selects a value for the field. This option also enables users to search for values in the *Rich Editable Grid* [page 694].

Tip: If the field will contain 200 or more selections, this option is recommended for best performance in the SBM User Workspace. If you do not allow searching, fields that contain over 250 selection values are automatically set as searchable in the Editable Grid.

- **Single drop-down list**

  Select this option to let users select a value from a populated drop-down list.

Display

- **Span entire row on forms**

  Select this option to make this field the only field to appear on this row in the SBM User Workspace.

Search & Query

- **Appears in report field lists**

  Select this option to include the field in lists on report forms.

  Note: If this option is cleared after the field is used in a report (in the SBM User Workspace), the changed setting is ignored for that report; that is, the field will still appear until it is removed from the report definition.

  See General Tab of the Field Property Editor [page 140] for related information.

- **Appears on lookup form and relational field value lookup**

  Select this option to add the field to the Lookup form of the Advanced Lookup Tool and the *Relational Field Value Lookup* [page 694] form.

  Note: The field order for the Lookup and Relational Field Value Lookup forms is determined in SBM by the default field order of the table's first project in the project hierarchy. Projects are defined in SBM System Administrator.

- **Appears on searches for this table**

  Select this option to specify that the field will be available for selection on the Advanced Search page in the SBM User Workspace. This setting applies to primary and auxiliary tables. See Considerations for Advanced Search [page 176] for additional information.

- **On lookup or query-at-runtime forms**

  - **Allow searching**

    This option enables the *value find* and *relational field value lookup* [page 694] features on the Lookup form of the Advanced Lookup Tool, the Relational Field Value Lookup form, and Query-at-Runtime forms for reports. (See *Allow Searching* under *Style* above for a description of the *value find* feature.) The *relational field value lookup* [page 694] feature provides an advanced searching

mechanism that lets users find values for *Single Relational* and *Multi-Relational* fields. If this option is enabled, a small magnifying glass appears next to the **Find** button in the SBM User Workspace.

- **Show full list**

  This option lets users select a value from a drop-down list on the Lookup form of the Advanced Lookup Tool, the Relational Field Value Lookup form, and Query-at-Runtime forms for reports.

Single Selection Field Options

Single Selection fields let users make one selection for the field in the SBM User Workspace.

Style

- **Allow searching**

  Select this option to enable the *value find* [page 700] feature for the field on submit, transition, and update forms. This enables users enter to search criteria (an entire word, a few letters, or an asterisk) and then click the **Find** button (or press Enter) to perform the search. Results appear in a drop-down list, from which the user selects a value for the field. This option also enables users to search for values in the *Rich Editable Grid* [page 694].

  Tip: If the field will contain 200 or more selections, this option is recommended for best performance in the SBM User Workspace. If you do not allow searching, fields that contain over 250 selection values are automatically set as searchable in the Editable Grid.

- **Single drop-down list**

  Select this option to let users select a value from a populated drop-down list.

Values

Add, edit, remove, and sort the values from which users will be able to choose in the SBM User Workspace. Click the values or weights (cells) in the columns to modify them for each value or weight (row) you want to include. Use the **Move up** and **Move down** controls to sort values according to the order that you want users to see in the SBM User Workspace. You can also click the **Value** column heading to sort the values alphanumerically in ascending or descending order, and click the **Weight** column heading to sort the weights numerically in ascending or descending order.

Tip: You can use shortcut keys to edit and navigate values. For more information, see Selection Field Shortcut Keys [page 69].

The weight gives a numeric value to the selections in the field. Weights can be used in Trend reports and in *Summation* fields when users sum the values contained in the *Single Selection* field.

Tip: For accurate results when using weights, make sure the fields that use weighting always have values.

Default weight for new values: Specify the weight to be assigned to new values you add to the list. You can change the default weight for a value by typing in the **Weight**

column or by using the up and down arrow buttons that are displayed when you click its default weight in the **Weight** column.

For *primary table* [page 692] fields, you can set the status of the field to **Enabled** or **Disabled**.

Note: If you delete this field and later restore it, the values you specify are also restored. For more information, see Deleting and Restoring Fields [page 175].

Display

- **Span entire row on forms**

  Select this option to make this field the only field to appear on this row in the SBM User Workspace.

Search & Query

- **Appears in report field lists**

  Select this option to include the field in lists on report forms.

  Note: If this option is cleared after the field is used in a report (in the SBM User Workspace), the changed setting is ignored for that report; that is, the field will still appear until it is removed from the report definition.

  See General Tab of the Field Property Editor [page 140] for related information.

- **Appears on lookup form and relational field value lookup**

  Select this option to add the field to the Lookup form of the Advanced Lookup Tool and the *Relational Field Value Lookup* [page 694] form.

  Note: The field order for the Lookup and Relational Field Value Lookup forms is determined in SBM by the default field order of the table's first project in the project hierarchy. Projects are defined in SBM System Administrator.

- **Appears on searches for this table**

  Select this option to specify that the field will be available for selection on the Advanced Search page in the SBM User Workspace. This setting applies to primary and auxiliary tables. See Considerations for Advanced Search [page 176] for additional information.

- **On lookup or query-at-runtime forms**

  - **Allow searching**

    This option enables the *value find* and *relational field value lookup* [page 694] features on the Lookup form of the Advanced Lookup Tool, the Relational Field Value Lookup form, and Query-at-Runtime forms for reports. (See *Allow Searching* under *Style* above for a description of the *value find* feature.) The *relational field value lookup* [page 694] feature provides an advanced searching mechanism that lets users find values for *Single Relational* and *Multi-Relational* fields. If this option is enabled, a small magnifying glass appears next to the **Find** button in the SBM User Workspace.

  - **Show full list**

This option lets users select a value from a drop-down list on the Lookup form of the Advanced Lookup Tool, the Relational Field Value Lookup form, and Query-at-Runtime forms for reports.

Note: If a *Single Selection* field has 1,000 items or more, the Advanced Search ignores the **On lookup or query-At-runtime forms** setting, and instead displays it as a searchable field.

Multi-Selection Field Options

Multi-Selection fields let users make one or more selections for the field in the SBM User Workspace.

Style

- **Allow searching**

  Select this option to enable the *value find* [page 700] feature for the field on submit, transition, and update forms. This enables users enter to search criteria (an entire word, a few letters, or an asterisk) and then click the **Find** button (or press Enter) to perform the search. Results appear in a drop-down list, from which the user selects a value for the field. This option also enables users to search for values in the *Rich Editable Grid* [page 694].

  Tip: If the field will contain 200 or more selections, this option is recommended for best performance in the SBM User Workspace. If you do not allow searching, fields that contain over 250 selection values are automatically set as searchable in the Editable Grid.

- **List size**

  Specify the number of rows in which to display values for the field on forms in the SBM User Workspace. Typically, you specify the number of expected values. Note that this option is unavailable when the **Checkboxes** option is selected.

- **List box**

  Select this option to let users select one or more values from a populated drop-down list.

- **Checkboxes**

  Select this option to let users select one or more check boxes.

Values

Add, edit, remove, and sort the values from which users will be able to choose in the SBM User Workspace. Click the values (cells) in the columns to modify them for each value (row) you want to include. Use the **Move up** and **Move down** controls to sort values according to the order that you want users to see in the SBM User Workspace. You can also click the **Value** column heading to sort the values alphanumerically in ascending or descending order.

Tip: You can use shortcut keys to edit and navigate values. For more information, see Selection Field Shortcut Keys [page 69].

For *primary table* [page 692] fields, you can set the status of the field to **Enabled** or **Disabled**.

Note: If you delete this field and later restore it, the values you specify are also restored. For more information, see Deleting and Restoring Fields [page 175].

Display

- **Span entire row on forms**

  Select this option to make this field the only field to appear on this row in the SBM User Workspace.

Search & Query

- **Appears in report field lists**

  Select this option to include the field in lists on report forms.

  Note: If this option is cleared after the field is used in a report (in the SBM User Workspace), the changed setting is ignored for that report; that is, the field will still appear until it is removed from the report definition.

  See General Tab of the Field Property Editor [page 140] for related information.

- **Appears on lookup form and relational field value lookup**

  Select this option to add the field to the Lookup form of the Advanced Lookup Tool and the *Relational Field Value Lookup* [page 694] form.

  Note: The field order for the Lookup and Relational Field Value Lookup forms is determined in SBM by the default field order of the table's first project in the project hierarchy. Projects are defined in SBM System Administrator.

- **Appears on searches for this table**

  Select this option to specify that the field will be available for selection on the Advanced Search page in the SBM User Workspace. This setting applies to primary and auxiliary tables. See Considerations for Advanced Search [page 176] for additional information.

- **On lookup or query-at-runtime forms**

  - **Allow searching**

    This option enables the *value find* and *relational field value lookup* [page 694] features on the Lookup form of the Advanced Lookup Tool, the Relational Field Value Lookup form, and Query-at-Runtime forms for reports. (See *Allow Searching* under *Style* above for a description of the *value find* feature.) The *relational field value lookup* [page 694] feature provides an advanced searching mechanism that lets users find values for *Single Relational* and *Multi-Relational* fields. If this option is enabled, a small magnifying glass appears next to the **Find** button in the SBM User Workspace.

  - **Show full list**

    This option lets users select a value from a drop-down list on the Lookup form of the Advanced Lookup Tool, the Relational Field Value Lookup form, and Query-at-Runtime forms for reports.

User Field Options

User fields let users select one user or group from the field in the SBM User Workspace.

Style

- **Allow searching**

  Select this option to enable the *value find* [page 700] feature for the field on submit, transition, and update forms. This enables users enter to search criteria (an entire word, a few letters, or an asterisk) and then click the **Find** button (or press Enter) to perform the search. Results appear in a drop-down list, from which the user selects a value for the field. This option also enables users to search for values in the *Rich Editable Grid* [page 694].

  Tip: If the field will contain 200 or more selections, this option is recommended for best performance in the SBM User Workspace. If you do not allow searching, fields that contain over 250 selection values are automatically set as searchable in the Editable Grid.

- **Single drop-down list**

  Select this option to let users select a value from a populated drop-down list.

Associated Roles

Select the roles that determine which values users can select in the SBM User Workspace. For *User* fields, users can select a single value. For *Multi-User* and *Multi-Groups* fields, multiple values can be selected.

Display

- **Span entire row on forms**

  Select this option to make this field the only field to appear on this row in the SBM User Workspace.

Search & Query

- **Appears in report field lists**

  Select this option to include the field in lists on report forms.

  Note: If this option is cleared after the field is used in a report (in the SBM User Workspace), the changed setting is ignored for that report; that is, the field will still appear until it is removed from the report definition.

  See General Tab of the Field Property Editor [page 140] for related information.

- **Appears on lookup form and relational field value lookup**

  Select this option to add the field to the Lookup form of the Advanced Lookup Tool and the *Relational Field Value Lookup* [page 694] form.

  Note: The field order for the Lookup and Relational Field Value Lookup forms is determined in SBM by the default field order of the table's first project in the project hierarchy. Projects are defined in SBM System Administrator.

- **Appears on searches for this table**

Select this option to specify that the field will be available for selection on the Advanced Search page in the SBM User Workspace. This setting applies to primary and auxiliary tables. See Considerations for Advanced Search [page 176] for additional information.

- **On lookup or query-at-runtime forms**
  - **Allow searching**

    This option enables the *value find* and *relational field value lookup* [page 694] features on the Lookup form of the Advanced Lookup Tool, the Relational Field Value Lookup form, and Query-at-Runtime forms for reports. (See *Allow Searching* under *Style* above for a description of the *value find* feature.) The *relational field value lookup* [page 694] feature provides an advanced searching mechanism that lets users find values for *Single Relational* and *Multi-Relational* fields. If this option is enabled, a small magnifying glass appears next to the **Find** button in the SBM User Workspace.

  - **Show full list**

    This option lets users select a value from a drop-down list on the Lookup form of the Advanced Lookup Tool, the Relational Field Value Lookup form, and Query-at-Runtime forms for reports.

Multi-User Field Options

Multi-User fields let users select one or more users or groups from the field in the SBM User Workspace.

Style

- **Allow searching**

  Select this option to enable the *value find* [page 700] feature for the field on submit, transition, and update forms. This enables users enter to search criteria (an entire word, a few letters, or an asterisk) and then click the **Find** button (or press Enter) to perform the search. Results appear in a drop-down list, from which the user selects a value for the field. This option also enables users to search for values in the *Rich Editable Grid* [page 694].

  Tip: If the field will contain 200 or more selections, this option is recommended for best performance in the SBM User Workspace. If you do not allow searching, fields that contain over 250 selection values are automatically set as searchable in the Editable Grid.

- **List size**

  Specify the number of rows in which to display values for the field on forms in the SBM User Workspace. Typically, you specify the number of expected values. Note that this option is unavailable when the **Checkboxes** option is selected.

- **List box**

  Select this option to let users select one or more values from a populated drop-down list.

- **Checkboxes**

  Select this option to let users select one or more check boxes.

Associated Roles

Select the roles that determine which values users can select in the SBM User Workspace. For *User* fields, users can select a single value. For *Multi-User* and *Multi-Groups* fields, multiple values can be selected.

- **Selection mode**
  - **Individual users**
    Select this option to omit groups from the displayed list of users in the SBM User Workspace.

    CAUTION: Selecting this option could negatively affect performance.

  - **Groups & users**

    Select this option to include groups and users in the displayed list.

    If you change a *Multi-User* field's **Selection mode** to **Groups & users** in SBM Composer, that field will not only display individual user names, but it will also display one or more group names as available selections in the SBM User Workspace if the members of a group were designated as a possible selections for the *Multi-User* field within SBM System Administrator.

    Note: If you change the **Selection mode** for either a *Multi-User* or *Secondary Owner* [page 696] field back to **Individual users** after populating groups in either field, performance may be impacted when the groups are unrolled in order to display large lists of individual users. Alternatively, you can improve performance by selecting **Groups & users** for any *Secondary Owner* field that is auto-populated by a *Multi-Group* or *Multi-User* field that contains groups: the list of users that would otherwise appear in *Secondary Owner* are rolled up into groups after selecting **Groups & users** in that case.

Display

- **Span entire row on forms**

  Select this option to make this field the only field to appear on this row in the SBM User Workspace.

Search & Query

- **Appears in report field lists**

  Select this option to include the field in lists on report forms.

  Note: If this option is cleared after the field is used in a report (in the SBM User Workspace), the changed setting is ignored for that report; that is, the field will still appear until it is removed from the report definition.

  See General Tab of the Field Property Editor [page 140] for related information.

- **Appears on lookup form and relational field value lookup**

Select this option to add the field to the Lookup form of the Advanced Lookup Tool and the *Relational Field Value Lookup* [page 694] form.

Note: The field order for the Lookup and Relational Field Value Lookup forms is determined in SBM by the default field order of the table's first project in the project hierarchy. Projects are defined in SBM System Administrator.

- **Appears on searches for this table**

  Select this option to specify that the field will be available for selection on the Advanced Search page in the SBM User Workspace. This setting applies to primary and auxiliary tables. See Considerations for Advanced Search [page 176] for additional information.

- **On lookup or query-at-runtime forms**
  - **Allow searching**

    This option enables the *value find* and *relational field value lookup* [page 694] features on the Lookup form of the Advanced Lookup Tool, the Relational Field Value Lookup form, and Query-at-Runtime forms for reports. (See *Allow Searching* under *Style* above for a description of the *value find* feature.) The *relational field value lookup* [page 694] feature provides an advanced searching mechanism that lets users find values for *Single Relational* and *Multi-Relational* fields. If this option is enabled, a small magnifying glass appears next to the **Find** button in the SBM User Workspace.

  - **Show full list**

    This option lets users select a value from a drop-down list on the Lookup form of the Advanced Lookup Tool, the Relational Field Value Lookup form, and Query-at-Runtime forms for reports.

Multi-Group Field Options

Multi-Group fields let users select one or more groups from the field in the SBM User Workspace.

Style

- **Allow searching**

  Select this option to enable the *value find* [page 700] feature for the field on submit, transition, and update forms. This enables users enter to search criteria (an entire word, a few letters, or an asterisk) and then click the **Find** button (or press Enter) to perform the search. Results appear in a drop-down list, from which the user selects a value for the field. This option also enables users to search for values in the *Rich Editable Grid* [page 694].

  Tip: If the field will contain 200 or more selections, this option is recommended for best performance in the SBM User Workspace. If you do not allow searching, fields that contain over 250 selection values are automatically set as searchable in the Editable Grid.

- **List size**

Specify the number of rows in which to display values for the field on forms in the SBM User Workspace. Typically, you specify the number of expected values. Note that this option is unavailable when the **Checkboxes** option is selected.

- **List box**

  Select this option to let users select one or more values from a populated drop-down list.

- **Checkboxes**

  Select this option to let users select one or more check boxes.

Associated Roles

Select the roles that determine which values users can select in the SBM User Workspace. For *User* fields, users can select a single value. For *Multi-User* and *Multi-Groups* fields, multiple values can be selected.

Display

- **Span entire row on forms**

  Select this option to make this field the only field to appear on this row in the SBM User Workspace.

Search & Query

- **Appears in report field lists**

  Select this option to include the field in lists on report forms.

  Note: If this option is cleared after the field is used in a report (in the SBM User Workspace), the changed setting is ignored for that report; that is, the field will still appear until it is removed from the report definition.

  See General Tab of the Field Property Editor [page 140] for related information.

- **Appears on lookup form and relational field value lookup**

  Select this option to add the field to the Lookup form of the Advanced Lookup Tool and the *Relational Field Value Lookup* [page 694] form.

  Note: The field order for the Lookup and Relational Field Value Lookup forms is determined in SBM by the default field order of the table's first project in the project hierarchy. Projects are defined in SBM System Administrator.

- **Appears on searches for this table**

  Select this option to specify that the field will be available for selection on the Advanced Search page in the SBM User Workspace. This setting applies to primary and auxiliary tables. See Considerations for Advanced Search [page 176] for additional information.

- **On lookup or query-at-runtime forms**

  - **Allow searching**

    This option enables the *value find* and *relational field value lookup* [page 694] features on the Lookup form of the Advanced Lookup Tool, the Relational Field Value Lookup form, and Query-at-Runtime forms for reports. (See *Allow*

Searching under *Style* above for a description of the *value find* feature.) The *relational field value lookup* [page 694] feature provides an advanced searching mechanism that lets users find values for *Single Relational* and *Multi-Relational* fields. If this option is enabled, a small magnifying glass appears next to the **Find** button in the SBM User Workspace.

- **Show full list**

  This option lets users select a value from a drop-down list on the Lookup form of the Advanced Lookup Tool, the Relational Field Value Lookup form, and Query-at-Runtime forms for reports.

Single Relational Field Options

Single Relational fields let users select one item from a different table to associate with the current item.

Style

- **Allow searching**

  Select this option to enable the *value find* [page 700] feature for the field on submit, transition, and update forms. This enables users enter to search criteria (an entire word, a few letters, or an asterisk) and then click the **Find** button (or press Enter) to perform the search. Results appear in a drop-down list, from which the user selects a value for the field. This option also enables users to search for values in the *Rich Editable Grid* [page 694].

  Tip: If the field will contain 200 or more selections, this option is recommended for best performance in the SBM User Workspace. If you do not allow searching, fields that contain over 250 selection values are automatically set as searchable in the Editable Grid.

- **Single drop-down list**

  Select this option to let users select a value from a populated drop-down list.

Values

Note: If these controls show "(External Application)" and "(External Table)," changing them could break relationships among applications. SBM Composer will prompt you to confirm the change.

- **Application**

  Select the *application* [page 679] containing the table of values with which you want to populate this field.

- **Table**

  Select the table containing the values with which you want to populate this field.

Display

- **Span entire row on forms**

  Select this option to make this field the only field to appear on this row in the SBM User Workspace.

Search & Query

- **Appears in report field lists**

  Select this option to include the field in lists on report forms.

  Note: If this option is cleared after the field is used in a report (in the SBM User Workspace), the changed setting is ignored for that report; that is, the field will still appear until it is removed from the report definition.

  See General Tab of the Field Property Editor [page 140] for related information.

- **Appears on lookup form and relational field value lookup**

  Select this option to add the field to the Lookup form of the Advanced Lookup Tool and the *Relational Field Value Lookup* [page 694] form.

  Note: The field order for the Lookup and Relational Field Value Lookup forms is determined in SBM by the default field order of the table's first project in the project hierarchy. Projects are defined in SBM System Administrator.

- **Appears on searches for this table**

  Select this option to specify that the field will be available for selection on the Advanced Search page in the SBM User Workspace. This setting applies to primary and auxiliary tables. See Considerations for Advanced Search [page 176] for additional information.

- **Inactive items appear in selection lists**

  Select this option to specify that inactive items will be included in lists on transition forms and report forms.

- **On lookup or query-at-runtime forms**

  - **Allow searching**

    This option enables the *value find* and *relational field value lookup* [page 694] features on the Lookup form of the Advanced Lookup Tool, the Relational Field Value Lookup form, and Query-at-Runtime forms for reports. (See *Allow Searching* under *Style* above for a description of the *value find* feature.) The *relational field value lookup* [page 694] feature provides an advanced searching mechanism that lets users find values for *Single Relational* and *Multi-Relational* fields. If this option is enabled, a small magnifying glass appears next to the **Find** button in the SBM User Workspace.

  - **Show full list**

    This option lets users select a value from a drop-down list on the Lookup form of the Advanced Lookup Tool, the Relational Field Value Lookup form, and Query-at-Runtime forms for reports.

Multi-Relational Field Options

Multi-Relational fields let users select one or more values from a different table to associate with the current item.

Style

- **Allow searching**

Select this option to enable the *value find* [page 700] feature for the field on submit, transition, and update forms. This enables users enter to search criteria (an entire word, a few letters, or an asterisk) and then click the **Find** button (or press Enter) to perform the search. Results appear in a drop-down list, from which the user selects a value for the field. This option also enables users to search for values in the *Rich Editable Grid* [page 694].

Tip: If the field will contain 200 or more selections, this option is recommended for best performance in the SBM User Workspace. If you do not allow searching, fields that contain over 250 selection values are automatically set as searchable in the Editable Grid.

- **List size**

  Specify the number of rows in which to display values for the field on forms in the SBM User Workspace. Typically, you specify the number of expected values. Note that this option is unavailable when the **Checkboxes** option is selected.

- **List box**

  Select this option to let users select one or more values from a drop-down list populated with selections added to the *Relational* field table.

- **Checkboxes**

  Select this option to let users select one or more check boxes.

Values

Note: If these controls show "(External Application)" and "(External Table)," changing them could break relationships among applications. SBM Composer will prompt you to confirm the change.

- **Application**

  Select the *application* [page 679] containing the table of values with which you want to populate this field.

- **Table**

  Select the table containing the values with which you want to populate this field.

Display

- **Span entire row on forms**

  Select this option to make this field the only field to appear on this row in the SBM User Workspace.

Search & Query

- **Appears in report field lists**

  Select this option to include the field in lists on report forms.

  Note: If this option is cleared after the field is used in a report (in the SBM User Workspace), the changed setting is ignored for that report; that is, the field will still appear until it is removed from the report definition.

  See General Tab of the Field Property Editor [page 140] for related information.

- **Appears on lookup form and relational field value lookup**

  Select this option to add the field to the Lookup form of the Advanced Lookup Tool and the *Relational Field Value Lookup* [page 694] form.

  Note: The field order for the Lookup and Relational Field Value Lookup forms is determined in SBM by the default field order of the table's first project in the project hierarchy. Projects are defined in SBM System Administrator.

- **Appears on searches for this table**

  Select this option to specify that the field will be available for selection on the Advanced Search page in the SBM User Workspace. This setting applies to primary and auxiliary tables. See Considerations for Advanced Search [page 176] for additional information.

- **Inactive items appear in selection lists**

  Select this option to specify that inactive items will be included in lists on transition forms and report forms.

- **On lookup or query-at-runtime forms**

  - **Allow searching**

    This option enables the *value find* and *relational field value lookup* [page 694] features on the Lookup form of the Advanced Lookup Tool, the Relational Field Value Lookup form, and Query-at-Runtime forms for reports. (See *Allow Searching* under *Style* above for a description of the *value find* feature.) The *relational field value lookup* [page 694] feature provides an advanced searching mechanism that lets users find values for *Single Relational* and *Multi-Relational* fields. If this option is enabled, a small magnifying glass appears next to the **Find** button in the SBM User Workspace.

  - **Show full list**

    This option lets users select a value from a drop-down list on the Lookup form of the Advanced Lookup Tool, the Relational Field Value Lookup form, and Query-at-Runtime forms for reports.

Sub-Relational Field Options

Sub-Relational fields allow access to additional fields in a related *primary table* [page 692] or *auxiliary table* [page 680]. *Sub-Relational* fields are tied to a *Single-Relational* field already in an *application* [page 679].

For example, if you have an Incidents table that contains a *Single-Relational* field to an Issues table, you can include a *Sub-Relational* field called Owner that users can use to view the owner of a related issue within the incident.

Tip: You must create a *Single Relational* field before you can create a *Sub-Relational* field.

Values

- **Relational field**

Select the *Single Relational* field that you are associating with the *Sub-Relational* field.

Note: Selecting a relational field enables the Sub-Field control.

- **Sub-field**

  Select the field from the Relational Field table whose value you want to display in the SBM User Workspace. For example, you could create a *Single Relational* field from a *Companies* table, and then create two new *Sub-Relational* fields: *Company Address 1* and *Company Phone Number*.

  Note: This control is unavailable until a relational field has been selected above.

Display

- **Span entire row on forms**

  Select this option to make this field the only field to appear on this row in the SBM User Workspace.

Search & Query

- **Appears in report field lists**

  Select this option to include the field in lists on report forms.

  Note: If this option is cleared after the field is used in a report (in the SBM User Workspace), the changed setting is ignored for that report; that is, the field will still appear until it is removed from the report definition.

  See General Tab of the Field Property Editor [page 140] for related information.

- **Appears on lookup form and relational field value lookup**

  Select this option to add the field to the Lookup form of the Advanced Lookup Tool and the *Relational Field Value Lookup* [page 694] form.

  Note: The field order for the Lookup and Relational Field Value Lookup forms is determined in SBM by the default field order of the table's first project in the project hierarchy. Projects are defined in SBM System Administrator.

- **On lookup or query-at-runtime forms**

  - **Allow searching**

    This option enables the *value find* and *relational field value lookup* [page 694] features on the Lookup form of the Advanced Lookup Tool, the Relational Field Value Lookup form, and Query-at-Runtime forms for reports. (See *Allow Searching* under *Style* above for a description of the *value find* feature.) The *relational field value lookup* [page 694] feature provides an advanced searching mechanism that lets users find values for *Single Relational* and *Multi-Relational* fields. If this option is enabled, a small magnifying glass appears next to the **Find** button in the SBM User Workspace.

  - **Show full list**

This option lets users select a value from a drop-down list on the Lookup form of the Advanced Lookup Tool, the Relational Field Value Lookup form, and Query-at-Runtime forms for reports.

Attributes Tab of the Field Property Editor

This tab is available when you select a field (except for the *Sub-Relational* type) in a table in the table editor.

Element	Description
Defaults	**Default value**: Where available, specify the value (or values) to be preselected or pre-entered for this field. For some field types, default values must be specified in SBM System Administrator as part of project or auxiliary table configuration after the process app is deployed. For *Date/Time* fields, see Date/Time Values [page 166] for a description of your options for populating this field. **CAUTION:** The "(Auto)" string is not supported as a default value for *Date/Time* fields. **Backfill to existing items**: Specify that the default value will apply to any existing items that were created before the default value was set. This option is available for *Binary/Trinary*, *Date/Time*, *Multi-Selection*, *Numeric*, *Single Selection*, and *Text* fields.
Attributes	**Required**: Specify that users must set a value for the field. In the SBM User Workspace, the label of a required field typically appears to the user in red or green italics, though that can be modified in SBM System Administrator. **Allow mass update**: Make the field available when users *mass update* [page 689] items in the project or *auxiliary table* [page 680]. Mass updates let users transition, update, or delete multiple primary items simultaneously and update or delete multiple auxiliary items simultaneously. See Mass Update Feature [page 166] for related information. **Require appended text**: Specify that users must append text to the field during every transition. This option is available when the **Required** check box is selected, and when **Journal** is selected on the **Options** tab. **Read only**: Configure the field so that it can be viewed but not edited by users.

Date/Time Values

This topic describes the date/time tool that is displayed when you click the down-arrow next to the Default Value field for a *Date/Time* field on Attributes Tab of the Field Property Editor [page 165].

Note: You can type a default value in the field, but the date/time tool is a convenient way to prevent problems introduced by typing errors.

Element	Description
Clear	Click this button to clear the previously specified value from the **Default value** field, so no default is specified. You can also delete the text in the **Default value** field in the Property Editor without invoking the tool.
List of values (Special Value tab)	Select a time period like **This Week** or **Last Month** from the **Start Of** or **End Of** column. Though **Now** is displayed in the **Start Of** column, it represents the current time at the moment of transition. **Note:** Assuming a default value of **Now**, the value stored in the database is equal to the number of seconds that have passed since 12:00:00 a.m. in the submitter's or modifier's time zone. As an alternative to the values listed, you can type `plus` *n* or `minus` *n* directly in the **Default value** field to offset the value of `Today` by a number of days. You can specify a date up to 5000 days from today (`plus 1`, `plus 2`, ..., or `plus 5000`) or up to 1000 days before today (`minus 1`, `minus 2`, ..., or `minus 1000`). **Note:** Clicking the column heads *does not* sort the list Instead, it clears the **Default value** field.
Exact Value	Select this tab to display the calendar tool, which includes a time field. When you have set the date and time, click **Accept**. Click **Special Value** to return to the keyword tool.

Mass Update Feature

Mass update [page 689] enables users to make the same change to multiple items at the same time. For example, if several primary items have been deferred from the system, all the items can be reactivated at the same time by performing a mass "Restart" transition. This activates all the items at the same time and places them in the same state.

Mass update enables users to simultaneously transition, update, or delete multiple primary items and to simultaneously update or delete multiple auxiliary items. You can also allow specific fields to be modified when users perform a mass update. For example, if an *Additional Notes* field is selected for mass update, users can enter a value in the field before completing a mass update. This value overwrites any previous values stored in the

field for items that are mass updated. If a value is not provided for the field during the mass update, the original values remain unchanged.

To enable the mass update feature in SBM Composer, select the **Allow mass update** check box on the **Attributes** tab of the field Property Editor for specific fields. In SBM System Administrator, you can override this setting for default fields in projects and for transition fields.

You can control which users can perform mass updates by granting the *Mass Update Items* privilege. An option to update all selected items will be available to these users at the bottom of report and search lists in the SBM User Workspace.

Note: The Rich Editable Grid also allows users to update multiple items at the same time. The Editable Grid differs from Mass update in that it enables users to make a variety of changes to information in multiple items at the same time. The Editable Grid is available for Listing reports in the SBM User Workspace. Users who have privileges to update items in reports can switch to edit mode and modify items in the list.

Considerations for Using the Mass Update Feature

Note the following considerations:

- Post Item or Create Subtask transitions are not available for mass updates; Copy transitions are available for mass updates, however.
- Transitions that are set to require attachments are not available for mass updates.
- When you select the **Allow mass updates** check box for *dependent fields* [page 683], be sure to do the same for all independent fields that drive the dependent field's values. If you fail to do this, users will not have any options to select when mass updating items.
- Custom forms and any features they include are not available for mass updates. For example, AE Plug-In JavaScript API functions on a custom form cannot be used for mass updates. Such functions include field validation; auto-filled fields; information messages; and filters on *Multi-Relational*, *Single Relational*, *Sub-Relational*, *Multi-Selection*, and *Single Selection* fields.

Note: For instructions on performing mass updates, refer to the *Serena® Business Manager User's Guide*.

Dependencies Tab of the Field Property Editor

This tab is available when you select a *Single Relational*, *Single Selection*, or *User* field in a table in the table editor.

Element	Description
Dependent Fields [page 683] list	Select the fields that are dependent on the value selected in this field. See Using Field Dependencies [page 168] for details and examples.
Edit Value Restrictions	See Using Field Dependencies [page 168] for details and examples.

Using Field Dependencies

Set up field dependencies for independent *Single Selection*, *Single Relational*, and *User* fields on the **Dependencies** tab in the field Property Editor. On the **Dependencies** tab, you define auto-populated fields that tailor the selection lists for dependent *Single Selection*, *Multi-Selection*, *Multi-Group*, *Multi-User*, *Single Relational*, *Multi-Relational*, and *User* fields. When a user selects a value from the independent field, the values available in the dependent field are limited to those specified.

For example, you can create a *Single Selection* field called Product that has a selection for each product your company develops and a *Single Selection* field called Version that has a selection for each product version. You can set up dependencies for the *Products* field that tailor the selections available in the *Versions* field. When a user selects a product from the *Product* field, only applicable versions are available in the *Versions* field. In this case, *Products* is the independent field, and *Versions* is the dependent field. See Single Selection Field Dependency Tutorial [page 169] for instructions.

You can set up a similar dependency using a *Single Relational* field. For example, you can create two auxiliary tables, Products and Versions. You can then establish a relationship between the two tables that allows specific version records in the Versions table to be available for each product in the Products table. For example, if Version 1 and Version 2 apply only to Product A, those are the only available selections when Product A is selected in the *Product* field. See Relational Field Dependencies Tutorial [page 173] for instructions.

The following table lists the available independent *field types* [page 686] and the field types that allow dependencies.

Independent field type	Allowable dependent field types
Single Selection	Single Selection, Multi-Selection, Multi-Group, Multi-User, User
Single Relational	Single Relational, Multi-Relational
User	Single Selection, Multi-Selection, Multi-Group, Multi-User, User

Note: *Multi-Selection*, *Multi-Group*, *Multi-User*, and *Multi-Relational* fields can be used as *dependent fields* [page 683], but not as independent fields. Also, field dependencies do not work for *Multi-Selection*, *Multi-Group*, *Multi-User*, and *Multi-Relational* fields that are displayed as check boxes in the SBM User Workspace.

- Setting Default Values for Dependencies [page 169]
- Single Selection Field Dependency Tutorial [page 169]
- User Field Dependency Tutorial [page 171]
- Relational Field Dependencies Tutorial [page 173]

Setting Default Values for Dependencies

When you set up a dependency between two fields, you can also specify a default selection for the dependent field (as seen by a user in the SBM User Workspace). The following table describes how the different options affect the dependent field when the independent field is changed.

Option	Behavior when the independent field value is changed
<Leave entry unchanged>	The dependent field selection will not change (assuming the previously selected value of the dependent field is also valid for the new value of the independent field).
<First valid entry>	The dependent field will be preset to the item listed immediately after **<None>**.
<None>	The dependent field will be preset to **<None>**.
list of values	The dependent field will be preset to the *value* you select here.

Single Selection Field Dependency Tutorial

Prerequisites:

- You have an *application workflow* [page 680] created in SBM Composer and are familiar with basic *application* [page 679] tasks, such as adding fields and field selections.
- To test this tutorial in the SBM User Workspace, you must have privileges to deploy to a running *environment* [page 684].

This tutorial demonstrates how to configure field dependencies that tailor the selection list for a *Single Selection* field. You can adapt this example for other field dependencies as needed.

In this tutorial, you create a *Product* field that has two available selections, Product A and Product B. You set up dependencies that limit the selections available in the *Version* field based on the selection made in the *Product* field. For example, if a user selects Product A from the *Product* field, a set of available selections unique to Product A are listed in the *Version* field.

The steps in this tutorial are performed exclusively in SBM Composer. You can then log in to the SBM User Workspace to test the results.

Tip: Override dependent field selection for independent *Single Selection* fields in auxiliary tables in SBM System Administrator.

To establish a *Single Selection* field dependency for a primary table:

1. In SBM Composer, open a process app, select a *primary table* [page 692], and add a *Single Selection* field named *Product*.

2. On the **Options** tab for the *Product* field, add the following values:
   - Product A
   - Product B
3. In the same table, add another *Single Selection* field named *Version*.
4. On the **Options** tab for the *Version* field, add the following values:
   - 2.0
   - 2.5
   - 3.0
   - 3.1
5. On the **Dependencies** tab for the *Product* field, select **Version** from the list of fields list to set it as the dependent field.
6. Click **Edit Value Restrictions**, then select the workflow that contains the *Version* field. The **Dependencies** tab for the workflow is displayed.
7. Under **Select independent field**, make sure **Product** is selected.
8. Set up the dependency for Product A:
   a. Under **When this field has a value of**, select **Product A**.
   b. Under **Restrict dependent field**, select **Version**.
   c. Under **To these values**, select **2.0** and **2.5** (clear the **3.0** and **3.1** check boxes).
   d. *(Optional)* Under **Change Value To**, select **2.5**.

   Note: This sets a default selection for the **Version** field to the currently supported version. When a call comes in to Customer Support, and the support analyst creates an issue for that product, the analyst has to change the **Version** field only if the call pertains to the earlier release. See Setting Default Values for Dependencies [page 169] for details.
9. Set up the dependency for Product B:
   a. Under **When this field has a value of**, select **Product B**.
   b. Under **Restrict dependent field**, select **Version**.
   c. Under **To these values**, select **3.0** and **3.1** (clear the **2.0** and **2.5** check boxes).
   d. *(Optional)* Under **Change Value To**, select **3.1**.
10. Deploy the changed process app.
11. When *deployment* [page 683] is complete, log in to the SBM User Workspace as a user who has privileges to view and update items in the application that contains the field dependency you added.

12. Submit an item into a project associated with the application workflow that contains the field dependency.

Selections available in the **Version** field are now dependent on what the user selects from the **Product** drop-down list. In this example, when **Product A** is selected, the **Version** field defaults to **2.5**, and the user can select **(None)**, **2.0**, or **2.5**. When **Product B** is selected from the **Products** list, the **Version** field defaults to **3.1**, and the user can select **(None)**, **3.0**, or **3.1**.

User Field Dependency Tutorial

Prerequisites:

- This tutorial assumes you are familiar with basic application tasks, such as adding fields and field selections, and with checking in, publishing, and deploying process apps.
- You must use SBM Composer and SBM System Administrator to set up the dependency. To complete this tutorial, you must have privileges to deploy to a running environment.
- Your application should contain two *User* fields with valid user selections. The fields should be called *Manager* and *Employee*. These users should have privileges to view, submit, and update items in the process app.

The following tutorial explains how to configure field dependencies that tailor the selection list for a *User* field. Dependencies will be set that limit the selections available in an *Employee* field depending on the selection made in the *Manager* field. For example, if a user selects Kathy Manager from the *Manager* field, Chris Tester and Hans Tester are available selections for the *Employee* field; if the user selects Joe Manager from the *Manager* field, Laura Engineer and Newton Engineer are available as selections for the *Employee* field.

Establish the Dependency in SBM Composer

In this step, you will create two *User* fields and establish a dependency between those fields.

To establish a dependency for a *User* field:

1. Open a process app in SBM Composer.
2. From App Explorer, select the primary table for your application.
3. Add a *User* field named *Manager*.
4. Add another *User* field named *Employee*.
5. Select the *Manager* field, and then select the **Dependencies** tab.
6. In the list of available dependent fields, select *Employee*.
7. Save, check in, publish, and deploy your changes.

Add User Selections for the Independent and Dependent Fields

In this step, you will add user selections to the fields created in the previous step.

To set user selections:

1. In SBM System Administrator, select the **Workflows** tab, select a workflow in the application for which you established the dependency, and then click **Edit**.
2. Select the **Default Fields** tab.
3. Select the *Manager* field, and then click **Edit**.
4. Select the **Options** tab, click **Add**, and then add Kathy Manager and Joe Manager as selections.
5. Click **OK**.
6. Select the *Employee* field, and then click **Edit**.
7. Select the **Options** tab, click **Add**, and then add Laura Engineer, Newton Engineer, Chris Tester, and Hans Tester as selections.
8. Click **OK**.

Set Dependent Field Selections

In this step, you will specify which field selections are available for the dependent field based on the selection users make for the independent field. For example, when a user selects Kathy Manager from the *Manager* field, Chris Tester and Hans Tester are the only selections available in the *Employee* field.

To set dependent field selections:

1. In SBM System Administrator, select the **Projects** tab.
2. Select a project assigned to the workflow you modified in the previous steps, and then click **Edit**.
3. Select the **Default Fields** tab, select the *Manager* field, and then click **Edit**.
4. Select the **Dependencies** tab.
5. Select Joe Manager, and then click **Dependencies**.
6. Select the **Override** check box, and then clear the check boxes next to Chris Tester and Hans Tester, and then click **OK**.
7. Select Kathy Manager, and then click **Dependencies**.
8. Select the **Override** check box, and then clear the check boxes next to Laura Engineer and Newton Engineer, and then click **OK**.
9. Save the changes by clicking **OK** on all project dialog boxes.

Test the Dependency

To test the dependency:

1. Log into the SBM User Workspace as a user who has privileges to the process app that contains the dependency.
2. Submit an item into the project that you modified in the SBM System Administrator in the previous steps.
3. On the **Submit** from, select Kathy Manager from the *Manager* field. Notice that Chris Tester and Hans Tester are available as selections for the *Employee* field.
4. Select Joe Manager from the *Manager* field. Notice that Laura Engineer and Newton Engineer are available as selections for the *Employee* field.

Relational Field Dependencies Tutorial

Prerequisites:

- You are familiar with basic *application* [page 679] tasks, such as adding tables, and with checking in, publishing, and deploying process apps.
- You can use the SBM User Workspace to finish setting up the dependency (adding items to auxiliary tables using the Manage Data feature).
- You have the necessary privileges to deploy to a running *environment* [page 684].

This tutorial demonstrates how to configure field dependencies that tailor the selection list for a *Single Relational* field added to a workflow. You can modify this example for other field dependencies as needed.

In this tutorial, you create two auxiliary tables, Products and Versions. You then establish a relationship between the two tables that allows specific version records in the Versions table to be available for each product in the Products table. For example, if Version 1 and Version 2 apply only to Product A, those versions are the only available selections when Product A is selected from the Products field in the SBM User Workspace.

To establish a *Relational* field dependency:

1. In SBM Composer, create two tables: Products and Versions. For this example, Products is the independent field table, and Versions is the dependent field table.
2. Add a *Single Relational* field of the independent field type to the dependent field table. For this example, add a *Single Relational* field to the Versions table, and select Products from the **Table** drop-down list on the **Options** tab.

   Tip: If you are using a custom form, be sure to add each new field to the form before you deploy the process app. If you are using the *quick form* [page 693], new fields are added automatically.
3. Open the *primary table* [page 692] for the application, and add a *Single Relational* field to serve as the independent field. For this example, name the field **Products**, and select **Products** from the **Table** drop-down list on the **Options** tab.
4. Still in the primary table for the application, add another *Single Relational* field to serve as the dependent field. For this example, name the field **Versions**, and select **Versions** from the **Table** drop-down list on the **Options** tab.

5. Select the *Single Relational* field that serves as the independent field. For this example, select the **Products** field.
6. On the **Dependencies** tab, select the Versions table from the list of *dependent fields* [page 683].
7. Still on the **Dependencies** tab, click **Edit Value Restrictions**, and select the workflow for which you want to restrict dependent field values.
8. On the **Dependencies** tab for the workflow:
   a. Under **Select independent field**, make sure **Products** is selected.
   b. Under **Restrict dependent field**, select **Versions**.
   c. Under **By matching these columns**, select the field you created in step 2, above.
9. Grant privileges to these tables. You can do this in SBM Composer by granting *role* [page 695] privileges for each table. Alternatively, you can assign user or group privileges to the table in SBM System Administrator. For this tutorial, users should be able to submit, update, and view items in both tables.
10. Deploy the changed process app.
11. When the *deployment* [page 683] is complete, log on to the SBM User Workspace as a user who has privileges to manage the Products and Versions tables.
12. Populate the Products table with items. For this example, add **Product A** and **Product B** items.
13. Populate the Versions table with the following items. When you add these items, select the listed value from the **Products** field:
    - Version 1 – Product A
    - Version 2 – Product A
    - Version 3 – Product B
    - Version 4 – Product B
    - Version 5 – Product B
14. Open the **Submit** form for a project used by the workflow in which you created the dependencies.
15. Click the **Find** button for the **Products** field. Notice that **Product A** and **Product B** are available as selections.
16. Select **Product A**. Notice that for the **Versions** field, **Version 1** and **Version 2** are available as selections in the **Versions** field.
17. Select **Product B** from the **Products** field. Notice that **Version 3**, **Version 4**, and **Version 5** are available as selections in the **Versions** field.

Deleting and Restoring Fields

SBM Composer enables you to delete all *custom fields* [page 682] from the *primary table* [page 692] and auxiliary tables of published process apps. To delete a field, select the field in the table editor and then press the Delete key, or right-click the field and then select **Delete**. When you delete a field, it appears dimmed in the table editor, but is not removed.

Important: System fields cannot be deleted.

You can restore a deleted field. When you do so, its properties are also restored. For example, when you restore a *Multi-Selection* field, all of the values that can be selected from it in the SBM User Workspace are restored. To restore a field, right-click the field and then select **Restore**.

Modifying Locked Fields in a Published Process App

After a process app is published, certain items are locked and cannot be modified. This is because data that depends on those items could have been created in the SBM User Workspace. For example, you cannot delete a state because it could still exist in the state history for one or more items, and you cannot change the style of a numeric field because items could already exist in that table with that field set to a number with the original style.

However, if you intend to deploy the process app to a different *environment* [page 684], you can "reset" published items so they can be modified. Suppose you have been deploying to a development environment while you are designing and testing a process app. During this phase, you discover that you need to change a *Numeric* field from an integer style to a floating point style before you deploy to the production environment. This topic lists the items that are locked, and describes how to reset the process app so the items can be modified.

Note: If you unlock a process app, modify a previously-locked field, and then deploy the process app to an existing environment, the change is ignored. To restore the original value, perform the procedure in this topic, and change the field value to its original value.

The following list describes the items that are locked after a process app is published:

- Fields, selection values, and states cannot be *permanently* deleted. They are instead *marked* deleted (or disabled, in the case of states).
- The **Backfill to existing items** check box is not available for *Multi-Selection* fields.
- Some styles options for *Numeric* and *Date/Time* fields cannot be changed.
- *Text* fields cannot be changed to the **Fixed length** style from the **Memo** or **Journal** style and cannot be changed from the **Fixed length** style to the **Memo** or **Journal** style.
- The related table for *Relational* fields cannot be changed.
- The database name for fields cannot be changed.
- The prefix for applications cannot be changed.

To reset a process app so you can modify locked items:

1. Export the process app. From the Composer menu, point to **Import and Export**, and then select **Export to File**.
2. Import the process app. From the Composer menu, point to **Import and Export**, and then select **Import from File**.
3. Modify items as needed.
4. Deploy to a different environment.
   a. From the Composer menu, point to **Deploy** and then select **Deploy**.
   b. In the **Deploy Process App** dialog box that opens, select a different environment in the **Deploy to** list, and complete other fields as needed.

Considerations for Advanced Search

Use the **Appears on searches for this table** option on Options Tab of the Field Property Editor [page 140] to specify that a field should appear on the Advanced Search page. Here are some things to remember as you specify fields for Advanced Search:

- Fields you select for Advanced Search are visible by all users who can use this feature.
- The **Keyword(s)** and **Project(s)** options are always available on the Advanced Search page.
- For newly created tables, no fields are set to display on the Advanced Search page.
- The field order for the Advanced Search page is determined by the default field order of the first project in the table in the project hierarchy. You can change the field order in SBM System Administrator.
- The project selected by your users determines which values are available for selection *field types* [page 686], such as *User* fields, but project selection has no effect on *relational fields* [page 694].
- Selection field types are automatically set to searchable if the number of selections exceeds the number specified in SBM System Administrator.
- To improve performance, you should limit the number of selection fields you include on the Advanced Search page.
- While you can include *Text* fields to the Advanced Search page, it could improve usability to include them in keyword searches instead, because this is their default setting. See Text Field Options [page 146] for information about the **Include field in keyword searches** option.
- To improve performance, you could limit the number of *Text* fields you allow for keyword searching. You can add individual *Text* fields to the Advanced Search page, but they are treated a bit differently than keyword searching. Items containing the search criteria are returned, but users must search for exact phrases. In addition, wildcard characters do not apply.

Chapter 11: Defining Application Reports

The following topics describe how to define *application* [page 679] reports:

Introducing Application Reports

Application [page 679] reports are *listing* reports that you define in SBM Composer along with other process app artifacts. (Listing reports return textual lists of primary or auxiliary items based on the display, sorting, and search options you select.)

Application reports are a starting point for users to pre-configure regular reports. Users can generate any number of regular reports from a single *application report* [page 679].

Application reports contain all of the settings that are part of a regular listing report. They also contain search criteria and fields that use the "Query at runtime" condition if potential values, such as user information, is not known to SBM Composer. This information is obtained from the user in the SBM User Workspace in response to prompts.

Note: For more information about the "Query at runtime" condition, see Basic Conditions [page 189].

Project names are based on *application workflow* [page 680] names. Application reports are created for base projects from the report definition defined in SBM Composer when the process app is deployed.

Users can run application reports from the SBM User Workspace. They can modify and save application reports as regular reports, but cannot save them as application reports. Application reports are purely design time artifacts, and to change them you must use SBM Composer and redeploy the changes.

Users with the appropriate privileges can find application reports by clicking the **Reports** filter in the SBM User Workspace, and then clicking **Browse Application Reports** under the **Advanced Tasks** heading. Application reports that users have privileges to see are displayed in the content pane. The privilege level for a *role* [page 695] determines whether a user can see an application report and is set in SBM Composer when the process app is created. The privilege level is a combination of the "Run reports" privileges set for the table, the **Privilege category** set in the definition of the application report, and table and workflow privileges.

Note: Application reports are automatically created, updated, and deleted as a result of deploying a process app, and are not promoted like regular reports.

Use Case

You create a process app and need an application report that a user (for example, an administrator in this case) can use to create various regular reports that pertain to the process app. Your application report includes an *Owner* field. You use the "Query at runtime" condition on this field. This means that the administrator is prompted to specify an owner for the report when he or she runs the report. The administrator then saves the report as a regular report, which other users can run.

Report Definition Editor

Use the report definition editor to design *application* [page 679] reports.

In the report definition editor, drag fields to the **Columns** and **Sort Order** blocks, and drag fields and operators to the **Search Filter** block to create the format and logic of the *application report* [page 679]. Additional options are available when you right-click in the report definition editor and right-click its fields and operators. For example, in the **Columns** block, you can right-click a column and select **Move Down**. This moves the column to the immediate right.

If you selected the **Always perform a primary sort by project** check box on the **Options** tab in the report definition Property Editor, **Project (hierarchy)** is the first sort criteria in the **Sort Order** block. This is relevant if you include multiple projects in the report search filter.

The **Associated Attachments** column in the **Columns** block is used to display any attachments that are present (for example, items, notes, URLs, and files). The **Search Filter Summary** block contains a read-only query string that represents the search filter as you build it.

The name of the report definition is displayed on the tab in the report definition editor. The name of the table the report is associated with is displayed in parentheses after the report definition name.

Report Palette

The **Report Palette** includes the following types of controls:

- **Logical Operators** include AND and OR search operators. These operators are used to specify search operators for fields and to group conditions to set a sequence for evaluating conditions. Operators can only be added to the **Search Filter** block.
- **Fields** include the fields that can be queried. These fields are in the *primary table* [page 692]. If you add or remove fields from the primary table, they are added or

removed from this section. Fields can be added to the **Columns**, **Search Order**, or **Search Filter** blocks.

Report Definition Property Editor

Use the following tabs in the report definition Property Editor to view and modify report options.

- General Tab of the Report Definition Property Editor [page 179]
- Options Tab of the Report Definition Property Editor [page 179]
- Calculations Tab of the Report Definition Property Editor [page 181]

General Tab of the Report Definition Property Editor

The following table describes the information and options that are displayed on the **General** tab of the report definition Property Editor.

Form

Element	Description
Name	The name of the report definition that was provided when the report definition was created. **Primary** or **Auxiliary** is appended to the name in the report definition editor to indicate the type of table that the report is created against.
Uses table	The table (primary or auxiliary) that the report is created against.
Type	**Listing Report** is displayed as a read-only field, because listing reports are the only type of report you can create from SBM Composer.
Privilege category	Select **Guest**, **User**, or **Manager**. This determines the report privileges. Privileges determine which users can access the different levels of reports.
Description	Type a description of the report definition.

Options Tab of the Report Definition Property Editor

The following table describes the information and options that are displayed on the **Options** tab of the report definition Property Editor.

Element	Description
Searching and Sorting	**Include items from sub-projects**—Select this check box to include subprojects of the selected project in the query. If one of your *preferred projects* [page 691] is a parent project, but its subprojects are not in your preferred projects list, subprojects of the parent are searched if this check box is selected. **Enable dynamic column sorting**—Select this check box to display column headers as links. Click to sort the data in ascending or descending order. The administrator determines if this option is available. If the listing report is part of a multi-view report, dynamic column sorting is not available. **Always perform a primary sort by project** —If you include multiple projects in the report search filter, select this check box to sort items by project hierarchy. If you clear this check box and do not provide other sorting criteria, items are listed randomly.  **Note:** These check boxes are disabled for reports that use auxiliary tables.
Display	**Hide project titles**—Select this check box to prevent the project hierarchy headers from being displayed in the report. This check box is disabled for reports that use auxiliary tables. **Remove line breaks from memo/text fields**—Select this check box to remove line breaks from *Text* and *Memo* fields in the report. This option is useful if you plan to export the report data to another application, such as Microsoft Excel.
Columns of Linked Data	**Include linked files from attachments**—Select this check box to display links to any files attached to items. Links are displayed only if users have privileges to view attachments. **Include linked URLs from attachments**—Select this check box to display any URLs attached to items. Links are displayed only if users have privileges to view attachments. **Include linked notes**—Select this check box to display any notes attached to items. Links are displayed only if users have privileges to view notes. **Include linked items**—Select this check box to display any links to other items associated with the item being viewed. Users can click the item link to view the item in the Item Details pane in the SBM User Workspace. Links are only displayed if users have privileges to view the linked item.  **Note:** If any of these check boxes are selected, an additional column labeled **Associated Attachments** appears at the end of the report columns.

Calculations Tab of the Report Definition Property Editor

Users can include calculations on *Numeric*, *Binary/Trinary*, and *Date/Time* fields in their report results. The available calculations are addition, subtraction, multiplication, and division.

To add a calculation to a report:

1. Select the **Add column** check box.
2. In the **Display** box, type the name that will be displayed as the Custom Header in the report.
3. In the **Formula** drop-down list, do one of the following:
   a. Click the **Fields** tab and select a field that will serve as the first operator.
   b. Click the **Special Date** tab and select a value such as **Start of Next Week** or **End of This Month**.
4. Select an operand from the next drop-down list.
5. In the second calculation drop-down list, select a field that will serve as the second operator.

The result type is displayed in the **Result Type** column. For example, if you specified **Description = Last Modified Date - Submit Date**, the **Result Type** is **Elapsed Time**.

Note: Each selected calculated column is added to the field palette. To appear in the report, they must be added to the **Columns** or **Sort Order** block.

Creating an Application Report

An application report is a listing report that provides a list of primary or auxiliary items that meet your report criteria. Users view the list of items returned by the report in the **Item List** pane in the SBM User Workspace and then click an item link to view detailed information in the **Item Details** pane.

To create an *application report* [page 679]:

1. To create a report for a *primary table* [page 692], perform the following steps:
   a. Click the **Reports** filter in App Explorer.
   b. Right-click **Report Definitions** and select **Add New Listing**.

Note: Alternatively, you can click **Element** in the **New** area of the Ribbon, and select **Listing**, or right-click a primary table in App Explorer and select **Create Report Definition for this table**.

2. To create a report for an *auxiliary table* [page 680], right-click the auxiliary table in App Explorer and select **Create Report Definition for this table**.
3. The three blocks described in the following table are displayed in the report definition editor.

Section	Description
Columns	Contains options that let you select which columns to display in the report. For more information, see Selecting Fields to Display as Columns [page 182].
Sort Order	Lets you specify sorting options for the items returned in the report. For more information, see Sorting Search Results [page 183].
Search Filter	Lets you narrow your search for items. For more information, see Using Search Filters [page 184].

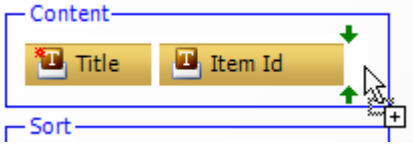
Selecting Fields to Display as Columns

The **Columns** block lets you select which fields to display as columns in the report. The *Item Id* and *Title* fields are included by default, but can be removed.

Tip: The fields you select to display as columns in a report can be different from the fields you select for sorting or the fields you use in the filter.

To select the fields to display as columns in the report:

1. Drag a field from the **Report Palette** to the **Columns** block. Green up and down arrows indicate where the field is to be dropped if you release the mouse.



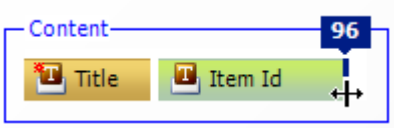
2. Release the mouse when you are satisfied with the location of the field in the block.

   Note: You can have only one instance of a field in the **Columns** block. If you drag a field that is already in the block, the field moves to the right side of the block and is selected (green).

3. If you change your mind and want to delete a field, select the field and then press the Delete key.

4. To change the order of the fields, select a field and use the drag-and-drop operation to move it.

5. To change the width of a column, select the right edge of the field and drag it until it is the size you want. The width of the column, in pixels, is displayed at the top right corner of the field. Use the splitter cursor to drag the dotted line to the right or to the left. When the field reaches its minimum width, **min** is shown instead of the number of pixels.



Note: The minimum width is determined by the display text. You can reset this to auto size by dragging the right edge of the field to the left or by right-clicking the field and selecting **Reset to Auto Size**.

Sorting Search Results

Sort options let you sort search results based on values in selected fields. For example, you can sort items by the state they are in or by their active or inactive status.

Note: The **Project (hierarchy)** field is displayed is displayed if you select **Always perform a primary sort by project** on the **Options** tab of the Property Editor.

Tip: The fields you select for sorting can be different from the fields you select to display as columns or the fields you use in the filter.

To sort search results:

1. Drag a field from the **Report Palette** to the **Sort By** field in the **Sort Order** block.

   Note: The sort fields cannot include *MultiGroup*, *MultiRelational*, *MultiSelection*, *MultiUser*, or non-fixed length *Text* fields. *SubRelational* fields use the field type that they reference, and follow the rules stated in the previous sentence.

2. If you want to sort by additional fields, drag them to the **then by** fields. You can specify a maximum of four fields.

3. To toggle the sort order for each field, click the up or down arrow next to the field. This determines whether the search results in each column are sorted in ascending or descending order.

Using Search Filters

A search filter is a collection of conditions and logical operators. Search filters (also known as "expressions") let you narrow your search for items. You can define a search filter by making selections from the list of operators and fields in the **Report Palette**. In expressions, fields are known as "conditions."

For example, to create a Listing report that includes items that are in the **StartState** state and that are assigned to the current user, create the following expression:

State = StartState

AND

Owner in (Current User)

Tip: The fields you select for filtering can be different from the fields you select to display as columns and the fields you use to sort returned data.

To use search filters:

1. Drag a field from the **Report Palette** to the **Search Filter** block.
2. Some operators can be changed. Click the field and select another operator from the popup window that opens. For a description of operators, see Report Operators [page 184].
3. Select the field value or field values for the condition. This creates an expression. For example, if you are working with an *Approvers* field, the names that are available as selections for the field are displayed in the menu that opens. You can select one or more names from the list. Field values are described in Custom Fields [page 134].
4. To group expressions to set a sequence for evaluating conditions, drag a logical operator from the **Report Palette** onto the expression. This inserts the logical operator in the place where the expression was and moves the expression to the first child of the new operator. For more information about using the drag-and-drop operation in various scenarios, see Drag-and-Drop Behavior [page 193]. For examples of report logic, see Report Logic Examples [page 192].

You can view a query string that represents the search filter in the **Search Filter Summary** block.

Report Operators

Logical operators let you set a sequence for evaluating conditions. *Condition* operators let you select a function for a selected field.

In the report definition editor, **Search Filter** block, AND operators are purple and OR operators are blue. The selected operator and line are lighter in color than unselected operators and lines.

Logical Operators

The following table describes the AND and OR logical operators.

Operator	Description
AND	Drag **AND** from the **Logical Operators** section of the **Table Palette** onto the condition. The AND operator returns all items that meet all conditions defined in the search parameters. For example, the conditions "Owner in Joe Manager" AND "State in New" return all items that are in the "New" state *and* are owned by Joe Manager.
OR	Drag **OR** from the **Logical Operators** section in the **Table Palette** onto the condition. The OR operator returns any items that meet the conditions defined in the search parameters. For example, the conditions "Owner in Joe Manager" OR "State in New" return any items that are in the "New" state *or* are owned by Joe Manager.

Condition Operators

The condition operators let you select a function for the field that is selected in the report editor. The following table describes the operators that are available for different *field types* [page 686].

Operator	Description
= (equal to)	Use the equal to operator to find exact values. For example, select the *Active/Inactive* field with the = operator and the Active value to display all active items in the specified project. For *Date/Time* fields, dates used with this operator are treated as a Date-only field. For example, select the *Submit Date/Time* field with the = operator and type yesterday's date in the **Value** menu to find all of the items submitted into the specified project yesterday.
> (greater than)	Use the greater-than operator to find larger values than the value specified. For example, select the *Submit Date/Time* field with the > operator and type the last day of last year in the **Value** menu to find all of the items submitted into the specified project for this calendar year. Use this operator with the less-than operator to define all items between two specified dates.
< (less than)	Use the less-than operator to find smaller values than the value specified. For example, select the *Submit Date/Time* field and the < operator, and then type 2/27/2009 in the **Value** menu to return all items with a submit date and time up to 2/26/2009 11:59:59 p.m. Use this operator with the greater than operator (>) to define all items between two specified dates.

Operator	Description
>= (greater than or equal to)	User the greater-than-or-equal-to operator to find identical and greater values than the value specified. For example, select the *Submit Date/Time* field with the >= operator and type the last day of last year in the **Value** menu to find all of the items submitted into the specified project this calendar year. Use this operator with the less-than sign to define all items between two specified dates.
<= (less than or equal to)	Use the less-than-or-equal-to operator to find identical and lesser values than the value specified. For example, select the *Submit Date/Time* field with the <= operator and type the first day of this year in the **Value** menu to find all of the items submitted into the specified project before this calendar year.
<> (not equal to)	Use the not-equal-to operator to find values not equal to the value specified. For *Date/Time* fields, dates used with this operator are treated as Date-only fields. For example, select the *Submit Date/Time* field with the <> operator and type a date in the **Value** menu to find all items that were not submitted into the specified project on that date. For *Date/Time* fields, dates used with this operator are treated as Date-only fields.
in	Use this operator to select a single value or multiple values as criteria and return items for which either one or more values are selected for the field. For example, for the *Last State Changer* field, select two users from the **Value** menu and "In" from the operator drop-down list to display the items that last changed state by either user.
not in	Use this operator to select values that are not selected for a field. For example, for an *Engineer* field, select a user from the **Value** menu and "Not In" from the operator drop-down list to display items for users other than the engineer selected for your search criteria.
contains all	Use this operator to select one or more values to return items that contain all values in the field. For example, select the "v2.0", "v2.1," and "V3.0" values as your search criteria for a *Change in Version* field, and then select "Contains All" from the operator drop-down list. Items in which "v2.0", "v2.1," **AND** "V3.0" are selected for the *Change in Version* field are returned. Multiple search conditions are allowed.

Operator	Description
contains any	Use this operator to select one or more values to return items that contain any values in the field. Multiple conditions are allowed. For example, select the "v2.0", "v2.1," and "V3.0" values as your search criteria for a *Change in Version* field, and then select "Contains Any" from the operator drop-down list. Items in which "v2.0," "v2.1," **OR**"V3.0" are selected for the *Change in Version* field are returned. Multiple search conditions are allowed.
does not contain all	Use this operator to select one or more values to return items that do not contain all specified values in the field. Multiple conditions are allowed. For example, select the "v2.0", "v2.1," and "V3.0" values as your search criteria for a *Change in Version* field, and then select "Does Not Contain All" from the operator drop-down list. Items in which "v2.0," "v2.1," **AND** "V3.0" are not selected for the *Change in Version* field are returned. Multiple search conditions are allowed.
does not contain any	Use this operator to select one or more values to return items that do not contain any values in the field. Multiple conditions are allowed. For example, select the "v2.0," "v2.1," and "V3.0" values as your search criteria for a *Change in Version* field, and then select "Does Not Contains Any" from the operator drop-down list. Items in which "v2.0," "v2.1," **OR** "V3.0" are not selected for the *Change in Version* field are returned. Multiple search conditions are allowed.
contains	Use this operator to search for keywords in a *Text* field or *Sub-Relational* field or the Item Type Prefix option. SBM automatically includes wildcard characters at the beginning and the end of the search criteria. For example, type the value *icons* for a title search and the "contains" operator to return all items that contain the word "icons" in the title. Use this operator to search for exact phrases or single keywords in the *Text* field, or for searching by items by a specific item prefix. Use this operator to search for exact phrases or single keywords in the *Text* or *Sub-Relational* field, or for searching by items by a specific item prefix.
like	Like is a comparison expression that returns data that is like the value selected. This operator gives you complete control over how SBM uses wildcard characters. For example, select the *Title* field and the "Like" operator, and then type a word in the **Value** menu followed by an asterisk to display items that have this word only at the beginning of the title. Use this operator to search for multiple phrases or keywords throughout any *Text* or *Sub-Relational* field or the Item Type Prefix option. For example, select the *Title* field and the "like" operator, and then type *change*modern* to return items that contain both the words "change" and "modern" in the title, and where "change" precedes "modern."

Operator	Description
not contains	Use this operator to return items that do not contain specified keywords in the queried *Text* or *Sub-Relational* field or the Item Type Prefix option. When you run the report, SBM automatically includes wildcard characters at the beginning and the end of the search criteria. For example, type the value *icon* and use the "Not Contains" operator for a title search to return items that do not contain the word *icon* in the title. Use this operator to search for exact phrases or single keywords not in the field or item prefix.
not like	This is a comparison expression that returns data that is not like the selected value. Select the **Title** field and the "not like" operator, and then type the first or last word of the title to exclude that item from the report. You can use an asterisk (*) in the search as a wildcard character. For example, type the value `*icons*` for a title search to return items that do not contain the word icons. Use this operator to search for multiple phrases or keywords not in the field or item prefix.
like (zero-filled)	Use this operator to include leading zeros in your criteria for the Item ID field. For example, if there are leading zeroes on the ID number, they may be left off your search criteria. For example, to find item ID number "BUG00017," type **17**. To search for multiple items by Item ID, separate each Item ID with a space.
not like (zero-filled)	Use this operator to not include leading zeros in your criteria for the *Item ID* field. For example, if there are leading zeroes on the ID number, they must be included in your criteria. For example, to find item ID number "BUG00017," you must type `00017`.
: (colon)	Use the : (colon) operator to specify all keywords that should be included in a search against all *Text* fields enabled for searching by your administrator. For example, if you select the **Text Fields With All Keywords** option and type graphic and file as field values, all items that contain the word graphic AND file are returned. The colon can be used on Text Fields With All Keywords option
: (colon)	Use the : (colon) operator to specify any keyword that should be included in a search against all *Text* fields enabled for searching by your administrator. For example, if you select the **Text Fields With Any Keywords** option and type graphic and file as field values, all items that contain the word graphic OR file are returned. The colon can be used on Text Fields With Any Keywords option

Available Operators

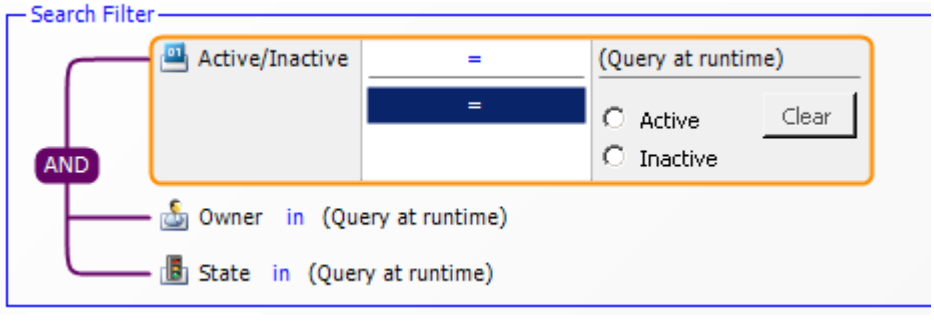
The following table shows the operators that are available for each field type.

Field Type	Operators
Binary	=
Company, Contact, Folder, Item Type, Project, Single Relational, Single Selection, State, Sub-Relational, Trinary, User	in, not in
DateTime, Numeric, Summation	=, <>, >, >=, <, <=
Multi-Group, Multi-Relational, Multi-Selection, Multi-User	contains all, contains any, does not contain all, does not contain any
Text, Item Type, Prefix Option	contains, like, not contains, not like
Item Id	contains, like, like (zero-filled), not contains, not like
Sub-Relational	The available operators depend on the selected sub-field type.

Basic Conditions

Reports that use basic conditions let you define search filters by selecting fields and search criteria for those fields. When you click a field that you dragged from the **Table Palette** to the **Search Filter** block, a panel opens that contains the conditions that make up the search criteria for your report. An expression is created after you define the search criteria. Multiple expressions are linked with a solid line and contain either an AND or an OR operator. You can select an expression to edit it.

Note: For information about *custom fields* [page 682] and custom *field types* [page 686], see Custom Fields [page 134]. For information about operators, see Report Operators [page 184].



The type of field that you drag determines the options and values that are available to you. The following list describes common features of conditions.

- **Current User** – All *User*, *Multi-User*, and *Multi-Group* fields have a special "Current User" field value that matches all records where the user running the report appears

in that field. For example, add the *Submitter system field* [page 699] from the **Fields** section of the **Report Palette**. Select the **in** operator, and then select the **Current User** check box.

This expression will match only records in which the user running the report (the "Current User") submitted the issue. In *Multi-Group* fields, a field is matched only when the current user is a member of one or more groups associated with the field.

- **None** – With certain field types, you can select the **None** check box. This selection returns all items meeting the rest of the report criteria that do not have a value for the selected field.
- **Query at runtime** – All fields, except *Summation* and *Numeric* fields, have a special "query at runtime" field value. You can use this when the value you want to use for the search filter is not known at design time. The user will be prompted to select the value when the report is run.

  For example, a *User* field contains users that are defined in SBM System Administrator after *deployment* [page 683] and therefore cannot be specified at design time. You use the "query at runtime" value in SBM Composer, and then, after the process app is deployed, users can set the values when they create a regular listing report from the *application report* [page 679].
- **Clear** – Most fields have a **Clear** button in the popup window. Use this button to clear selected values or to return to the default value of **Query at runtime**.

The following table lists the field types and describes the options that are available to them.

Field Type	Options
Selection, User	If you can select values for a field, you can search for specific values or select multiple values to add to your condition. For example, if you select an *Approvers* field, the names that are available as selections for the field are available in the **Available Values** list to the right of the operators. You can select approvers from the list to add them to your query. If you select a selection field, the values that are available as selections for the field are available in the **Available Values** column. Use the right arrow button to move them to the **Selected Values** column. In some cases, a text box contains **Type here to search for a value**. You can type a few letters in this box to filter the list of values. When you do this, the only values that are displayed are those that contain the letters you typed.

Field Type	Options
Date/ Time	Select an option, such as **Start of Last Week** on the **Special Value** tab. Alternately, on the **Exact Value** tab, click a date and select a time, and then click **Accept**. If you want to specify an absolute number, select the **Now** checkbox, select **+** or **-**, and type or select the number of days. For example, a Help Desk manager wants a report showing all incidents that were submitted in the last 30 days. You select the **≤** operator, select the **Now** check box, select **–**, and then type 30 in the **days** combo box. The expression becomes **Date ≤ 30 days ago.**.
Relational	SBM Composer does not have access to runtime data. Therefore, the only options are **Query at runtime** and **None**.
Non-Selection	If a field does not let you select a specific value, type a value in the text box. For example, if you select a *Text* field, such as the *Description* field, type a keyword or keywords to search for. *Date/Time*, *Numeric*, and *Summation* fields also let you specify a value. In some cases, the text box contains **Enter a wildcard pattern**. For information about using wildcards, see Wildcards [page 191].
Binary	If you select a *Binary* field, such as the *Active/Inactive* field, select one value for your search criteria.
Trinary	If you select a *Trinary* field, select one, two, or three values for your search criteria.

Wildcards

You can use wildcard characters to search for items and field values if **Enter a wildcard pattern** is in the text box in the search filter for the field.

The following guidelines apply to wildcard characters:

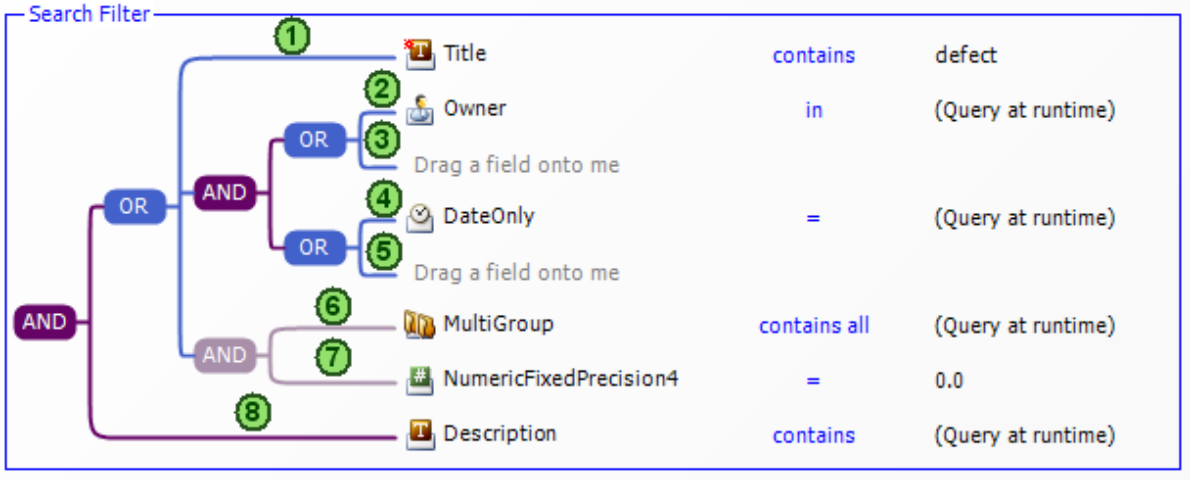
- Asterisks (*) and percent signs (%) serve as wildcard characters. A wildcard character matches zero or more consecutive characters.
- Underscores (_) match a single character.
- Wildcard characters are automatically applied to the beginning and end of your criteria. Items containing all of your criteria are returned.
- You can override automatic wildcards by including at least one wildcard in your criteria. Wildcard characters can be placed anywhere in the box and could return different results depending on the location of the wildcard character. For example, *ed returns all items ending in ed, while ed* returns all items beginning with ed.
- To find all items, type a wildcard character (* or %) or leave the box empty.
- Leading and trailing spaces are removed.

Report Logic Examples

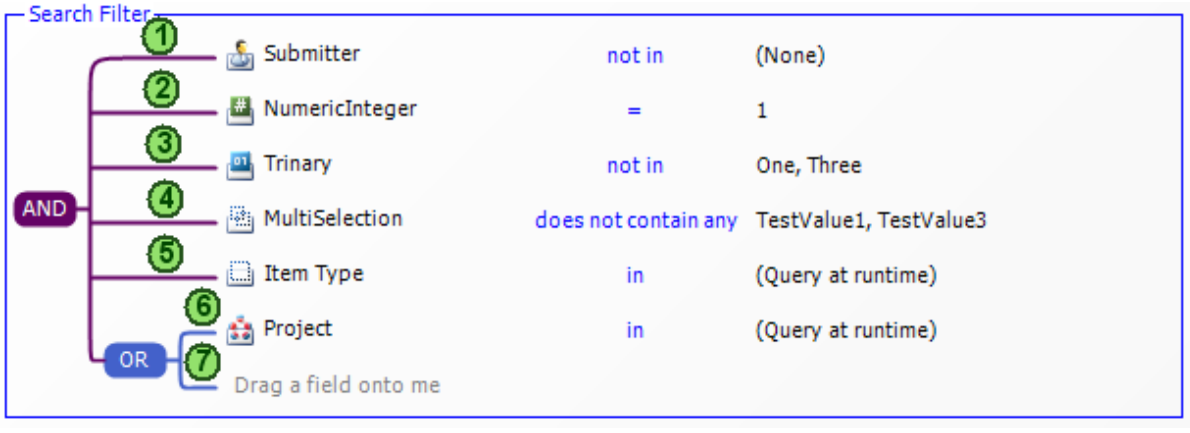
The following examples illustrate how reports are constructed.

Note: AND and OR operators always have at least two child nodes. The nodes are either fields or **Drag a field onto me** placeholders.

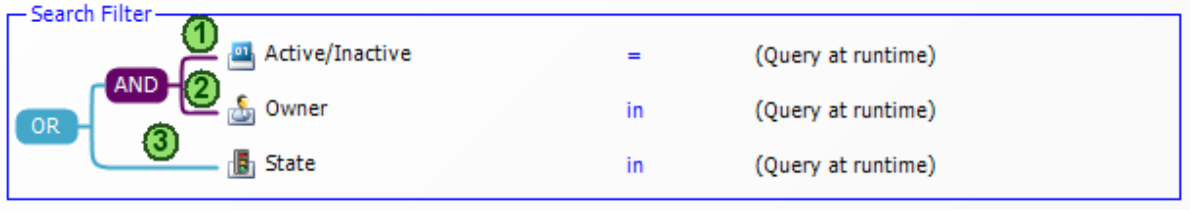
(1 or ((2 or 3) and (4 or 5)) or (6 and 7)) and 8



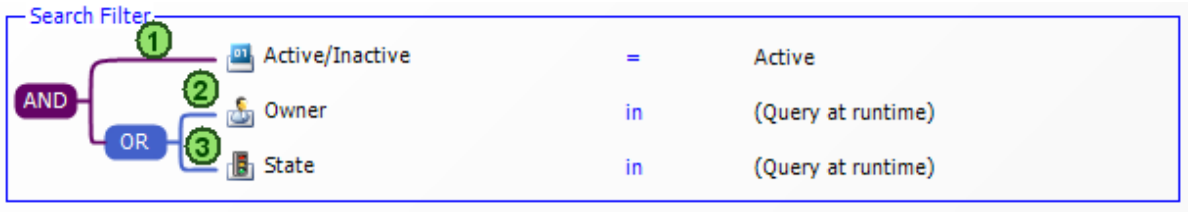
1 and 2 and 3 and 4 and 5 and (6 or 7)



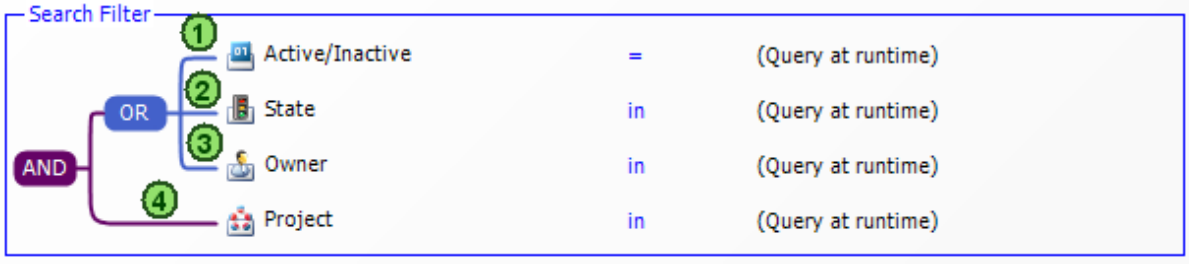
(1 and 2) or 3



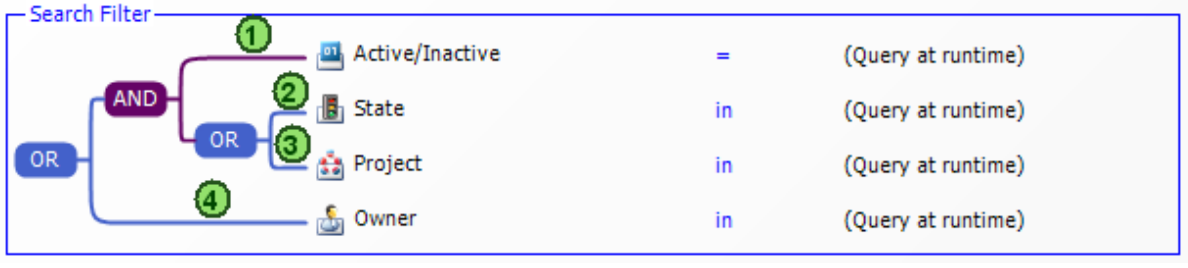
1 and (2 or 3)



(1 or 2 or 3) and 4



(1 and (2 or 3)) or 4



Drag-and-Drop Behavior

The following points describe how to use the drag-and-drop operation to add fields and operators to a search filter.

Drag-and-Drop Operation	Behavior
AND to AND	To add a new expression to an AND group, drag a field from the palette onto another field in the group. The new field is added above the field onto which you dragged.
OR to AND	To add an OR group to an AND group, drag the **OR** operator to one of the lines extending from the AND group. If there is a field on the top line, the OR group is added to the bottom line of the AND group. If no field is on the top line, the OR group is added to the top line of the AND group.
OR to OR	To add a new expression to an OR group, drag a field from the palette onto another field in the group. The new field is added above the field onto which you dragged.

Drag-and-Drop Operation	Behavior
AND to OR	To add an AND group to an OR group, drag the **AND** operator to one of the lines extending from the OR group. If there is a field on the top line, the AND group is added to the bottom line of the OR group. If no field is on the top line, the AND group is added to the top line of the OR group.
Existing Field	To move an existing field within an AND or OR group, drag it to another field. The existing field is placed above the field onto which you dragged it.

Chapter 12: Forms, Images, and Styles

This section contains the following information:

About Forms

Forms are pages in the SBM User Workspace in which users submit, transition, update, and view items. In SBM Composer, you specify which forms appear for items in an *application* [page 679]. You can choose the automatically generated forms, called *quick forms*, or create custom forms for primary and auxiliary items.

For primary items, each *application workflow* [page 680] has a default state form and a default *transition form* [page 700].

- The state form provides a read-only view of the field values of a *primary item* [page 691] as the primary item resides in a particular state. For example, Elizabeth, a development manager, receives a notification that a bug was submitted. When she

opens the item, she sees the form for the **New** state. It is a read-only form that displays the values of fields such as *Title*, *Description*, and *Summary*. These fields were filled in by the person who submitted the issue.

- The transition form lets users update primary items when they execute a transition, such as **Submit**, **Copy**, or **Update**. For example, after Elizabeth clicks the **Assign** button on the state form, the form for the **Assign** transition opens. It contains fields such as *Developer* and *Target Turnover*, in which she selects the developer that will fix the bug and the build in which it should be fixed.

Note: All states and transitions in the application workflow use the default forms unless you override them for an individual state or transition.

Note: You can use SBM System Administrator to further specify forms for individual projects.

For auxiliary items, each *auxiliary table* [page 680] is associated with a view form and an edit form.

- The view form provides a read-only view of the field values for an *auxiliary item* [page 680].
- The edit form lets users submit and update auxiliary items.

Quick Forms and Custom Forms

This topic summarizes the key differences between the auto-generated quick forms and the custom forms that you can create.

Privilege Sections

On a *quick form* [page 693], fields are grouped in *privilege sections*—all fields associated with a specific privilege section, such as User, Advanced, and Manager, are organized in a single visual section on the form. Each section will be visible or accessible to users based on their privileges.

On a custom form, each field is still associated with a privilege section and visible or accessible to users based on their privileges; however, you can create *visual sections* on the form and organize fields within them in any way you like.

For example, you might create a field called *Project Priority*, which you assign to the User privilege section. In the quick form, the field would appear automatically in the User section for users with privileges to work with *User* fields. If you create a custom form, you might add a tab control with the label **Backlog Info** and place the field within that tab; the *Project Priority* field would appear on the **Backlog Info** tab, though still only for users with privileges to work with *User* fields.

Note: The *Rich Editable Grid* [page 694] and the *Mass Update* [page 689] feature are governed by privilege sections and not field placement on custom forms. For example, if you remove a field from an Update form, but users have privileges to update the field, they can do so from the Editable Grid.

Custom Layout and Controls

Quick forms have a static layout based on privilege sections. Custom forms can be laid out using SBM Composer's design templates or with your own custom layout. You can also

add your own images, hyperlinks, static text, buttons, and JavaScript files to custom forms.

Custom Transition Controls

You can add custom transition controls to custom forms for states and transitions. For more information, see About Custom Transition Controls [page 200].

Section 508 and DDA

Quick forms are Section 508- and DDA-compliant, in that you can create custom forms that meet Section 508 and DDA requirements; however, SBM Composer does not validate or enforce compliance.

Detail Controls Sections

Quick forms always include a number of Detail Controls sections, such as *State Change History* [page 697], Notes, and Attachments. These sections appear for an item based on the user's privileges and preference settings. With custom forms, you can choose to include or exclude the available sections.

Note: If you are using the SourceBridge integration, you must include the Version Control section on a custom form.

Choosing Quick Forms or Custom Forms

These simple scenarios could help you decide between using a quick form and creating a custom form.

Scenario	Quick form	Custom form
You need to get up and running quickly.	✔	
You want fields grouped and ordered by privileges.	✔	
You want forms that are automatically Section 508-compliant.	✔	
You want to use the quick forms as a starting point, and then make minor modifications, such as removing some of the Item Details sections.		✔ (based on the quick form)
You want to differentiate forms that are used in different applications.		✔
You want flexibility in overall layout styles.		✔
You want to add custom information to the form, such as static help text and links to pages on your intranet.		✔

Scenario	Quick form	Custom form
You want to use custom transition controls (buttons, hyperlinks, or images) in addition to or instead of the standard transition buttons at the top of the form.		✔

Note: To change the appearance of the SBM User Workspace beyond item forms, administrators with advanced knowledge of HTML and JavaScript can make global *template* [page 699] changes. For details, refer to the *SBM System Administrator Guide*.

Creating Custom Forms

You can create custom forms for use throughout an *application* [page 679]. For primary items, you can create a state or transition form for an *application workflow* [page 680] or an individual state or transition. For auxiliary items, you can create a view or edit form for an *auxiliary table* [page 680].

Custom forms are of two types:

- State
- Transition

You can create a custom form based on a blank form, a *quick form* [page 693], or an existing custom state or transition form in the open process app. The types do not have to match (that is, you can create a state form based on a *transition form* [page 700], and you can create a transition form based on a state form).

To create a custom form:

1. Select the *design element* [page 683] for which you want to create a custom form. If you want to assign the form later, select an application.

Design element	Step
Application	In App Explorer, right-click the **Forms** heading (or an existing form under that heading), select **Add New**, and then select **State Form** or **Transition Form**. Then continue to step 3.
Application	In the **New** area on the **Home** tab of the Ribbon, click **Element** and then click **State Form** or **Transition Form**. Then continue to step 3.
Application workflow	In App Explorer, select the application workflow. Then select the **Forms** tab of the workflow Property Editor.

Design element	Step
Individual state or transition	In App Explorer, select the application workflow. In the application editor, select the state or transition, and then select the **Form** tab of the state or transition Property Editor.  **Note:** You cannot create a custom form for the **Submit**, **Email**, **Any**, and **Deleted** states.
Auxiliary table	In App Explorer, right-click the **Forms** heading (or an existing form under that heading), select **Add New**, and then select **State Form** or **Transition Form**. Then continue to step 3.
Auxiliary table	In App Explorer, select the auxiliary table, and then select the **Forms** tab of the table Property Editor.

2. Click the **New** button.
3. The **Form Configuration** dialog box opens. Complete the dialog box as described in Form Configuration Dialog Box [page 633].
4. To change the overall layout for the form, select a layout from the **Form Layout** area on the **Design** tab of the Ribbon.
5. To create visual sections on the form, drag container controls (**Expander**, **GroupBox**, **Panel**, and **Tab**) onto the form, configuring the rows and columns of the containers as needed.

    **Note:** You can drag container controls into existing cells on the form, creating visual subsections.

6. Drag field controls, detail controls, and other controls onto the form as needed. Each field control and detail control can be used only once on the form.

    **Note:** To remove a control from the form, select it and press the Delete key, or right-click the control, and click **Delete**. Deleted field controls and detail controls are restored to the **Form Palette**, making them available for use.

7. Use the tabs in the form Property Editor to view and modify the various aspects of the selected form:
   - General [page 201]
   - JavaScripts [page 206]
   - Rows [page 206]
   - Columns [page 207]
   - Appearance Tab of the Form and Control Property Editor [page 214]
8. To view a mockup of the form, click **Preview** on the **Design** tab of the Ribbon. The preview opens in the Form Preview Dialog Box [page 634]. You can use the controls

at the top (for example, the **Context** list) to view the form as it will be seen in a different workflow, state, or transition and by a user in a different *role* [page 695]. You can click the buttons at the top of the form to move through the states and transitions.

The new form appears under the **Forms** heading in App Explorer. You can right-click the form to rename, duplicate, or delete it. The form is also available for selection in the applicable design elements.

Tip: To view the states and transitions that use a particular custom form, click **Preview**. The states and transitions associated with the custom form are in the **Context** list.

About Custom Transition Controls

You can add custom transition controls to custom forms for primary and auxiliary tables. Custom transition controls can be used in addition to the standard transition buttons in the button bar at the top of the form. For example, you might want to put a custom transition button in the middle of a form, so the user can click it without having to scroll to the top of the form, but want to keep the standard button for that transition in the button bar.

A custom transition control can also be used instead of a button in the button bar. For example, you might have a state with two outgoing transitions, **Get Info from Submitter** and **Begin Work**. You could keep the button for the **Begin Work** transition in the button bar, but remove the **Get Info from Submitter** button from the button bar and instead add a custom button for it on the other side of the form, separate from the button bar.

Note: Because a sub-workflow inherits the custom forms used by its parent workflow, custom transition controls are also inherited. If you want to add or remove custom transition controls to a form in a sub-workflow, duplicate the parent form in the sub-workflow, change the controls, and associate the new form with the sub-workflow.

For detailed information, see Behavior Tab of the Control Property Editor [page 208] and Custom Transition Control Tutorials [page 260].

Key Benefits

- A button, hyperlink, or image control can be associated with a transition, and that transition will be executed when the user clicks the control in the SBM User Workspace.
- The controls can be used in addition to or instead of standard transition buttons.
- You can place the controls anywhere on a custom state or transition form, and can give them arbitrary labels.
- One form can be used for many states, because you can add a single control whose label changes dynamically based on its associated transition in the current state.

Form Editor

Use the form editor to design a custom form. You can base the form on a *quick form* [page 693] or on an existing form, or you can start with an empty form. For details, refer to Creating Custom Forms [page 198].

In the form editor, drag controls from the **Form Palette** and organize them on the form. Additional options are available when you right-click the form and its controls. Such options include deleting controls, adding and deleting columns and rows, and opening associated fields.

Tip: For shortcut keys you can use in the form editor, see Form Shortcut Keys [page 69].

Palette

The form palette includes the following types of controls:

- **Field Controls** represent all fields in the associated table. Each field control can be used only one time on the form.
- **Detail Controls** represent all Item Details sections that can appear for an item, such as Attachments, Change History, and Notes. Each detail control can be used only once on the form.
- **Container Controls** represent the visual sections that you can use to group controls together on the form.
- **Other Controls** represent the custom elements that you can include on the form, such as images, static text, and hyperlinks. These controls can be used as needed and are configured in the Property Editor.
- **Widgets** represent specialized *widget* [page 701] types, including widgets that you can use to embed almost any valid HTML or JavaScript, the content found at any valid URL, a YouTube video, or an Amazon.com search, and so on.

For detailed information about the **Form Palette**, see Form Palette [page 215].

Form Property Editor

Use these tabs in the form Property Editor to view and modify the various aspects of the selected form:

- General [page 201]
- Javascripts [page 206]
- Rows [page 206]
- Columns [page 207]
- Appearance Tab of the Form and Control Property Editor [page 214]

Note: If you select a field on the form, the field Property Editor replaces the form Property Editor.

General Tab of the Form and Control Property Editor

The information and options that appear on the **General** tab depend on what you selected in the form or control editor.

Form

Control	Description
Name	The name of the form.
Description	An area for an optional description of the form.
Options *(Transition form)*	Select the **Remove transition buttons matching custom transition controls** check box if you added custom transition controls and do not want the standard transition buttons for those transitions to appear in the button bar at the top of the form. (See Custom Transition Controls [page 197] for details.) Select the **Validate required fields before form submit** check box to specify that all required fields must contain a valid value before a user submits the form in the SBM User Workspace. The validation takes place on fields that the workflow requires (those fields marked as **Required** in the field Property Editor) and on fields that the SBM JavaScript Library conditionally requires. (JavaScript that performs this validation is present on every custom transition and state form.) If this check box is not selected, validation takes place on only those fields that the JavaScript Library conditionally requires.  **Note:** Workflow fields can be overridden as required or not required at the workflow level in SBM Composer and in SBM System Administrator.
Options *(State form)*	Select the **Show button bar** check box if you added custom transition controls but want to keep the standard transition button bar at the top of the form. Clear this check box if you want to limit the available transitions to those associated with custom transition controls. Select the **Remove transition buttons matching custom transition controls** check box if you added custom transition controls and do not want the standard transition buttons for those transitions to appear in the button bar at the top of the form. (See Custom Transition Controls [page 197] for details.)

Control	Description
Size guide	Select the **Show** check box to display a shaded boundary around the portion of the form that will be visible to users in a single screen. When the option is selected, use the **Width** and **Height** controls to set the desired size. Instead of typing a specific number in the **Width** and **Height** controls, you can type `a` to automatically set the width of the column or the height of the row to fit the contents of the column or row. The **Size guide** option does not affect the behavior of the form. It is simply an aid in designing forms that minimize the need for scrolling.  **Tip:** If you know the likely size of your users' screens, set the **Size guide** dimensions a little smaller. If you design your form to fit within the **Size guide** parameters, it is more likely that your users will be able to see and use all the controls on the form without having to scroll.

Field Controls and Their Labels

Control	Description
This control is linked to field '*field-name*' in table '*table-name*'.	Click this link (in the upper right corner of the Property Editor, next to the drop-down list of form components) to view the field's definition in the '*table-name*' table. Click  on the Quick Access toolbar to return to the field's properties in the form editor.
Name	SBM Composer automatically assigns unique names to field controls.
Description	An area for optional comment about the control.
Text	For a field label, the display name of the field. For a field control, the value depends on the context, as visible in the form preview.
Options	*(some field types)* Show as dual listboxes in design/preview mode.

Detail Controls and Container Controls

Control	Description
Name	The control name must be unique within the form.  **Note:** SBM Composer automatically assigns unique names to detail controls.
Description	An area for optional comment about the control.
Display text	The label to display on the expander, group box, or tab.  **Note:** This field is ignored for panel controls, which show no label.
Options	**Expander**: Display the contents in a collapsible/expandable area. Set the **Show collapsed on startup** option to specify whether the expander should be collapsed initially when the form opens. **Group box**: Display the contents in a bounded box that includes the specified title text. **Panel**: Display the contents in an unbounded box that does not include the specified title text. **Hide if no attachments have been added**: *(Attachments Detail control)* Clear this check box if you want the **Attachments** section to appear on a custom form, even if there are no attachments. **Hide if no notes have been added**: *(Notes Detail control)* Clear this check box if you want the **Notes** section to appear on a custom form, even if there are no notes. **Show lines between rows**: *(Container controls)* Display thin lines between rows within this container. This can make a wide form easier for your users to read.  **Note:** You cannot change a tab container to a different container type and cannot change another container type to a tab container.  **Note:** This option does not apply to **Detail Controls**.

Other Controls and Their Labels

Control	Description
Control name	The control (or label) name must be unique within the form.

Control	Description
Description	An area for optional comment about the control.
Value(s)	*(Combo Box, List Box)* The values to be available for your users to choose from in the SBM User Workspace. Type one value on each line.
Default text	*(Edit Box)* The text you want to display to your users when they first display the form. You could use this field to provide a default value or a prompt for the data you want them to enter. If you select the **Multi-line edit box** option, the default text (and anything your users type in its place) will wrap. If you select the **Password** option, the default text (and anything your users type in its place) will appear as a string of black dots.
Options	*(Edit Box)* Select **Edit box** to create a single-line text entry field, and then select **Password** if you want characters in the field to be replaced by black dots. Select **Multi-line edit box** to create a multiple-line text area, and use the **Width** and **Height** controls (in the **Size** area on the **Design** tab of the Ribbon) to control the dimensions of the text area. *(Combo Box, List Box)* Select **Combo box** to create a drop-down list, in which only the first item listed for **Value(s)** above is initially displayed in the SBM User Workspace. The other values are displayed when users click the arrow at the right side of the combo box. Select **List box** to create a scrolling list. Select **Allow multiple selection** to use the GetFieldValues [page 662], SetFieldValues [page 663], GetMultiListValues [page 663], and SetMultiListValues [page 665] JavaScript API functions with the list. Use the **Height (in rows)** control to specify how many of the items in the **Value(s)** control above will be displayed at a time.
Display text	*(Button)* The text to be displayed on the button. *(HyperLink)* The "clickable" text for the link. *(Image)* Text that will be displayed as a "tool tip" when the user holds the mouse pointer over the image in the SBM User Workspace or in place of the image if it cannot be loaded for some reason. This text may be recognized by text-to-speech software (also known as *screen readers*). *(Text)* The text to be displayed. The text you type here can include HTML tags (`<b>` and `</b>`, for example). They will be included in the HTML code for the page that is rendered in the SBM User Workspace. *(Edit/Combo/List Box label)* The text to be displayed next to the associated **Edit box**, **Combo box**, or **List box** control. SBM Composer creates a default label when you place one of these controls on your form, and then change that label text to reflect the purpose of the control.

Control	Description
Image	*(Image)* The image to display. Select an image already added to this process app, or select **(New image...)** to add a new image to the process app and use it for this control.

Widgets

See Form Widgets [page 223].

JavaScripts Tab of the Form Property Editor

You use the **JavaScripts** tab of the form Property Editor to include your own JavaScripts in a custom form. The SBM Server executes the scripts when the form is opened in the SBM User Workspace.

Note: By default, the scripts are not executed during form preview. For more information, see Form Preview Dialog Box [page 634].

Element	Description
Add	If you select **New**, opens the JavaScript editor, where you can create or import a JavaScript file. The JavaScript file is automatically included in the selected form. If you select a JavaScript file from the drop-down list, that file is included in the selected form. This list includes files under the **JavaScripts** heading in App Explorer that are not already included in the form.
Import	Imports a JavaScript file into the process app and includes it in the selected form.
Remove	Removes the selected JavaScript file from the form (but not from the *application* [page 679], because the JavaScript file may be used by other forms).
Move up/down	Moves the selected JavaScript file higher or lower in the list. The order of the files in the list determines the order in which they are included by the SBM Server. This is important if you are reusing JavaScript files in several different forms. Files in the list may require variables to be defined or scripts to be executed in previous includes for them to function properly.

Rows Tab of the Form and Control Property Editor

The **Rows** tab of the form Property Editor lets you manipulate the rows in the selected form or container control. Rows and columns are used to position and size the controls that you put on the form.

Element	Description
Type	Select **Fixed**, **Percentage**, or **AutoSize**.

Element	Description
Height	For fixed rows, type the number of pixels. For percentage rows, type the percentage of total height you want this row to occupy. Other rows are adjusted as needed for a total of 100%. For autosize rows, the height is irrelevant and cannot be set.
Add row above/ below	Adds a row above or below the selected cell.
Delete row	Deletes the entire row containing the selected cell.

Columns Tab of the Form and Control Property Editor

The **Columns** tab of the form Property Editor lets you manipulate the columns in the selected form or container control. Use rows and columns to position and size the controls that you put on the form.

Element	Description
Type	Select **Fixed**, **Percentage**, or **AutoSize**.
Width	For fixed columns, type the number of pixels. For percentage columns, type the percentage of total width you want this column to occupy. Other columns are adjusted as needed for a total of 100%. For autosize columns, the width is irrelevant and cannot be set. For mixed (autosize and percentage) columns, the label portion is wide enough to accommodate the longest label, and the field portion is allocated by percentage.
Add column left/right	Adds a column to the left or right of the selected cell.
Delete column	Deletes the entire column containing the selected cell.

Refresh Tab of the Control Property Editor

The **Refresh** tab is available on the **Edit Box**, **HyperLink**, **Image**, and **Text** controls. Use this tab to specify the form data that should be displayed in the control, and define when the data should be refreshed.

Note: Instead of typing or pasting text into the **Contents**, **Image Url**, and **Display text** fields, you can use the *string builder tool* [page 698] to insert references to table fields and form controls, and to insert HTML tags. For more information, see Using the String Builder Tool [page 249].

Element	Description
Contents	Enter the content that should be displayed in the **Edit Box** control.
Image Url	Enter the URL of the image that should be displayed in the **Image** control.
Display text	Enter the text that should be displayed in the **HyperLink** or **Text** control.
On page load	If this check box is selected, the content is retrieved when the form is initially displayed in the SBM User Workspace or in the form preview.  **Note:** This check box is automatically selected if the **Contents**, **Image Url**, or **Display text** field does not refer to any data fields, and if the form does not include any buttons, images, or hyperlinks.
On data change	If this check box is selected, the content is updated when the field that maps to the **Contents**, , or **Display text** field is changed in the SBM User Workspace or in the form preview.  **Note:** This check box is disabled if the fields listed above do not refer to any data field.
On click	In the drop-down list box, select one of the buttons, images, or hyperlinks that are on this form. The content is updated when the control is clicked in the SBM User Workspace or in the form preview.  **Note:** This check box is disabled and the drop-down list is hidden if the form does not include any button, image, or hyperlink.

Behavior Tab of the Control Property Editor

The **Behavior** tab is available on the **Button**, **HyperLink**, and **Image** control Property Editors.

Use this tab to specify the content and appearance of a Web page or popup window that opens when a user clicks the control on a custom form in the SBM User Workspace.

Tip: The Web page or popup window can provide guidance about using the custom form (for example, specific information about a field). For more information about providing guidance to users, see Chapter 23: Providing Guidance to Users [page 405].

You can also use this tab to use custom transition controls in addition to or instead of the standard transition buttons at the top of a custom form. These custom controls can have arbitrary labels and can be placed anywhere on the form. For more information, see About Custom Transition Controls [page 200].

On a default custom state form, you can use a placeholder for the control label. The placeholder is replaced by the name of the outgoing transition from the current state. If you associate multiple states with this form, you can add a single control that can be used to transition an item out of each state, because the label changes automatically, as described above.

You can associate multiple controls with a single transition. This is useful when you want to repeat controls in different parts of a long form, so users do not have to scroll to the standard button bar at the top of the form.

Note: See Custom Transition Control Tutorials [page 260] for a specific use case and instructions.

State Form Options (Primary Table)

Element	Description
Open URL	Specifies that a Web page opens when the user clicks the control.
URL	The URL of the Web page that opens when the user clicks the control. Type { if you want to insert references to table fields or other form controls.

Element	Description
Target	Whether the Web page should open in the same window as the SBM User Workspace, or in a separate window. You can also type the name of a window or an iframe (inline frame) in this field. For example, suppose you have a **Help** button for two fields on the form. You type `Help` in the **Target** box for each button instead of selecting **New Window** or **Same Window**, and enter a unique URL for each button. When a user clicks the **Help** button for the first field, information about the first field is displayed in a new window. When the user clicks the **Help** button for the second field, information about the second field replaces the information in the window. **Note:** When the control is on a *transition form* [page 700], **Same Window** is not an option.
Open custom popup	Specifies that a popup window opens when the user clicks the control.
Body HTML	The content of the popup window. You can type HTML code if you want the text formatted, or type text if you do not want the text formatted. Type `{` if you want to insert references to table fields or other form controls. There is no limit to the number of characters you can type. **Important:** JavaScript within the HTML code is not supported. **Note:** To make text wrap as you enter content, right-click in this box and then select **Wrap Lines**. A check box indicates that this option is seleted. To make the text be on one line with a scroll bar, select **Wrap Lines** again to clear the check box.
Options	Default values for features that specify the appearance of the popup window. You can change the values and add additional features and values. For a description of the features, see Creating Links to Web Pages or Popup Windows [page 405].
Reset	Restores the default values for the features that specify the appearance of the popup window.
Perform a transition	Specifies that an outgoing transition from a state that uses this custom form will be executed when the user clicks a custom transition control.

Element	Description
Mapping	Shows which transition will be executed for a specific state when a user clicks the custom transition control. You can see the application workflow where the transition is defined in the **Defining application workflow** column.  **Note:** Mapping also applies to all sub-workflows of the defining application workflow.
Show transition name	If you select this option, you can configure the control to use the transition name as its label. This is useful when using a single control to trigger different transitions depending on the current state. This option is also useful for a single transition, because if the transition name changes in the application workflow, the label changes automatically.  **Tip:** It is recommended that you type `auto` in the **Height** and **Width** boxes in the **Size** section on the **Design** tab of the Ribbon. If you specify fixed values instead, the label for a long transition name could be truncated. If you do not select this check box, the display text entered on the **General** tab of the control Property Editor is shown.
Open	Opens the application workflow for the selected row in the **Transitions** list. The transition in the selected row is selected in the workflow editor.
Add	Opens the Add Transition Dialog Box [page 620], which lets you select the transition that should be associated with the selected state.  **Note:** You can add a transition only if the custom control is on a form that is associated with a state or that is the default state form for the workflow.  **Tip:** If you selected **Show transition name** and want to have a single control for multiple transitions, add each transition (one at a time) to this tab.
Remove	Removes the selected row from the **Transitions** list.

Transition Form Options (Primary and Auxiliary Tables)

Element	Description
Open URL	Specifies that a Web page opens when the user clicks the control.
URL	The URL of the Web page that opens when the user clicks the control. Type `{` if you want to insert references to table fields or other form controls.

Element	Description
Target	Whether the Web page should open in the same window as the SBM User Workspace, or in a separate window. You can also type the name of a window or an iframe (inline frame) in this field. For example, suppose you have a **Help** button for two fields on the form. You type `Help` in the **Target** box for each button instead of selecting **New Window** or **Same Window**, and enter a unique URL for each button. When a user clicks the **Help** button for the first field, information about the first field is displayed in a new window. When the user clicks the **Help** button for the second field, information about the second field replaces the information in the window. **Note:** When the control is on a *transition form* [page 700], **Same Window** is not an option.
Open custom popup	Specifies that a popup window opens when the user clicks the control.
Body HTML	The content of the popup window. You can type HTML code if you want the text formatted, or type text if you do not want the text formatted. Type `{` if you want to insert references to table fields or other form controls. There is no limit to the number of characters you can type. **Important:** JavaScript within the HTML code is not supported. **Note:** To make text wrap as you enter content, right-click in this box and then select **Wrap Lines**. A check box indicates that this option is seleted. To make the text be on one line with a scroll bar, select **Wrap Lines** again to clear the check box.
Options	Default values for features that specify the appearance of the popup window. You can change the values and add additional features and values. For a description of the features, see Creating Links to Web Pages or Popup Windows [page 405].
Reset	Restores the default values for the features that specify the appearance of the popup window.
Submit the form	Lets you create a custom transition control that performs the same function as the standard **OK** button. The control can be placed anywhere on the form, and can have a unique label. **Note:** On the **General** tab of the form Property Editor, specify whether the standard **OK** button should remain in the button bar on the top of the form.

Element	Description
Cancel the form	Lets you create a custom transition control that performs the same function as the standard **Cancel** button. The control can be placed anywhere on the form, and can have a unique label.  **Note:** On the **General** tab of the form Property Editor, specify whether the standard **Cancel** button should remain in the button bar on the form.

State Form Options (Auxiliary Tables)

Element	Description
Open URL	Specifies that a Web page opens when the user clicks the control.
URL	The URL of the Web page that opens when the user clicks the control. Type `{` if you want to insert references to table fields or other form controls.
Target	Whether the Web page should open in the same window as the SBM User Workspace, or in a separate window. You can also type the name of a window or an iframe (inline frame) in this field. For example, suppose you have a **Help** button for two fields on the form. You type `Help` in the **Target** box for each button instead of selecting **New Window** or **Same Window**, and enter a unique URL for each button. When a user clicks the **Help** button for the first field, information about the first field is displayed in a new window. When the user clicks the **Help** button for the second field, information about the second field replaces the information in the window.  **Note:** When the control is on a *transition form* [page 700], **Same Window** is not an option.
Open custom popup	Specifies that a popup window opens when the user clicks the control.

Element	Description
Body HTML	The content of the popup window. You can type HTML code if you want the text formatted, or type text if you do not want the text formatted. Type { if you want to insert references to table fields or other form controls. There is no limit to the number of characters you can type.  **Important:** JavaScript within the HTML code is not supported.  **Note:** To make text wrap as you enter content, right-click in this box and then select **Wrap Lines**. A check box indicates that this option is seleted. To make the text be on one line with a scroll bar, select **Wrap Lines** again to clear the check box.
Options	Default values for features that specify the appearance of the popup window. You can change the values and add additional features and values. For a description of the features, see Creating Links to Web Pages or Popup Windows [page 405].
Reset	Restores the default values for the features that specify the appearance of the popup window.
Update the item	Lets you create a custom transition control that performs the same function as the standard **Update** button. The control can be placed anywhere on the form, and can have a unique label.  **Note:** On the **General** tab of the form Property Editor, specify whether the standard **Update** button should remain in the button bar at the top of the form.
Delete the item	Lets you create a custom transition control that performs the same function as the standard **Delete** button. The control can be placed anywhere on the form, and can have a unique label. On the **General** tab of the form Property Editor, specify whether the standard **Delete** button should remain in the button bar at the top of the form.

Appearance Tab of the Form and Control Property Editor

The **Appearance** tab is available on the controls in the **Detail Controls** and **Container Controls** sections of the **Form Palette**. This tab lets you fine-tune the appearance of cells and containers on the form.

Element	Description
Table settings *(Forms and Container Controls)*	**Cell spacing**. Specify the number of pixels that should be between cells. **Cell margins**. Specify the number of pixels that should surround the content of cells. **Restore defaults**. Restores the default value of 0 for the cell spacing, and 4 for the cell margins.
Container padding *(Container Controls and Detail Controls)*	**Top.** Specify the number of pixels that should be at the top of the container. **Left.** Specify the number of pixels that should be at the left of the container. **Bottom.** Specify the number of pixels that should be at the bottom of the container. **Right.** Specify the number of pixels that should be at the right of the container. **Restore defaults**. Restores the default values of 0 for the top, 20 for the left, 0 for the bottom, and 20 for the right.

Form Palette

In the form editor, you drag controls from the **Form Palette** and organize them on the form. You then use the control Property Editor to configure the controls. This topic describes the controls in each section of the **Form Palette**.

Field Controls

The controls in the **Field Controls** section represent fields in your *primary table* [page 692]. You can use each field control one time in each form.

Note: For information about field control options, see General Tab of the Form and Control Property Editor [page 201].

Detail Controls

The controls in the **Detail Controls** section represent sections in the SBM User Workspace that contain detailed information about items. You can use each detail control one time in each form. The following table describes these controls.

Note: For information about detail control options, see General Tab of the Form and Control Property Editor [page 201].

Note: Some sections are displayed on the form in the SBM User Workspace only if they contain something. For example, the **Item Notifications** section is not displayed until a notification is added to the item. This applies to the **Attachments**, **Notes**, **Item Notifications**, **Subtasks**, and **Version Control** sections. The **Attachments** and **Notes** sections can be seen, however, if you clear an option on the **General** tab of the **Attachments** and **Notes** Property Editors. For more information about how sections are populated in the SBM User Workspace, see the online help for the SBM User Workspace or the *Serena® Business Manager User's Guide*.

Control	Description
Attachments	Section that contains file attachments, item links, and URLs.
Change History	Section that contains information pertaining to changes to items. For example, a *change history* [page 681] entry is added to an item when it is submitted and each time it is transitioned or updated.
Notes	Section that contains notes and e-mail messages attached to an item.
Item Notifications	Section that contains item *notifications* [page 689] that a user subscribed to.
State Change History	Section that contains a graphical or tabular representation of the states and transitions a *primary item* [page 691] moved through as it is tracked through a workflow.  **Note:** This control is used on state forms only.
Subtasks	Section that contains links to subtasks or principal primary items.
Version Control	Section that contains source control information associated with a primary item. This section is used if your system has an integration to a version control tool.

Control	Description
New Note	Control that lets users type in a comment when they transition an item. On the **Options** tab of the transition Property Editor, you can specify whether this field should be displayed, and whether it should be a mandatory field.  **Note:** This control is used on transition forms only.
Integrations	Section for external URL integrations.

Container Controls

The controls in the **Container Controls** section represent types of containers that hold other controls or information. For example, a **History** tab could contain the **Change History** detail control. Container controls serve the same purpose, but differ in appearance. A container control can contain another other container controls. For example, a **System** tab could contain group boxes that contain different types of system information.

 **Note:** For information about container control options, see General Tab of the Form and Control Property Editor [page 201] and Container Control Options [page 219].

Control	Description
Expander	An expander control can be expanded to show its contents, or collapsed to hide the contents. Users can use expander controls to view or hide information based on their interests.
GroupBox	A group box provides groups related information in an outlined area on the form.
Panel	A panel is a rectangular area that holds related information.
Tab	A user can click a tab in a series of tab controls to view different types of information. Tab controls save space on the form.

Other Controls

The **Other Controls** section contains controls that users interact with in the SBM User Workspace. It also lets you add images to a form.

Note: For information about options for these controls, see General Tab of the Form and Control Property Editor [page 201].

Important: Button, **HyperLink**, and **Image** controls can be used to open Web pages and popup windows, and to act as custom transition controls that can transition items to another state in an application workflow. The custom transition controls can be used in addition to or instead of standard transition buttons at the top of a form. For more information, see Behavior Tab of the Control Property Editor [page 208].

Control	Description
Button	A button that opens the Web page you specify, opens a popup window, or executes a transition. **Note:** Standard buttons that users click at the top of a transition form to transition items are automatically added to a button bar at the top of state and transition forms. These buttons can be removed if custom transition controls are added to the form. (See **Important** note, above.)
Combo Box	Lets the user type an item or select an item from a list.
Edit Box	Lets the user type or edit items in a field. For example, this could be a **Password** field.
HyperLink	A link that opens the Web page you specify, opens a popup window, or executes a transition.
Image	Lets you add an image to the form. If configured, opens the Web page you specify, opens a popup window, or executes a transition.
List Box	Lets the user select an item from a list.
Text	Lets the user type information into a field. For example, this could be a **Description** field in which the user types the description of a software problem.

Widgets

The controls in the **Widgets** section represent widgets (small applications that provide a special function). For detailed information about the available widgets, see Form Widgets [page 223].

Container Control Options

In the form editor, you can right-click a container control and select from options relevant to that control.

Note: You can also access container control options in the control Property Editor. For information about these options, see General Tab of the Form and Control Property Editor [page 201].

Expander Control Options

The following table describes the options available from the menu when you right-click an expander control in a form.

Option	Description
Collapse	Collapses the control on the form. **Note:** The Collapse option collapses the control on the form, but not in Preview mode or in the SBM User Workspace. To collapse the control in these places, select **Show Collapsed on Startup** on the **General** tab of the control Property Editor.
Select Element	For information about the **Select Element** option, see Selecting Parent Controls or Cells on a Form [page 255].
Row Sizing	Lets you select the sizing behavior for the rows in the control. For more information, see Resizing Columns and Rows [page 253].
Column Sizing	Lets you select the sizing behavior for the columns in the control. For more information, see Resizing Columns and Rows [page 253].
Add Row Above	If you select the control, adds a row to the form above the control. If you select a cell within the control, adds a row to the control, above the selected cell.
Add Row Below	If you select the control, adds a row to the form below the control. If you select a cell within the control, adds a row to the control, below the selected cell.
Add Column Left	If you select the control, adds a column to the form to the left of the control. If you select a cell within the control, adds a column to the control, to the left of the selected cell.
Add Column Right	If you select the control, adds a column to the form to the right of the control. If you select a cell within the control, adds a column to the control, to the right of the selected cell.
Delete Row(s)	If you select the control, deletes the row containing the control. If you select a cell within the control, deletes the row below the selected cell. You can select multiple cells to delete multiple rows.

Option	Description
Delete Column(s)	If you select the control, deletes the column containing the control. If you select a cell within the control, deletes the column to the right of the selected cell. You can select multiple cells to delete multiple columns.
Container Options	Changes the container control to the selected type of container control.
Delete	Deletes the selected container control.
Edit Text	Lets you change the label on the container control.
Show Properties	Shows the Property Editor for the container control, if not already shown.

GroupBox Control Options

The following table describes the options available from the menu when you right-click a group box control in a form.

Option	Description
Select Element	For information about the **Select Element** option, see Selecting Parent Controls or Cells on a Form [page 255].
Row Sizing	Lets you select the sizing behavior for the rows in the control. For more information, see Resizing Columns and Rows [page 253].
Column Sizing	Lets you select the sizing behavior for the columns in the control. For more information, see Resizing Columns and Rows [page 253].
Add Row Above	If you select the control, adds a row to the form above the control. If you select a cell within the control, adds a row to the control, above the selected cell.
Add Row Below	If you select the control, adds a row to the form below the control. If you select a cell within the control, adds a row to the control, below the selected cell.
Add Column Left	If you select the control, adds a column to the form to the left of the control. If you select a cell within the control, adds a column to the control, to the left of the selected cell.
Add Column Right	If you select the control, adds a column to the form to the right of the control. If you select a cell within the control, adds a column to the control, to the right of the selected cell.

Option	Description
Delete Row(s)	If you select the control, deletes the row containing the control. If you select a cell within the control, deletes the row below the selected cell. You can select multiple cells to delete multiple rows.
Delete Column(s)	If you select the control, deletes the column containing the control. If you select a cell within the control, deletes the column to the right of the selected cell. You can select multiple cells to delete multiple columns.
Container Options	Changes the container control to the selected type of container control.
Delete	Deletes the selected container control.
Edit Text	Lets you change the label on the container control.
Show Properties	Shows the Property Editor for the container control, if not already shown.

Panel Control Options

The following table describes the options available from the menu when you right-click a panel control in a form.

Option	Description
Select Element	For information about the **Select Element** option, see Selecting Parent Controls or Cells on a Form [page 255].
Row Sizing	Lets you select the sizing behavior for the rows in the control. For more information, see Resizing Columns and Rows [page 253].
Column Sizing	Lets you select the sizing behavior for the columns in the control. For more information, see Resizing Columns and Rows [page 253].
Add Row Above	If you select the control, adds a row to the form above the control. If you select a cell within the control, adds a row to the control, above the selected cell.
Add Row Below	If you select the control, adds a row to the form below the control. If you select a cell within the control, adds a row to the control, below the selected cell.
Add Column Left	If you select the control, adds a column to the form to the left of the control. If you select a cell within the control, adds a column to the control, to the left of the selected cell.

Option	Description
Add Column Right	If you select the control, adds a column to the form to the right of the control. If you select a cell within the control, adds a column to the control, to the right of the selected cell.
Delete Row(s)	If you select the control, deletes the row containing the control. If you select a cell within the control, deletes the row below the selected cell. You can select multiple cells to delete multiple rows.
Delete Column(s)	If you select the control, deletes the column containing the control. If you select a cell within the control, deletes the column to the right of the selected cell. You can select multiple cells to delete multiple columns.
Container Options	Changes the container control to the selected type of container control.
Delete	Deletes the selected container control.
Show Properties	Shows the Property Editor for the container control, if not already shown.

Tab Control Options

The following table describes the options available from the menu when you right-click a tab control in a form.

Option	Description
Add New Tab	Adds a new tab to the control, to the right of the rightmost tab.
Select Element	For information about the **Select Element** option, see Selecting Parent Controls or Cells on a Form [page 255].
Row Sizing	Lets you select the sizing behavior for the rows in the control. For more information, see Resizing Columns and Rows [page 253].
Column Sizing	Lets you select the sizing behavior for the columns in the control. For more information, see Resizing Columns and Rows [page 253].
Add Row Above	If you select the control, adds a row to the form above the control. If you select a cell within the control, adds a row to the control, above the selected cell.
Add Row Below	If you select the control, adds a row to the form below the control. If you select a cell within the control, adds a row to the control, below the selected cell.

Option	Description
Add Column Left	If you select the control, adds a column to the form to the left of the control. If you select a cell within the control, adds a column to the control, to the left of the selected cell.
Add Column Right	If you select the control, adds a column to the form to the right of the control. If you select a cell within the control, adds a column to the control, to the right of the selected cell.
Delete Row(s)	If you select the control, deletes the row containing the control. If you select a cell within the control, deletes the row below the selected cell. You can select multiple cells to delete multiple rows.
Delete Column(s)	If you select the control, deletes the column containing the control. If you select a cell within the control, deletes the column to the right of the selected cell. You can select multiple cells to delete multiple columns.
Delete	Deletes the selected container control.
Show Properties	Shows the Property Editor for the container control, if not already shown.

Form Widgets

You can put a variety of content on your custom forms using *widgets*. Widgets are small applications that are part of the user interface and provide a special function. Widgets enable you to create what are sometimes called "rich interface process apps." In the form editor, the palette contains several specialized *widget* [page 701] types, including widgets that you can use to embed almost any valid HTML or JavaScript, the content found at any valid URL, a YouTube video, or an Amazon.com search, to name a few. For more information about form editors, see Form Editor [page 200].

Drag widgets from the **Form Palette** onto the form, just as you do with the controls on the palette. Use the tabs in the widget Property Editor to view and modify the various aspects of the selected widget. For more information about the Property Editor, see Property Editors [page 65].

Use the controls on the **Design** tab of the Ribbon to set the alignment and size of the selected widget. The **Alignment** options (left, right, top, bottom, center, fill) work as expected and determine which handles are available for "stretching" the selected widget to the desired size. The **Size** options provide precise control.

Using References to Table Fields and Form Controls

When you select a form widget in the form editor, you can use the "string builder" tool on certain tabs in the widget Property Editor to insert references to table fields and form controls. This creates dynamic widgets that are driven by *application* [page 679] data and by the actions of the user. For information about using the string builder, see Using the String Builder Tool [page 249].

Using References to Widget Data

For grid-style widgets (that is, the Amazon Search widget and the REST Grid widget), you can insert references to widget data in form controls. You can also do this in the Embedded Report widget. For information about doing this, see Binding to Widget Data [page 248].

The following topics provide detailed information about how to use widgets on custom forms.

Amazon Search Widget

The Amazon Search widget lets users quickly search for items at amazon.com. This could be useful in a training application that lets employees search for an industry-related book they need.

This topic describes how to configure the Amazon Search widget.

General Tab

The following table describes the fields on the **General** tab of the widget Property Editor.

Field	Description
Control name	Type the name by which the widget is uniquely identified. No other widget or control on this form can have the same name.
Caption	Type the text to appear in the title bar above the widget. If you leave this field blank, and do not select the **Paging** check box next to the **Options** field, the widget will not have a title bar.

Field	Description
Access key	Type your amazon.com access key ID, a forward slash (/), and your access secret. For example: `<AccessKey>/<AccessSecret>`. If you do not already have an Amazon Web services account, click **Sign up for Amazon access key**. You should receive an e-mail message with instructions on how to obtain an access key ID and access secret.
Options	**Border:** Select this check box to draw a thin line around the widget. **Scroll bars:** Select this check box to include scroll bars in the widget. This provides more content than the allotted space. **Paging:** Select this check box to add "first page," "previous page," "next page," and "last page" controls to the title bar when more than ten items are returned from the search operation.  **Note:** If this check box is cleared, only the first ten items returned from the search operation are accessible to users in the SBM User Workspace. If this check box is selected, the widget will have a title bar, even if the **Caption** field is blank.
Description	Type an optional description of the widget.

Query Tab

The following table describes the fields on the **Query** tab of the widget Property Editor.

Field	Description
Category	Do one of the following: • Select one of the Amazon-defined categories to narrow the search. • Select **All** to search all of Amazon. • Select **<Custom>** and then type a different category in the box to the right of the list. • Use the *string builder tool* [page 698] to insert references to table fields and form controls.  **Note:** For information about this tool, see Using the String Builder Tool [page 249].
Keywords	*(Optional)* Type terms that represent the item for which you want to search. Type a space between each keyword. Alternately, you can use the string builder tool to insert references to table fields and form controls. If this field is blank, then you must specify a value for either the **Title** or **Author/Artist** field.

Field	Description
Title	*(Optional)* Type a specific title (such as the name of a book, CD, or song). Alternately, you can use the string builder tool to insert references to table fields and form controls. If this field is blank, then you must specify a value for either the **Keywords** or **Author/Artist** field.
Author/ Artist	*(Optional)* Type a specific name for the author or artist. Alternately, you can use the string builder tool to insert references to table fields and form controls. If this field is blank, then you must specify a value for either the **Keywords** or **Title** field.
Condition	Do one of the following: • Select one of the Amazon-defined conditions to narrow the search. • Select **All** to include all matching items, regardless of condition • Select **<Custom>** and then type a different condition in the box to the right of the list. • Use the string builder tool to insert references to table fields and form controls.
Price	*(Optional)* Type the minimum and maximum prices that you are willing to pay. Use US dollars and cents in *XX.XX* format.

Result Tab

The following table describes the fields on the **Result** tab of the widget Property Editor.

Field	Description
Render as	**Grid:** Select this option if you want the search results to be presented as an HTML table in which each row contains one returned item, with a titled column for each data element you select under **Grid columns**. **Tiles:** Select this option if you want the search results to be presented as an HTML table in which each table cell contains all the content for a single item, and each row contains the number of cells you specify in the **per row** list. ***N* per row:** This list is enabled if you selected the **Tiles** option. Specify the number of tiles, or table cells, that you want on each row. For example, if you specify `2`, there are will be five rows in the initial presentation (assuming that there are at least ten items to be returned).  **Note:** If you specify a number other than `1`, `2`, `5`, or `10`, (that is, a number evenly divisible by ten), the last row will contain fewer tiles than the other rows. **Enable selection of rows:** Lets users populate a control such as a text box with the data from a selected row in the SBM User Workspace.
Columns *(Grid only)*	**Result data:** Shows the data elements you can include in the results of the search. As you type in the box above the list, SBM Composer filters the list to show only the data elements with names that contain the characters you typed. Double-click a data element in the list (or select one and click the right arrow) to add the data element from the **Result data** list to the **Grid columns** list.  **Note:** Adding a data element to the **Grid columns** list does not remove it from the **Result data** list. You can add the same data element to the **Grid columns** list multiple times. **Grid columns:** Shows the data elements you can include in the results of the search, and defines the order (from left to right) in which they appear on the form. Select the data elements in this list, and use the up and down arrows to change the order in which the data elements appear as columns in the results of the search. Double-click a data element in the list (or select one and then click the left arrow) to remove the data element from the list.

Field	Description
Content *(Tiles only)*	Type or paste the content for each tile, in the returned results. Alternately, you can use the string builder tool to insert references to table fields and form controls. For example, you could use the following code to put a small HTML table in each tile: `<table width='100%'> <tr> <td rowspan='2' width='80px'>{SmallImage}</td> <td>{ItemAttributes.Title}</td> </tr> <tr> <td>{OfferSummary.LowestNewPrice}</td> </tr> </table>`
Sort Order	Select one of the Amazon-defined values to specify the order in which the returned results are listed in the SBM User Workspace.

Refresh Tab

See Refresh Tab of the Widget Property Editor [page 247].

Example

A literature student wants to order a new book about William Shakespeare and wants to spend between $25.00 and $50.00 for it. Her school has an application that lets students specify criteria such as keywords, price range, and condition when they select books to order. The Amazon Search widget, which is embedded on a custom *transition form* [page 700], returns a set of results based on that criteria.

To configure this widget:

1. Add the following fields to the primary table of the process app:

Name	Type	Values
Condition	Multi-Selection	New Used
Keywords	Text	Not applicable
Maximum Price	Text	Not applicable
Minimum Price	Text	Not applicable

2. Add a custom transition form.

3. Drag the **Amazon Search** widget from the **Widgets** section of the **Form Palette** to the form editor.
4. Complete the **General** tab of the widget Property Editor. For details, see the "General Tab" section in this topic.
   a. Type `Books` as the **Control** name and **Caption**.
   b. Paste your amazon.com access key ID, type a forward slash (/), and then paste your access secret in the **Access key** box.
   c. Select all **Options** check boxes.
5. Complete the **Query** tab of the widget Property Editor. For details, see the "Query Tab" section in this topic.
   a. Select **Books** from the **Category** list.
   b. In the **Keywords** box, type { and then select **Keywords**. This means that the keyword the user types into the *Keywords* field in the SBM User Workspace is used as search criteria.
   c. Leave the **Title** and **Author/Artist** boxes blank.
   d. In the **Condition** list, select **<Custom>**. In the box to the right of the list, type { and then select **Condition**. This means that the condition that the user selects from the *Condition* field in the SBM User Workspace is used as search criteria.
   e. In the **Price (min)** box, type { and then select **MinPrice**. In the **Price (max)** box, type { and then select **MaxPrice**. This means that the prices the user types in the *MinPrice* and *MaxPrice* fields in the SBM User Workspace are used as search criteria.
6. Complete the **Result** tab of the widget Property Editor. For details, see the "Result Tab" section in this topic.
   a. Move the **OfferSummary.LowestNewPrice** and **SalesRank** fields from the **Result data** list to the **Grid columns** list.
   b. Select **Relevance** from the **Sort order** list.
7. On the **Refresh** tab, make sure the **On page load** and **On data change** check boxes are selected. The **On data change** list should contain all of the fields you mapped on the **Query** tab.
8. Deploy the process app.
9. Submit an issue in the SBM User Workspace.

   The widget should be present on the **New** state form, and should show the books that were found based on the search criteria.

Embedded Report Widget

The Embedded Report widget allows you to embed a Serena Business Manager report in a custom transition or state form.

After you save the report in the SBM User Workspace, you can obtain a special URL that contains the report reference name. The report reference name must be used if you want the embedded report to work after the process app containing the report is promoted to another environment. For more information, see Referencing a Report [page 250].

General Tab

The following table describes the fields on the **General** tab of the Property Editor for this widget.

Field	Description
Control name	Type the name by which the widget is uniquely identified. No other widget or control on this form can have the same name.
Caption	Type the text to appear in the title bar above the widget. If you leave this field blank, the widget will not have a title bar.
Report name	Displays the name of the configured SBM report. If none has been configured, click **Configure Report**, and complete the dialog box that opens as described in Embedded Report Configuration Dialog Box [page 631]. Click **Clear** to remove the configuration and start over.
Options	**Border:** Select this check box to draw a thin line around the widget. **Scroll bars:** Select this check box to include scroll bars in the widget. This provides more content than the allotted space.
Description	Type an optional description of the widget.

Query Tab

The **Query** tab lets you override the parameters and values for the URL specified in the Embedded Report Configuration Dialog Box [page 631]. The parameters are the labels on the left side of the **Query** tab. The values to their right can be static values or be bound to other field or control values. See Using the String Builder Tool [page 249] for information about binding values.

Refresh Tab

See Refresh Tab of the Widget Property Editor [page 247].

Flash Widget

The Flash widget enables users to view Flash movies and applications (`.swf` files).

This topic describes how to configure the Flash widget.

General Tab

The following table describes the fields on the **General** tab of the widget Property Editor.

Field	Description
Control name	Type the name by which the widget is uniquely identified. No other widget or control on this form can have the same name.
Caption	Type the text to appear in the title bar above the widget. If you leave this field blank, the widget will not have a title bar.
Content	Type or paste valid Flash code (typically an `<embed...>` or `<object ...>` tag) in this field. When the form is displayed in the SBM User Workspace, the code is rendered by the Web browser. You can make the widget more dynamic by using the *string builder tool* [page 698] to replace content and parameters in the Flash code.  **Note:** For information about this tool, see Using the String Builder Tool [page 249].  **Important:** If the code you type in this field includes any opening or closing braces ("{" or "}"), precede each of them with a backslash ("\"). Otherwise, SBM Composer tries to interpret the text inside of the braces as the name of a field or control on the form. (If you paste code with braces, you are prompted whether you want a backslash to be added. If you use the Treat '{' as a literal option in the string builder tool, a backslash is automatically added.)  **Note:** This widget is not sized automatically. To make the size match the **Width** and **Height** fields in the Ribbon, you must add `{_width}` or `{_height}` elements to the **Content** field.
Options	**Border:** Select this check box to draw a thin line around the widget. **Scroll bars:** Select this check box to include scroll bars in the widget. This provides more content than the allotted space.
Description	Type an optional description of the widget.

Refresh Tab

See Refresh Tab of the Widget Property Editor [page 247].

Flickr Widget

The Flickr widget lets users access photos and slideshows at flickr.com.

This topic describes how to configure the FlickrWidget.

General Tab

The following table describes the fields on the **General** tab of the widget Property Editor.

Field	Description
Control name	Type the name by which the widget is uniquely identified. No other widget or control on this form can have the same name.
Caption	Type the text to appear in the title bar above the widget. If you leave this field blank, the widget will not have a title bar.
Options	**Border:** Select this check box to draw a thin line around the widget. **Scroll bars:** Select this check box to include scroll bars in the widget. This provides more content than the allotted space.
Description	Type an optional description of the widget.

Options Tab

The following table describes the fields on the **Options** tab of the widget Property Editor.

Field	Description
Content type	**Photo:** Select this option to display a single Flickr photo. If you select this option, the **Photo URL** and **Click URL** fields are displayed. **Slideshow:** Select this option to display a Flickr slide show. If you select this option, the **Tags**, **User ID**, **Set ID**, and **Sort order** fields are displayed.
Photo URL *(Photo option only)*	1. On the Flickr page that shows the photo that you want to use on your form, click the all sizes button that is above the photo. 2. Click the link for the size you want to use. The sizes depend on the contributor, and could include small, medium, and large. The size (in pixels) is shown below each link. 3. Right-click the photo and select the copy link location link or select Properties and then copy the location of the photo (using `http` through `.JPEG`). 4. When you have the size you want, copy the URL to the Windows clipboard. 5. In SBM Composer, paste the URL into the **Photo URL** field.
Click URL *(Photo option only)*	Flickr slide shows cannot be hyperlinks, because they include mouse-driven user interaction. If you want the photo to be a clickable link to a Web page, type the URL of that Web page in this field.

Field	Description
Tags *(Slideshow option only)*	Type one or more Flickr tags to search for photos with these tags. Use commas to separate tags. Alternately, you can use the *string builder tool* [page 698] to insert references to table fields and form controls.  **Note:** For information about this tool, see Using the String Builder Tool [page 249].  **Note:** If you specify a Flickr tag, you can also specify a user ID to narrow the search. If you specify a set ID, however, the tag and user ID are ignored, because a set is a specific collection of photos.
User ID *(Slideshow option only)*	Type a Flickr contributor's user ID to search for photos by that contributor. Alternately, you can use the string builder tool to insert references to table fields and form controls. Click **Find Flickr ID** to open a Flickr tool that determines the Flickr user ID or group ID that is based on the address of a Flickr photo stream or group pool.  **Note:** If you specify a Flickr user ID, you can also specify a tag to narrow the search. If you specify a set ID, however, the tag and user ID are ignored, because a set is a specific collection of photos.
Set ID *(Slideshow option only)*	Type the Flickr set ID to display a specific photo set. Alternately, you can use the string builder tool to insert references to table fields and form controls.  **Note:** If you specify a set ID, the **Tags** and **User ID** fields are ignored, because a set is a specific collection of photos.
Sort order *(Slideshow option only)*	Select one of the Flickr-defined options: **Date Posted**, **Date Taken**, or **Most Interesting**. For example, if you select **Date Taken**, the most recent photos are displayed first.

Refresh Tab

See Refresh Tab of the Widget Property Editor [page 247].

Google Gadget Widget

The Google Gadget widget provides easy access to useful tools and information, such as a Google calendar viewer.

This topic describes how to configure the Google Gadget *widget* [page 701].

General Tab

The following table describes the fields on the **General** tab of the Property Editor for this widget.

Field	Description
Control name	Type the name by which the widget is uniquely identified. No other widget or control on this form can have the same name.
Caption	Type the text to appear in the title bar above the widget. If you leave this field blank, the widget will not have a title bar.
Content	Do the following: 1. Click **Search for Google Gadget**. 2. Find the gadget you want to use, and click the button that adds it to your Web site. 3. Experiment with the gadget settings until you are satisfied, and then click the button that gets the code. 4. Select all of the code, and copy it to the Windows clipboard. 5. Paste the copied code into the **Content** box in SBM Composer. Alternately, you can use the *string builder tool* [page 698] to insert references to table fields and form controls. **Note:** For information about this tool, see Using the String Builder Tool [page 249]. **Important:** If the code you type in this field includes any opening or closing braces ("{" or "}"), precede each of them with a backslash ("\"). Otherwise, SBM Composer tries to interpret the text inside of the braces as the name of a field or control on the form. (If you paste code with braces, you are prompted whether you want a backslash to be added. If you use the Treat '{' as a literal option in the string builder tool, a backslash is automatically added.) **Note:** This widget is sized automatically. You cannot change the size of the widget by adding `{_width}` or `{_height}` elements into the **Content** field.
Options	**Border:** Select this check box to draw a thin line around the widget. **Scroll bars:** Select this check box to include scroll bars in the widget. This provides more content than the allotted space.
Description	Type an optional description of the widget.

Refresh Tab

See Refresh Tab of the Widget Property Editor [page 247].

HTMLJavaScript Widget

The HTML/JavaScript widget lets you embed almost any valid HTML or JavaScript into a custom form.

This topic describes how to configure the HTML/JavaScript widget.

General Tab

The following table describes the fields on the **General** tab of the Property Editor for this widget.

Field	Description
Control name	Type the name by which the widget is uniquely identified. No other widget or control on this form can have the same name.
Caption	Type the text to appear in the title bar above the widget. If you leave this field blank, the widget will not have a title bar.
Content	Type or paste the HTML or JavaScript code in this field. When the form is displayed, the code is rendered by the Web browser. Alternately, you can use the *string builder tool* [page 698] to insert references to table fields and form controls. **Note:** For information about this tool, see Using the String Builder Tool [page 249]. **Note:** If you use the string builder tool in JavaScript code in the HTML/JavaScript widget, the code needs to be in an iframe (inline frame). If the code is in an iframe, it cannot reference other parts of the form. Therefore, the string builder tool should not be used in JavaScript code. **Important:** If the code you type in this field includes any opening or closing braces ("{" or "}"), precede each of them with a backslash ("\"). Otherwise, SBM Composer tries to interpret the text inside of the braces as the name of a field or control on the form. This prevents the form from working properly. (If you paste the brace, you are prompted whether you want a backslash to be added. If you use the Treat '{' as a literal option in the string builder tool, a backslash is automatically added.)
Options	**Border:** Select this check box to draw a thin line around the widget. **Scroll bars:** Select this check box to include scroll bars in the widget. This provides more content than the allotted space.
Description	Type an optional description of the widget.

Refresh Tab

See Refresh Tab of the Widget Property Editor [page 247].

PDF Widget

The PDF widget lets you embed a PDF document in a custom form in the SBM User Workspace, or open it in another window. The PDF document contains input fields that are typically empty until the process app runs. At that time, the input fields are populated with system data and data that users enter in fields on transition forms. The widget includes standard PDF document features, such as printing and saving to file. This means users can have a printed version of process app data, and can e-mail the PDF document containing the process app data to others.

Important: PDF documents created with Adobe® Acrobat® 9 Pro are supported. The PDF documents must be PDF version 1.4 or later.

Remember: To view PDF documents and use their features, users must have Adobe® Reader® installed on their computers. To view PDF documents in preview mode or test your process app, you must have this application installed as well. If this application is not installed, a link to the Adobe Reader download page is displayed.

Note: In preview mode, the PDF *template* [page 699] is displayed, but the input fields are not populated with runtime data.

Note: The input field names can contain a maximum of 128 characters.

Tip: To support Unicode characters, make sure that "plain text" is not enabled for fields during the creation of the PDF document.

General Tab

The following table describes the fields on the **General** tab of the Property Editor for this widget.

Field	Description
Control name	Type the name by which the widget is uniquely identified. No other widget or control on this form can have the same name.
Caption	Type the text to appear in the title bar above the widget. If you leave this field blank, the widget will not have a title bar.
Template	Browse to a PDF document that contains interactive form fields that will be filled in when the process app runs. This is the PDF document that users can print or save to file. **Important:** You are asked to select another PDF document if you select a PDF document that does not allow interactive form fields to be filled in, or that requires a password or certificate to open.

Field	Description
Display in	**Form:** Select this option if you want the PDF document to be displayed within the form. **New Window:** Select this option if you want the PDF document to be displayed in a new window.
Options	**Border:** Select this check box to draw a thin line around the widget. **Scroll bars:** Select this check box to include scroll bars in the widget. This provides more content than the allotted space.
Description	Type an optional description of the widget.

Parameters Tab

The **Parameters** tab contains the input fields that are in the PDF document you imported. You map table fields and form controls into the fields on this tab.

For example, suppose you have an *Item Type* [page 688] field on a Submit form. In the PDF document, you also have an **Item Type** input field. In this field on the **Parameters** tab, type { and then select **Item Type**. When a user selects a value for the *Item Type* field in the SBM User Workspace, the field in the PDF document is populated with the same value.

Important: The mapping of table fields and controls is preserved when you reimport a PDF document that was used as a template. For example, mapping is not affected if you change the type of a field in the PDF document, as long as the new field type contains inputs with the same names as the associated field in the SBM form. For example, you can change a Combo Box control to a List Box control, as long as the values that users can select are the same as the values configured for the associated field in SBM Composer.

Required PDF input fields are displayed on this tab with a red asterisk (*).

Note: For information about mapping fields and controls, see Using the String Builder Tool [page 249].

Note: You can type default values for fields on the form instead of mapping.

Refresh Tab

Use the **Refresh** tab to update values that refer to external data, such as field values. If the widget does not refer to any external data, the **Refresh** tab options are disabled.

The following table describes the fields on the **Refresh** tab of the Property Editor for this widget.

Field	Description
Refresh contents	**On page load:** If this check box is selected, the content of the widget is retrieved when the form is initially displayed in the SBM User Workspace or in the form preview.  **Note:** This check box is selected by default. **On data change:** If this check box is selected, the content of the widget is updated when fields that map to values on the **Parameters** tab are changed in the SBM User Workspace or in the form preview. **On click:** In the drop-down list box, select one of the buttons, images, or hyperlinks that are on this form. The content of the widget is updated when the control is clicked in the SBM User Workspace or in the form preview.

Supported SBM Field Types

All fields under **Field Types** in the **Table Palette** are supported except for *Summation* fields.

All fields under **System Fields** in the **Table Palette** are supported except for *Secondary Owner* [page 696].

All special fields are supported. These fields are available from the *string builder tool* [page 698]. (For more information, see Using the String Builder Tool [page 249].) Special fields include the following:

- `_height`
- `_width`
- `_ProjectID`
- `_RecordID`
- `_TableID`

Supported PDF Field Types

The following PDF *field types* [page 686] are supported:

- Check Box
- Drop-down List/Combo Box
- List Box
- Radio Button
- Text

Mapping Rules for Fields

SBM fields can be mapped to any PDF field type, but the values must match exactly. For example:

- If you are mapping a SBM *Single Selection* field to a PDF List Box field type, you must create the field with the same values in each *application* [page 679]. For example, if you add **Defect**, **Enhancement**, and **Internal Task** as values for a *Single Selection* field called "Request Type" in SBM Composer, you must enter the same values as list items in the List Box field in the PDF form.
- Similarly, if you are mapping a SBM *Single Selection* field to a PDF Radio Box field type, the values must match. If you add **Vacation**, **Sick**, and **Jury Duty** as values for a *Single-Selection* field called "Time-Off Type" in SBM Composer, you must enter the same values as item names in the PDF form.

Note: An exception to this rule is when you map a SBM *Binary* field to a Check Box PDF field type that is used to toggle between two settings. In this situation, the mapping is handled internally, and the values do not have to match. There is no adverse effect if the PDF form requires values of "On" and "Off", and the SBM *Binary* field requires "Yes" and "No."

Error Messages

The following error messages could be generated during the configuration of the PDF widget.

- `PDF Template does not contain field with name.`
- `PDF Template contains field of unknown type.`
- `Invalid values.`
- `No field mapping found in the PDF Template.`
- `Unable to read fields from PDF Template.`

To correct these problems, modify or replace the PDF document being used as the template.

REST Grid Widget

REST is an acronym for "representational state transfer." It is a type of software architecture for distributed hyperlinked systems such as the Web. It transmits domain-specific data, and outlines how resources are defined and addressed. The REST Grid widget lets you access external data (through a REST service) and display it in a tabular format in a process app. You can pull data from any Web service that supports the REST format. These Web services include the XML and JSON REST Web services.

When you type a URL to a REST service that requires basic authentication, a popup window is automatically presented, in which you type the user name and password. (The REST service provider sends you the user name and password when you register for the service.)

In SBM Composer you can only configure REST services that can return repeating (iterating) elements. However, in the SBM User Workspace, the REST Grid widget can display a single item if a particular REST service call returns only one element.

Important: If the SBM Server is set up with Microsoft Windows NT Challenge/ Response authentication, the SBM Server must be configured in a specific way to ensure that the REST Grid widget works. For more information, see the *Serena® Business Manager Installation and Configuration Guide*.

Note: During widget configuration, you will receive a warning message if the amount of data in the result set exceeds 100 rows. The message includes ways to limit the result set.

General Tab

The following table describes the fields on the **General** tab of the Property Editor for this widget.

Field	Description
Control name	Type the name by which the widget is uniquely identified. No other widget or control on this form can have the same name.
Caption	Type the text to appear in the title bar above the widget. If you leave this field blank, the widget will not have a title bar.
URL	Displays a URL to a configured REST service. If none has been configured, click **Configure URL**. Complete the dialog box that opens as described in REST Service Configuration Dialog Box [page 641]. Click the **Clear** button to remove the URL and start over.
Options	**Border:** Select this check box to draw a thin line around the widget. **Scroll bars:** Select this check box to include scroll bars in the widget. This provides more content than the allotted space. **Paging:** Select this check box to include buttons at the top of the widget that let users navigate between pages in the result set. **Note:** The configured REST service must support paging. **Include lower page bar:** Select this check box if you want to include paging buttons at both the top and the bottom of the widget.
Description	Type an optional description of the widget.

Query Tab

The **Query** tab lets you override the parameters and values for the REST service specified in the REST Service Configuration Dialog Box [page 641]. The parameters are the labels on the left side of the **Query** tab. The values to their right can be static values or be bound to other field or control values.

If **Paging** is selected on the **General** tab, you must map a parameter responsible for paging to the `RESTGridWidget._currentPage` or `RESTGridWidget._currentIndex` parameter.

For example, suppose the **URL** field on the **General** tab contains `http://search.twitter.com/search.json?lang=en&q=devo&rpp=15&page=2`. On the **Query** tab, use the *string builder tool* [page 698] to associate the `RESTGridWidget._currentPage` variable with the **page** parameter. Otherwise, during validation, you receive a warning message, and when the user pages, the previous results are displayed. (See Using the String Builder Tool [page 249] for instructions.)

Result Tab

The following table describes the fields on the **Result** tab of the Property Editor for this widget.

Field	Description
Render as	**Grid:** Select this option if you want the search results to be presented as an HTML table in which each row contains one returned item, with a titled column for each data element you select under **Grid columns**. **Tiles:** Select this option if you want the search results to be presented as an HTML table in which each table cell contains all the content for a single item, and each row contains the number of cells you specify in the per row list. ***N* per row:** This list is enabled if you selected the **Tiles** option. Specify the number of tiles, or table cells, that you want on each row. For example, if you specify `2`, there are will be five rows in the initial presentation (assuming that there are at least ten items to be returned). **Note:** If you specify a number other than `1`, `2`, `5`, or `10` (that is, a number evenly divisible by ten), the last row will contain fewer tiles than the other rows. **Enable row selection and use of row data as field content:** Lets users populate a control such as a text box with the data from a selected row in the SBM User Workspace. If this check box is cleared, the **Columns the grid will display** table contains an additional **Action on click** column.

Field	Description
Columns *(Grid only)*	**Data returned by service:** Shows the data elements you can include in the result set. As you type in the box above the list, SBM Composer filters the list to show only the data elements with names that contain the characters you typed. Double-click a data element in the table (or select one and click the right arrow) to move the data element from the **Data returned by service** table to the **Columns the grid will display** table. **Columns the grid will display:** The **Value** column shows the data elements from the widget that you can include in the result set. In the **Column name** column, you can accept the default column name to be displayed in the SBM User Workspace or type another name. In the **Display as** column, select **Text**, **Url**, **Image**, or **Hidden**. (*Hidden* means that the data will not be visible as a column in the grid, but the data can still be mapped as inputs to other fields.) In the **Action on click** column, select the action to be taken when an user clicks the field in theSBM User Workspace. The **Action on click** column is not available if you are binding to widget data, because when you bind to widget data, there are other links, and users would find it difficult to determine what they are clicking.  **Note:** For more information about binding to widget data, see Binding to Widget Data [page 248]. To remove a data element from the list, double-click the data element or select the data element and then click the left arrow.

Grid columns *(Tiles only)*	Type or paste the content for each tile in the result set.
	
	Alternatively, you can use the *string builder tool* [page 698] to insert references to table fields and form controls.
	
	For example, you could use the following code to put a small HTML table in each tile:
	
	`<table width='100%'>`
	`  <tr>`
	`  <td rowspan='2' width='80px'>{SmallImage}</td>`
	`  <td>{ItemAttributes.Title}</td>`
	`  </tr>`
	`  <tr>`
	`  <td>{OfferSummary.LowestNewPrice}</td>`
	`  </tr>`
	`</table>`

Refresh Tab

See Refresh Tab of the Widget Property Editor [page 247].

Example 1

As part of creating a product incident request, a service representative has to select the product that an incident is related to from a list. Instead of replicating the product data in SBM to populate the list, a REST Grid widget can be used to retrieve the information

dynamically from the company's product catalog. The REST Grid widget can also display product images, so the representative has a visual way to select the product.

Example 2

A writer for a financial magazine has an item assigned to him for each date he needs to submit an article for publication. The custom form for the **New** state has a REST Grid widget that returns the financial questions people asked on the magazine's Web site. The writer uses these questions as ideas for the content of the article.

To configure this widget:

1. Add a custom state form.
2. Drag the **REST Grid** widget from the **Widgets** section of the **Form Palette** to the form editor.
3. Complete the **General** tab of the widget Property Editor:
   a. Type `Finance` in the **Control name** box.
   b. Type `Common Questions` in the **Caption** box.
   c. Click the **Configure URL** button. The **REST Service Configuration** dialog box opens.
   d. Paste `http://answers.yahooapis.com/AnswersService/V1/questionSearch?appid=YahooDemo&query=stocks&output=json` in the address combo box.
   e. Click **Update outputs**, and then click **OK** to close the dialog box.
   f. Select all of the **Options** check boxes.
4. On the **Result** tab of the widget Property Editor, move **Data returned by service** items to the **Columns the grid will display** table. For example, move the **CategoryName**, **Subject**, **Content**, and **Date** items.
5. On the **Forms** tab of the Property Editor for the **New** state, select the new custom form.
6. Deploy the process app.
7. Submit an issue in the SBM User Workspace.

   The widget should be present on the **New** state form. The columns you specified on the **Result** tab of the widget Property Editor are displayed and populated with data.

Silverlight Widget

The Silverlight widget lets you embed Silverlight applications in your application.

Note: To embed a Silverlight *widget* [page 701] from another server or in an on-premise environment, download the Visifire components from http://www.visifire.com/download_silverlight_charts.php and install them in the `\productName\Application Engine\bin` directory on the SBM Application Engine Web Server. Contact your Application Engine server administrator for assistance if you do not have access to this server.

For detailed Visifire installation instructions, visit http://visifire.com/visifire_charts_documentation.php and read the "Introduction" section.

Because the Silverlight widget is not embeddable across sites, embed its content in the Web Page widget. For information about the Web Page widget, see Web Page Widget [page 245]. This topic describes how to configure the Silverlight widget.

General Tab

The following table describes the fields on the **General** tab of the Property Editor for this widget.

Field	Description
Name	Type the name by which the widget is uniquely identified. No other widget or control on this form can have the same name.
Caption	Type the text to appear in the title bar above the widget. If you leave this field blank, the widget will not have a title bar.
Content	Type or paste the valid Silverlight code. When the form is displayed, the code is rendered by the Web browser. Alternately, you can use the *string builder tool* [page 698] to insert references to table fields and form controls. **Note:** For information about this tool, see Using the String Builder Tool [page 249]. **Important:** If the code you type in this field includes any opening or closing braces ("{" or "}"), precede each of them with a backslash ("\"). Otherwise, SBM Composer tries to interpret the text inside of the braces as the name of a field or control on the form. (If you paste code with braces, you are prompted whether you want a backslash to be added. If you use the Treat '{' as a literal option in the string builder tool, a backslash is automatically added.) **Tip:** An alternative way of doing this is to create a Web page at an accessible URL, include the Silverlight content on that page, and specify the URL of the page in a Web Page widget that you place on your custom form.
Options	**Border:** Select this check box to draw a thin line around the widget. **Scroll bars:** Select this check box to include scroll bars in the widget. This provides more content than the allotted space.
Description	Type an optional description for the widget.

Refresh Tab

See Refresh Tab of the Widget Property Editor [page 247].

Web Page Widget

The Web Page *widget* [page 701] lets users view the content of an HTML page.

General Tab

The following table describes the fields on the **General** tab of the Property Editor for this widget.

Field	Description
Control name	Type the name by which the widget is uniquely identified. No other widget or control on this form can have the same name.
Caption	Type the text to appear in the title bar above the widget. If you leave this field blank, the widget will not have a title bar.
URL	Type the address of the Web page you want to display. Alternately, you can use the *string builder tool* [page 698] to insert references to table fields and form controls.  **Note:** For information about this tool, see Using the String Builder Tool [page 249]. When the form is displayed, the widget is replaced with the content from that URL (along with a caption, a border, and scroll bars, if you enable those options.)
Options	**Border:** Select this check box to draw a thin line around the widget. **Scroll bars:** Select this check box to include scroll bars in the widget. This provides more content than the allotted space. **Use SSO Authentication:** Select this check box to enable Single Sign-On (SSO) authentication for SSO-enabled URLs.
Description	Type an optional description for the widget.

Refresh Tab

See Refresh Tab of the Widget Property Editor [page 247].

WidgetBox Widget

Widget Box is a repository of widgets you can select from and include on your Web page. For example, you can embed widgets into your social networking profile, and embed chat rooms in your Web page.

This topic describes how to configure the WidgetBox *widget* [page 701].

General Tab

The following table describes the fields on the **General** tab of the Property Editor for this widget.

Field	Description
Control name	Type the name by which the widget is uniquely identified. No other widget or control on this form can have the same name.
Caption	Type the text to appear in the title bar above the widget. If you leave this field blank, the widget will not have a title bar.
Content	Do the following: 1. Click **Search for WidgetBox widget**. 2. On the WidgetBox site, find the widget you want to use, click the Get Widget button, and then copy the Widget code to the Windows clipboard. 3. In SBM Composer, paste the code into the **Content** box. **Important:** If the code you type in this field includes any opening or closing braces ("{" or "}"), precede each of them with a backslash ("\"). Otherwise, SBM Composer tries to interpret the text inside of the braces as the name of a field or control on the form. This prevents the form from working properly. (If you paste the brace, you are prompted whether you want a backslash to be added. If you use the Treat '{' as a literal option in the *string builder tool* [page 698], a backslash is automatically added.) Alternately, you can use the string builder tool to insert references to table fields and form controls. **Note:** For information about this tool, see Using the String Builder Tool [page 249].
Options	**Border:** Select this check box to draw a thin line around the widget. **Scroll bars:** Select this check box to include scroll bars in the widget. This provides more content than the allotted space.
Description	Type an optional description of the widget.

Refresh Tab

See Refresh Tab of the Widget Property Editor [page 247].

YouTube Widget

The YouTube widget lets you embed YouTube videos on your Web site.

This topic describes how to configure the YouTube *widget* [page 701].

General Tab

The following table describes the fields on the **General** tab of the Property Editor for this widget.

Field	Description
Name	Type the name by which the widget is uniquely identified. No other widget or control on this form can have the same name.
Caption	Type the text to appear in the title bar above the widget. If you leave this field blank, the widget will not have a title bar.
URL/Video ID	Type or paste the URL of the YouTube Web page for the video you want to display. Alternately, you can enter only the video ID portion of the URL. The video ID is the text between `v=` and `&` in the URL. Entering the video ID (such as `RBQoWgXepa4`) has the same effect as entering the entire URL (`http://youtube.com/watch?v=RBQoWgXepa4&feature=related`). Instead of typing or pasting the URL, you can use the *string builder tool* [page 698] to insert references to table fields and form controls.  **Note:** For information about this tool, see Using the String Builder Tool [page 249].
Options	**Border:** Select this check box to draw a thin line around the widget. **Scroll bars:** Select this check box to include scroll bars in the widget. This provides more content than the allotted space.
Description	Type an optional description of the widget.

Refresh Tab

See Refresh Tab of the Widget Property Editor [page 247].

Refresh Tab of the Widget Property Editor

This topic describes how to configure the **Refresh** tab in the Property Editor for each *widget* [page 701].

Refresh Tab

Use the **Refresh** tab to specify when values that refer to external data, such as field values, are updated. If the widget does not refer to any external data, the **Refresh** tab options are disabled.

The following options are available for the **Refresh contents** field on the **Refresh** tab.

On page load: If this check box is selected, the content of the widget is retrieved when the form is initially displayed in the SBM User Workspace or in the form preview.

Note: This check box is selected by default.

On data change: If this check box is selected, the content of the widget is updated when the values of certain fields are changed in the SBM User Workspace or in the form preview. These are the fields that are bound to database fields in the widget Property Editors. (See Using the String Builder Tool [page 249] for information about binding fields.)

On click: In the drop-down list box, select one of the buttons, images, or hyperlinks that are on this form. The content of the widget is updated when the control is clicked in the SBM User Workspace or in the form preview.

Binding to Widget Data

In grid-style widgets, users can access the data in the grid to populate controls such as edit boxes. In the SBM User Workspace, users select the row in the grid whose data they want to put in the control. The grid-style widgets include the Amazon Search *widget* [page 701] and the REST Grid widget.

Note: If the form is a *primary table* [page 692] form (for example, in a workflow), then only primary table data can be accessed. If the form is for an *auxiliary table* [page 680], then only auxiliary table data can be accessed.

To bind to widget data:

1. Drag a widget from the **Form Palette** to the form and configure it.

   Note: For information about configuring the Amazon Search widget, see Amazon Search Widget [page 224]. For information about configuring the REST Grid widget, see REST Grid Widget [page 239].

2. Drag a control such as an edit box from the **Form Palette** to the form and give it a label. For example, if you are going to populate the edit box with addresses from the REST Grid widget, give the label the name `Addresses`. If you are going to populate the edit box with authors from the Amazon Search widget, give the label the name `Authors`.

3. Enlarge the widget so that it consumes the whole form. To do so, perform the following steps:

   a. Select the form. Sizing indicators are displayed for each column and row.

   b. Click the sizing indicator for the column containing the widget until it changes to **%**. Repeat this step for the row.

   c. Select the widget and click the **Autofill Vertically** icon in the **Alignment** area.

   Note: For more information about resizing, see Resizing Columns and Rows [page 253].

4. In the Property Editor for the widget, on the **Results** tab, select **Enable row selection**, if it is not already selected.

   Note: If you clear this check box, the **Grid columns** table contains an additional **Action on click** column. This option is not available if you are binding to widget data, because when you bind to widget data, there are other links, and it would be unclear what you are clicking.

5. Still in the Property Editor, on the **Results** tab, move fields you want to bind from the **Result data** table to the **Grid columns** table.
6. Select the control on the form.
7. In the Property Editor, click the **Refresh** tab.
8. In the **Contents** field, use the *string builder tool* [page 698] to enter the grid data you want users to select to populate this edit box.

   Note: For information about using the string builder tool, see Using the String Builder Tool [page 249].

In the SBM User Workspace, users can select a row of data, and the control is populated with that data.

Using the String Builder Tool

The "string builder" tool lets you insert references to table fields and form controls, and insert references to grid data in grid-style widgets. This creates dynamic forms and widgets that are driven by *application* [page 679] and grid widget data and the actions of the user. You can combine static data and dynamic data elements in the string. The dynamic data elements represent replaceable text that drives the appearance of input data. The dynamic data is either stored in the database table or on the form.

You can also use the *string builder tool* [page 698] to insert HTML tags. This functionality is available in fields that contain text, such as the **Content** field in the HTML/JavaScript *widget* [page 701], and the **Display text** field in a Text control.

Note the following points:

- If you use the string builder tool in JavaScript code in the HTML/JavaScript widget, the code will be generated into an iframe (inline frame). If the code is in an iframe, the widget cannot reference other parts of the form. Therefore, the string builder tool should not be used in JavaScript code.
- The **Treat '{' as a literal** option in the string builder tool menu is used to start typing JavaScript code. To use { as a literal character, it must be "escaped" with a backslash (\). The backslash is automatically added when you select this option from the string builder tool menu.

To use the string builder tool:

1. Type { in one of the fields that says **Type { to add a dynamic value**.
2. In the menu that opens, select an appropriate control, field, or data element.

The following examples illustrate how the string builder tool works.

- For the Flickr widget, **Tags** field, you could select **Title**. This means that whatever the user types in the **Title** field on the form becomes the value of the **Tags** field. The search would look for photos whose tag equals this value.
- For the Amazon Search widget, you could add a combo box to the form and type a selection of categories from which the user can select. This reduces the number of categories the user has to scan. Then, in the **Category** field, you could select **<Custom>** from the list, and in the text box, type `{` and then select the **Category** combo box from the menu.
- For the Flickr widget, you could type a constant value for the **Tags** field, and then for the **User ID** field, type `{` and select **Owner**. Because in this case, the **Owner** field in the database contains Flickr user IDs, this field is dynamically populated with the user ID.
- For the Google Gadget and Flash widgets, in the **Content** field, you could type `{`, select **_width**, type `{`, and then select **_width**. This causes the content of the widget to automatically take the value of the **Width** and **Height** fields in the **Size** area of the Ribbon.

  Note: The Google Gadget and Flash widgets are automatically sized, so you cannot type a static value for the width and height in the **Content** field. You can type a static value for the width and height for the Silverlight, HTML/JavaScript, and WidgetBox widgets.
- For the Web Page widget, **URL** field, you could type the URL for a social networking site, and then type `{`, select **Candidate First Name**, type `+`, type `{`, and then select **Candidate Last Name**. This creates a URL that takes the user (for example, a hiring manager) to the Web page for that candidate.
- For the HTML/Javascript widget, **Content** field, you could type `zips={` and then select **Title**. **Title** is a text box on the form in which users type their zip code to retrieve weather information.

Referencing a Report

You can embed an SBM report in a custom form. If you use a special report URL that contains a report reference name, the report will still work after the process app is promoted to another environment (such as a production environment).

If you do not use the report reference name, the report URL will contain a report ID and field IDs (for Query at Runtime fields). Report IDs and QAR field IDs change when a process app is promoted to another environment, so the report will not work in the new environment.

To reference a report:

1. Drag an Embedded Report Widget [page 229] from the **Form Palette** to a custom state or transition form.
2. In the SBM User Workspace, create the report you want to reference.

3. Save the report, and in the **Save As** section on the page, type a report name in the **Title** box, and type a reference name in the **Reference Name** box. The reference name must be unique within an application, but can be the same as the report title.

   Restriction:

   - The **Reference Name** box is only present if you have the Deploy privilege, which is granted in SBM Application Administrator.
   - This box is disabled if the **Privilege Category** is **Private**.

4. In the confirmation message, click **Reference Link**. A message opens that contains the report URL.
5. Alternatively, run the report, click the **Copy URL to Clipboard** icon, and then select **Reference Link** in the dialog box that opens.
6. Press CTRL+C to copy the report URL to the Windows Clipboard.
7. Back in SBM Composer, click **Configure Report** on the **General** tab of the widget Property Editor.
8. Paste the URL into the Embedded Report Configuration Dialog Box [page 631] that opens.
9. The report URL may contain parameters that could be mapped to field or control values that are provided by the user when the process app runs. To map these parameters to these values, click the **Query** tab in the widget Property Editor, and follow the instructions in Using the String Builder Tool [page 249].

Copying and Moving Controls

You can copy or move one or more controls from one part of a custom form to another. You can also copy or move them from one custom form to another one, even if the forms are in different applications in the same process app. This makes it easy to maintain custom forms.

To copy or move a control:

1. Open the source form.
2. Select the control on the form.
3. Right-click and select **Copy** if you want to copy the control, or **Cut** if you want to move the control.
4. If you want to copy or move the control to another form, open the target form.
5. Right-click an empty cell and select **Paste.**

Note the following points:

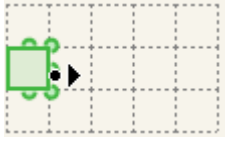
- All of the controls within a **Container** control (for example, a **GroupBox**) are copied or moved with it.
- You cannot add a control that is invalid for the target form (for example, a **State Change History** section cannot be added to a *transition form* [page 700]).

- The target form must have enough empty space to accommodate the controls you want to paste.
- If you paste a control from the **Field Controls** section of the **Form Palette**, the existing field is removed. There can be only one instance of such a control on a form. For example, if you copy the **Title** control and paste it on the same form, only the pasted version remains.
- If you copy a control from the **Field Controls** section to a form in another application, the control is copied only if a matching control is found. For example, suppose you copy an **Incident Type** control to a target form in another application. The target form has an **Item Type** control. Because both of these controls have a **Control name** of "ItemId_Control," the **Incident Type** control will replace the **Item Type** control on the target form.
- If you select multiple controls, the **Paste** operation must be performed in the location relative to where you performed the **Copy** or **Move** operation. For example, suppose you have a row with three cells, and each cell contains a button. Select the three buttons, right-click the last button, and then select **Copy**. To paste, select the last cell in an empty row and then select **Paste**. If you select the first cell, you get a message that the target space does not have enough empty cells. This behavior is consistent with how the drag-and-drop operation works in the form editor.
- If row or column spanning was applied to source cells, the spanning is maintained in the target cells, if there is enough space. If there is not enough space, the existing cell size is used.

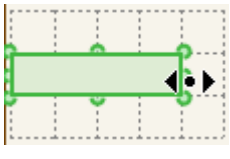
Spanning Columns and Rows

You can span rows or columns by dragging a cell over adjacent, empty cells. You use green, circular "handles" to perform the drag operation. A black arrow on top of each handle indicates the directions in which you can drag. If the arrow does not have a pointer, you cannot drag in that direction. For example, in the following illustration, the arrow points to the right, so you can only drag the cell to the right. The black arrow only appears when you hover over a green handle.

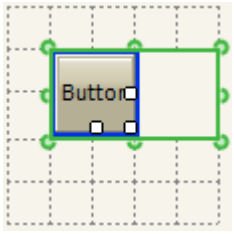
Note: This drag operation replaces the **Column Span** and **Row Span** fields that used to be in the Ribbon.



The following illustration shows a row that was spanned by dragging the cell to the right over three cells. The arrow is bidirectional, so you can drag the handle on the right side of the selection to the right or to the left.



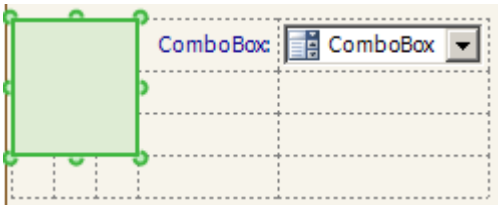
When you select a single control, you can resize the control using the square handles, or the cell using the round handles.



You can select contiguous cells and drag them at the same time. The following illustration shows three selected cells.

The following illustration shows the effect of dragging the three cells at the same time. There are handles at the left boundary of the form, because you could drag the cells from the left to the right.

You cannot drag the selected cell in the following illustration to the right, because it would overlap the combo box control. (The handles on the right indicate that you can drag the cell to the left to make it narrower.



Resizing Columns and Rows

You can manipulate the columns and rows in a selected form or container control. Use columns and rows to position and size the controls that you put on a form.

Note: Postioning and sizing for various field types may vary depending on the Web browser in which they are displayed. For best results, test your custom forms in all browsers used by your users.

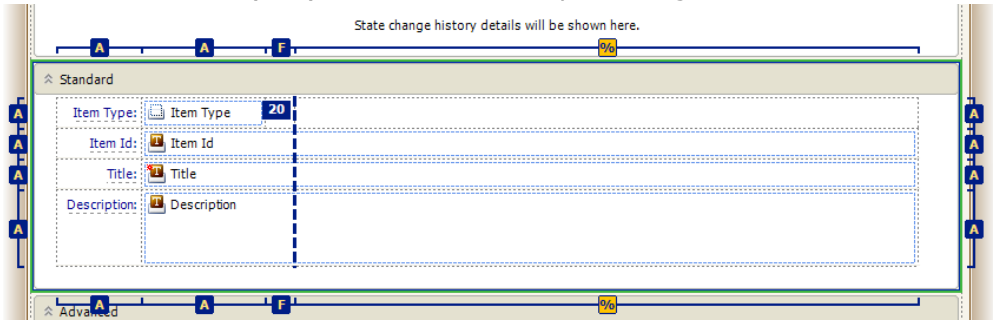
To resize rows and columns:

1. Select the form.
2. In the form Property Editor click the **Columns** tab or the **Rows** tab.
3. Select a column or row. The current sizing behavior is shown in the **Type** column. The following table describes the sizing behaviors.

Sizing Behavior	Description
Fixed	Lets you specify, in pixels, the width of a column or the height of a row. **Note:** If the specified size of the column (row) is smaller than the minimum content size, then the column (row) assumes the size of the content.
Percentage	Lets you specify the percentage of the remaining width (height) that the columns (rows) can occupy after the fixed and autosized columns (rows) are allotted. **Note:** If the specified size of the column (row) is smaller than the minimum content size, then the column (row) assumes the size of the content.
AutoSize	The width of the column or height of the row is automatically set to fit the contents of the column (row). This setting cannot be changed. **Note:** This sizing behavior is incompatible with the justification options on the Ribbon. This is because when a column (row) is autosized, the content is centered and the justification setting is ignored.

Note: If the content of the columns (rows) exceeds the width (height) of the container, scroll bars appear in the form preview and in the SBM User Workspace. When this occurs, there is no space left for percentage columns (rows) and the columns (rows) have the minimum size.

Note: To see whether a column (row) is fixed, autosized, or a percentage, click the form. An **F** indicates that the column or row is of fixed size, an **A** indicates that a column (row) is autosized, and a **%** sign indicates that the size of the column (row) is measured as a percentage of the form.



4. To change a sizing behavior, do one of the following:
   - In the Property Editor, select the **Type** field in one of the rows. In the menu that opens, select the sizing behavior you want.
   - Click the form and then click **F**, **A**, or **%**. This toggles between sizing behaviors.

- Right-click a column (row), select **Column Sizing** or **Row Sizing**, and then select **Fixed**, **Percentage**, or **AutoSize**.

5. To resize a fixed or percentage column (row) from the Property Editor, in the **Width** or **Height** column, type the number of pixels for the column (row) or the percentage of the total width (height) that the column (row) should occupy on the form.
6. To resize a fixed or percentage column (row) by using the drag-and-drop operation, hover the cursor over the right edge of the column or the bottom edge of the row. The cursor turns into the splitter shape, and a bold dotted line highlights the edge and displays the size of the column (row) in pixels or a percentage. Drag the line to change the size of the column (row).

Selecting Parent Controls or Cells on a Form

From a menu, you can select a parent control or cell that you want to modify in a form. This lets you see the parent hierarchy for that control and makes it easy to select a control that is behind another control on the form.

Note: You can also select a control, and then press the Esc key to select the next highest control in the hierarchy.

To select containers:

1. Select a control in a form, or the form itself.
2. Right-click and select **Select Element**. In the menu that opens, a hierarchical list of controls is displayed. The parent control is at the bottom of the list.
3. Click a control in the menu to select it in the form. If you click **(Selected cell)** or **(Containing cell)**, the cell in which the control was placed is selected.

Note: If you drag a field control (or label control), the corresponding label control (or field control) is also moved.

Adding Images and Icons

You can use custom images and icons throughout an *application* [page 679]. For example, you can add your company logo to a custom form, and have your company Web site open when users click this logo. Alternatively, you can select an image to appear as the background for a custom form.

You can also replace certain system-provided icons for primary and auxiliary tables.

To add an image or icon to an application:

1. Select the application to which you want to add an image or icon.
2. Do one of the following:
   - In the App Explorer, right-click the **Images** heading, and select **Add New Image** or **Add New Icon**.
   - In the **New** area of the **Home** tab of the Ribbon, click the down arrow on the **State Form** button, and then click **Image** or **Icon**.

3. Locate the image or icon file and click **Open**. For images, the supported file types are `.bmp`, `.jpg`, `.gif`, and `.png`. For icons, only `.ico` files are supported.
4. After you add the file, the image or icon opens in the image editor. Modify its properties as needed. See Image Editor [page 256] for details.

The image or icon is now available in the following locations:

- **Icons** tab of the table Property Editor (icons only)
- Property Editor for an image control on a custom form
- Background area for a custom form (images only)

Note: You can also add images or icons in the context of a specific custom form or table.

Image Editor

The image editor lets you view and modify the properties of an imported image or icon.

Element	Description
Name	By default, this is the file name of the imported image or icon without the extension. You can change this name.
Description	An area for an optional description of the image or icon.
Details	The width and height of the imported image or icon in pixels. A preview of the image appears below this information.
File	The original location and file name of the imported image or icon.
Reimport	Replaces the image or icon with a different or updated file. Click this button to locate and reimport the file.

Using JavaScript in Custom Forms

You can include your own JavaScript files in custom forms and make calls to them from certain form controls. The SBM Server includes your scripts when the form is opened or the form control is clicked.

Important: JavaScripts are only executed in the SBM User Workspace and during form preview in SBM Composer. (By default, JavaScripts are not executed during form preview. To execute them during form preview, select the **Enable JavaScripts** check box in the Form Preview Dialog Box [page 634].)

Note: Perform this procedure if you want to use the JavaScript across forms. For a single form, use the HTMLJavaScript Widget [page 235].

Note: If you change a JavaScript file and then redeploy the process app, you might not see the change in the SBM User Workspace, because the JavaScript file is cached in the Web browser. To see the change, clear the cache from the Web browser. You can do this from the Web browser menu or by pressing Ctrl+F5.

To add a JavaScript file to an *application* [page 679]:

1. In App Explorer, open the application to which you want to add a JavaScript file.
2. Right-click the **JavaScripts** heading, and select **Add New JavaScript**.
3. Do one of the following:
   a. To create a new JavaScript file, type or paste the source code into the empty box.
   b. To import a JavaScript file, click **Import**, locate the file, and then click **Open**. The source code is displayed, and the path to the file is added to the **Source file** box.

You can use the JavaScript file when you select the following:

- **Custom form**: In the **JavaScripts** tab of the form Property Editor, you can select the file from the **Add** list. You can also import a JavaScript file and include it in the form by clicking **Import**. See JavaScripts Tab of the Form Property Editor [page 206] for details.
- **Button, HyperLink, or Image control**: In the **URL** field on the **General** tab of the control Property Editor, you can make a call to a JavaScript file. For buttons, enter the function to be called for the `onclick` event. For **HyperLink** and **Image** controls, create a JavaScript URL by typing `javascript:function()`.

JavaScript Editor

The JavaScript editor lets you create a new JavaScript file, or view and modify an imported JavaScript file.

Element	Description
Name	By default, this is the file name of the imported JavaScript file without the `.js` extension. You can change this name, if you prefer.
Description	An area for an optional description of the JavaScript file.
Source file	The original location and file name of the imported JavaScript file.
Import	Replaces the JavaScript file with a different or updated file. Click this button to locate and import the file.
Save to file	Creates a JavaScript file from the source code in the *Source code* box and saves it to the location you specify.  **Note:** Use this command only if you want to use the file outside of SBM Composer.
Source code	An area for typing, pasting, and editing JavaScript source code.
Script size	The size of the JavaScript file, in number of rows and bytes.

Customizing Styles

You can add and customize styles for the following elements in an *application* [page 679]:

- **Workflow annotations:** The style is applied to annotations you add to an application workflow.

   **Note:** You can override default styles for annotations on the **Appearance** tab of the Ribbon.

- **Application workflows:** The style is applied to the workflow editor.
- **Decisions:** The style is applied to decisions you add to an application workflow.
- **Forms**: The styles are applied to overall forms, detail controls, container controls, other controls, and widgets.
  The following table describes the default styles that are applied to the controls.

Control Type	Default Style
Detail Controls	Expander Style

Control Type	Default Style
Container Controls (Expander, GroupBox, Panel)	Expander Style
Container Controls (Tab)	Tab Style
Other Controls (Button, Combo Box, Edit Box, List Box)	Control Style
Other Controls (Image, Text)	Label Style
Other Controls (HyperLink)	HyperLink Style
Widgets	Control Style

Note: You can override default styles for individual forms or controls on the **Design** tab of the Ribbon.

- **States:** The style is applied to states in an application workflow.
- **Swimlanes:** The style is applied to swimlanes in an application workflow.

  Note: You can override default styles for individual swimlanes on the **Appearance** tab of the Ribbon.

  Important: When you add a new swimlane, the background color and line settings you specified in the style editor are overridden. To apply these settings to new swimlanes, select the swimlane, and then select the desired style on the **Style** area on the **Appearance** tab.
- **Transition styles:** The styles are applied to the Exception, Normal, Optional, and Preferred transition styles in the workflow editor and in the visual workflow your users see in the SBM User Workspace.

  Note: You can set transition styles for individual transitions on the **Design** tab in the workflow editor. For details, refer to About Transition Styles [page 322].

To add or customize a style:

1. In App Explorer, select the application for which you want to customize a style.
2. Click the **Styles** heading in App Explorer.
3. Do one of the following:
   - In the **Name** area in the left pane, select the style that you want to customize.
   - To add a new style, right-click in the **Name** area, and then select **Add New** and the style type.

     Note: Not all style types can be added.

   - To copy an existing style, right-click the style, and select **Duplicate**.

4. Complete the fields in the styles editor as needed. For details, refer to Styles Editor [page 260].

Styles Editor

The styles editor lets you customize and add default styles for the design elements listed in Customizing Styles [page 258].

Element	Description	Notes
Name	The name of the style.	Read-only for system-provided styles; editable for custom styles.
Description	An area for an optional description of the style.	Applies to all styles.
Foreground/ Text	Specifies the color for text in an annotation, transition, or form control, or for an annotation border. Click the down arrow to select a standard color; click **More Colors** to specify a custom RGB value.	Applies to all styles except form-related, state form, or transition form styles.
Background	Specifies the background color for an annotation or form. Click the down arrow to select a standard color; click **More Colors** to specify a custom RGB value.	Applies to all styles.
Line style, Line width	Specifies the style and line width (in pixels) for an annotation border or transition arrow.	Applies to all styles except form-related styles.
Font	Specifies the text styling. Click the browse button to modify the text font, size, and other attributes.	Applies to all styles except application workflow, form, state form, and transition form styles.
Image	Specifies an image to use as the background for an annotation or form style. Select an existing image or select **Add** to import an image.	Applies to annotation, expander, state form, tab, transition form, and swimlane styles.

Custom Transition Control Tutorials

The tutorials in this section demonstrate ways to use custom transition controls.

Note: For information about these controls, see About Custom Transition Controls [page 200].

- Tutorial: Adding a Custom Transition Button to a State Form [page 261]
- Tutorial: Repeating Transition Buttons at the Bottom of a Long State Form [page 262]
- Tutorial: Replacing Standard Transition Buttons with Hyperlinks on a Transition Form [page 265]

Tutorial: Adding a Custom Transition Button to a State Form

This tutorial demonstrates how to add a custom transition button to a state form, in a location that is different from the standard transition button bar. It also demonstrates how to use a variable for the button label, so if the transition name changes in the workflow, the button label is automatically updated.

Prerequisites:

Before you start this tutorial, create a new process app from the **Application Process App** template in the Create New Process App Dialog Box [page 626], and add the following states and transitions to the application workflow:

- In Dispatch ("active" state)
- In Progress ("active" state)
- Waiting for User ("active" state)
- Resolved ("active" state)
- Closed ("completed" state)
- Dispatch ("regular" transition from **New** state to **In Dispatch** state)
- Assign ("regular" transition from **In Dispatch** state to **In Progress** state)
- Request Info ("regular" transition from **In Progress** state to **Waiting for User** state)
- Provide Info ("regular" transition from **Waiting for User** state to **In Progress** state)
- Resolve ("regular" transition from **In Progress** state to **Resolved** state)
- Close ("regular" transition from **Resolved** state to **Closed** state)

Note: For instructions on creating a process app, see Chapter 6: Process App Tutorial [page 91].

Restriction: This tutorial only covers the steps involved in adding a custom transition button. It is not intended to cover all of the steps involved in creating a process app.

To add a custom transition button:

1. Click the **In Progress** state in the workflow editor.

2. Click the **Form** tab on the state Property Editor.
3. Click **New**. In the **Form Configuration** dialog box, select **Based on 'quick form' for: In Progress Regular State**.

   Note: For other custom form options, see Creating Custom Forms [page 198].

4. In the form editor, perform the following steps:
   a. Select the top section of the form, right-click, and then select **Add Row Above**.
   b. Drag a **Button** control from the **Form Palette** onto the new row.
   c. Click the **Align right** icon in the **Alignment** section on the **Design** tab of the Ribbon.
5. On the **Behavior** tab of the form Property Editor, perform the following steps:
   a. Select **Perform a transition**.
   b. Click **Add**.
   c. In the **Add Transition** dialog box, make sure the **In Progress** state is selected, select the **Request Info** transition, and then click **OK**.
   d. Select the **Show transition name** check box. The transition name will be displayed as the button label. If the name of the transition changes in the workflow, the button label will change automatically.
6. On the **General** tab of the form Property Editor, make sure the **Remove transition buttons matching custom transition controls** check box is selected. This option removes the **Request Info** transition button from the button bar.
7. To see the changes you made, click **Preview** in the Ribbon.

On the custom state form, the custom **Request Info** transition button is at the top right of the form, and the standard transition buttons (**Resolve**, **Update**, and **Delete**) are in the button bar at the top of the form.

Tutorial: Repeating Transition Buttons at the Bottom of a Long State Form

This tutorial demonstrates how to add custom transition buttons at the bottom of a long custom state form, so users do not have to scroll to the top of the form to click a standard transition button in the button bar. It also demonstrates how to use a single custom transition button for multiple transitions.

Prerequisites:

Before you start this tutorial, create a new process app from the **Application Process App** template in the Create New Process App Dialog Box [page 626], and add the following states and transitions to the application workflow:

- Assigned ("active" state)
- Work Started ("active" state)
- In Peer Review ("active" state)
- In QA ("active" state)
- Closed ("completed" state)
- Assign ("regular" transition)
- Start Work ("regular" transition)
- Review ("regular" transition)
- Test ("regular" transition)
- Close ("regular" transition)

Note: For instructions on creating a process app, see Chapter 6: Process App Tutorial [page 91].

Restriction: This tutorial only covers the steps involved in adding custom transition buttons. It is not intended to cover all of the steps involved in creating a process app.

To repeat standard transition buttons:

1. Create a custom state form. In the **Form Configuration** dialog box, select **Based on 'quick form' for: *application* Table**. (For instructions, see Creating Custom Forms [page 198].)
2. On the **Forms** tab of the workflow Property Editor, select the form you just created from the **Default state form** list.
3. Select the state form under the **Forms** heading in App Explorer. The state form opens in the form editor.
4. On the **General** tab of the form Property Editor, clear the **Remove transition buttons matching custom transition controls** check box. This step is necessary so buttons can be at both the top (button bar) and bottom of the form.
5. Drag a **Panel** control from the **Container Controls** section of the **Form Palette** to the bottom of the form, and drop it on the green row that appears. The Insert Dialog Box [page 636] opens.
6. In the **Insert Panel** dialog box, select **3** columns, **1** row, and **All autoSize**.

7. Drag a **Button** control from the **Other Controls** section of the **Form Palette** to each column in the new row.
8. Select the button in the first column. (This button will be used for all outgoing transitions you added to the application workflow.) Click the **Behavior** tab of the form Property Editor, and perform the following steps:
   a. Select **Perform a transition**.
   b. Select **Show transition name**. This option lets you add a single transition button for all transitions you added to the workflow. The button label changes automatically based on the current state. It also changes automatically if the transition name is changed in the application workflow.
9. Still on the **Behavior** tab, click **Add**. In the Add Transition Dialog Box [page 620], perform the following steps:
   a. Select the application workflow and the **New** state. Because there is only one outgoing transition from this state, the **Assign** transition is already selected. Click **OK**.
   b. Click **Add** again. Select the **Assigned** state. (The **Start Work** transition is selected.) Click **OK**.
   c. Click **Add** again. Select the **Work Started** state. (The **Review** transition is selected). Click **OK**.
   d. Click **Add** again. Select the **In Peer Review** state. (The **Test** transition is selected.) Click **OK**.
   e. Click **Add** again. Select the **In QA** state. (The **Close** transition is selected.) Click **OK**.

   Note: If you do not select **Show transition name**, you must type the button label in the **Display text** box on the **General** tab of the button Property Editor.
10. Select the button in the second column. (This button will be used to repeat the **Update** button, which is for a built-in transition and on the standard button bar). Perform the following steps:
    a. Click the **General** tab of the button Property Editor.
    b. Type `Update` in the **Display text** box.
    c. Click the **Behavior** tab of the button Property Editor.
    d. Select **Perform a transition**.
    e. Clear **Show transition name**. This ensures that the label you typed on the **General** tab will be displayed on the button instead of the transition name.
    f. Click **Add**. In the **Add Transition** dialog box, select the **[Any]** state and the **Update** transition. Click **OK**.
11. Repeat the previous step for the button in the third column, but type `Delete` in the **Display text** box, and in the **Add Transition** dialog box, select the **Delete** transition. (**Delete** is also a built-in transition.)

12. To see the changes you made, click **Preview** in the Ribbon.

On the custom state form, the transition buttons appear at both the top and bottom of the form. (The standard transition button bar is at the top of the form, and the custom transtition buttons are at the bottom of the form.)

Tutorial: Replacing Standard Transition Buttons with Hyperlinks on a Transition Form

This tutorial demonstrates how to replace the standard **OK** and **Cancel** buttons on a transition form with hyperlinks and change their labels.

Prerequisites:

Before you start this tutorial, create a basic process app from the **Application Process App** template in the Create New Process App Dialog Box [page 626], and add the following states and transitions to the application workflow.

- Assigned ("active" state)
- In Progress ("active" state)
- Closed ("completed" state)
- Assign ("regular" transition)
- Start Work ("regular" transition)
- Close ("regular" transition)

Note: For instructions on creating a process app, see Chapter 6: Process App Tutorial [page 91].

Restriction: This tutorial only covers the steps involved in adding custom transition buttons. It is not intended to cover all of the steps involved in creating a process app.

To replace standard transition buttons with hyperlinks and change their labels:

1. Click **Element** in the **New** section on the **Home** tab of the Ribbon, and then click **Transition Form**.
2. In the Form Configuration Dialog Box [page 633], select **Based on 'quick' form for *<application name>* Table**.
3. On the **Forms** tab of the workflow Property Editor, select the form you just created from the **Default transition form** list.
4. Select the custom transition form under the **Forms** heading in App Explorer. The form opens in the form editor.
5. Drag a **Panel** control from the **Container Controls** section of the **Form Palette** to the bottom of the form, and drop it on the green row that appears.
6. In the **Insert Panel** dialog box, select **2** columns, **1** row, and **All autoSize**.

7. On the **General** tab of the form Property Editor, make sure the **Remove custom transition buttons matching custom transition controls** check box is selected. This ensures that the hyperlinks you add will replace the standard buttons. If this check box is cleared, both hyperlinks and standard buttons will appear at the top of the form.
8. Drag a **HyperLink** control from the **Other Controls** section of the **Form Palette** to the first column in the new row.
   a. On the **Behavior** tab of the hyperlink Property Editor, select **Submit the form**.
   b. On the **General** tab of the hyperlink Property Editor, in the **Display text** box, type `Submit`).
9. Drag another **HyperLink** control to the second column of the new row.
   a. On the **Behavior** tab of the hyperlink Property Editor, select **Cancel the form**.
   b. On the **General** tab of the hyperlink Property Editor, in the **Display text** box, type `Abandon`).
10. To see the changes you made, click **Preview** in the Ribbon.

On the custom transition form, the standard **OK** and **Cancel** buttons are replaced with hyperlinks. **Submit** replaces the standard **OK** button label, and **Abandon** replaces the standard **Cancel** button label.

Chapter 13: Managing Workflows

This section contains the following topics about workflows in SBM Composer.

About Application Workflows

An *application workflow* [page 680] ensures the proper flow of primary items using a defined process that consists of fields, states, transitions, and decisions. Items must follow this application workflow from the time they are submitted to the time they are closed.

A sub-workflow is derived from an existing workflow. You could create a sub-workflow if you want to recreate the basic process defined in the existing workflow, but want to remove some steps or change the privileges on some transitions in the sub-workflow. Each workflow can contain multiple sub-workflows, and sub-workflows can contain their own sub-workflows.

Note: Items created by sub-workflows may be associated with each other, but do not have to be.

You design workflows in SBM Composer. In SBM System Administrator you assign the workflows to projects.

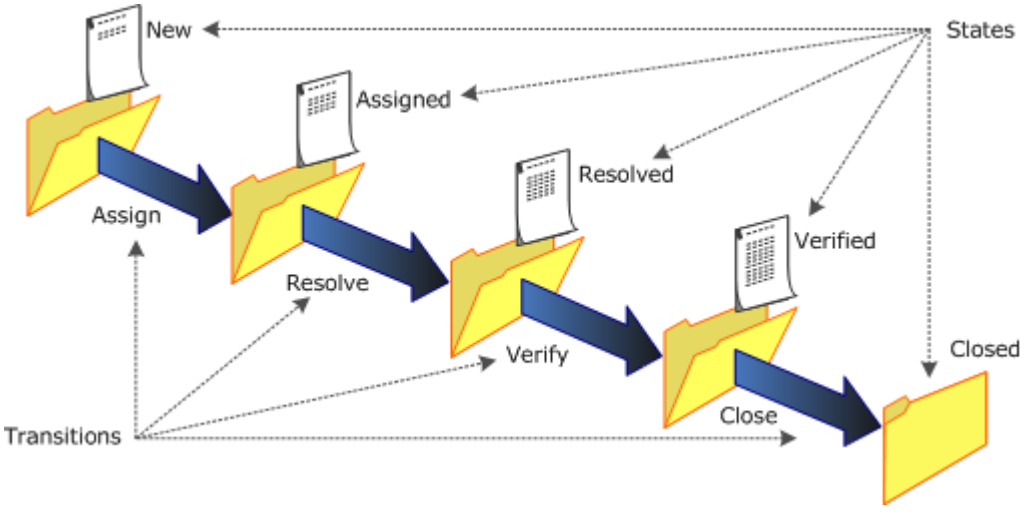
Elements of an Application Workflow

States: A state is a position in an application workflow where a *primary item* [page 691] resides. While an item resides in a given state, it has a *primary owner* [page 691] who is responsible for performing a specific task with the item before it can be transitioned to the next state. You can also set up an application workflow so that one or more users are secondarily responsible for items while they reside in a particular state.

Transitions: A transition activates the movement of a primary item from one state to another in the application workflow. A user who has *ownership* [page 690] of an item transitions the item to the next state in the workflow when the corresponding task is finished. Transitions can be customized in many ways. For example, you can customize transitions so that they apply only to certain item types, such as bugs and problem reports, or so that only members of particular groups can execute them.

Fields: Fields let users provide information about the primary items they are submitting or transitioning from state to state. The fields store the information in a *primary table* [page 692], located in a central database. They can be tailored to prompt users for pertinent information as a primary item moves through the application workflow.

The following figure shows an application workflow that defines the states and transitions an item must go through in order to reach a "closed" (completed) state.



Other Elements: You can add the following other elements to an application workflow.

- Decisions are required for *conditional routing* [page 682]. A decision has one incoming transition and two or more outgoing transitions. When an application

workflow reaches a decision, the rule for each outgoing transition is evaluated, and the outgoing transition with the first rule that evaluates to "true" is executed.

- Swimlanes are a way to visually group states in an application workflow, so others can quickly understand the workflow design. A swimlane typically represents either the activities performed by the same role or participant, or a distinct phase of the process defined by the workflow.
- Annotations are notes that you can add to a workflow or sub-workflow.
- The Relationships bar provides visibility into design elements, such as forms, that an application workflow uses. This lets you easily understand and explore the relationship between the states and transitions in the workflow and the related design elements.

Application Workflows Heading

The **Application Workflows** heading in App Explorer displays all the application workflows and sub-workflows that you defined for the selected application. You can click the **Workflow Design** filter at the bottom of App Explorer to see only the **Application Workflows** and **Orchestration Workflows** headings.

About the Relationships Bar

The Relationships bar provides visibility into design elements, such as forms, that an application workflow uses. This lets you easily understand and explore the relationship between the states and transitions in the workflow and the related design elements.

When you select a design element in the Relationships bar, the transitions or states that use the design element are highlighted in the application workflow. Likewise, when you select a transition or state in the application workflow, the referenced design elements are highlighted in the Relationships bar.

For example, suppose you want to see which forms are used for states and transitions in the workflow. You can click the **Forms** accordian in the Relationships bar, and an icon for each form appears. If you select a form in the Relationships bar, the states and transitions that use the form are highlighted in the application workflow.

You can double-click an icon in the Relationships bar to open the editor and Property Editor for the design element and make changes. Quick forms are an exception to this, because they cannot be modified.

Key Benefits

- Visual cues instantly demonstrate the relationship between an application workflow and the design elements it references.
- You can open the editor and Property Editor for a design element directly from the Relationships bar, so you do not have to navigate through App Explorer.

Related Topics

Using the Relationships Bar [page 269]

Using the Relationships Bar

This topic contains procedures for using the Relationships bar.

To show the Relationships bar:

1. Select the application workflow in App Explorer.
2. Click **Relationships** in the **View Mode** area of the **Design** tab of the Ribbon.
3. Click the **Relationships** tab at the bottom of the SBM Composer window.

To work with the Relationships bar:

1. Click the accordian for the relationships you want to see. The related design elements appear. For example, if you want to see the forms used by the workflow, click the **Forms** accordian.
2. To view the relationship between a design element and the workflow, do one of the following:
   - Click a design element in the Relationships bar. The states and transitions that use the design element are highlighted in the workflow.
   - Click a state or transition in the workflow. A green check box is displayed on the accordian or accordians with the related design elements. If one of these accordians is selected, the associated design elements are highlighted in the Relationships bar.
3. To open the editor and Property Editor so you can edit a design element, do one of the following:
   - Double-click the element in the Relationships bar.
   - Right-click the element in the Relationships bar and then select **Open *design element*** or **Open *design element* in New Tab**.

Restriction: You cannot open or edit quick forms.

To preview a custom form or orchestration workflow:

- Do one of the following:
  - Click **Relationships** in the Ribbon, click the **Forms** or **Orchestrations** accordian, and then hover over the design element in the Relationships bar.
  - Click **Presentation** in the Ribbon, and hover over the relationship hint on the state or transition in the workflow editor.

  A thumbnail view of the form or orchestration workflow opens.

Note: If multiple design elements of the same type (for example, orchestration workflows) are associated with a single state or transition, the thumbnail views are stacked so you can navigate among them.

To select the state or transition that uses a design element:

- Right-click the design element in the Relationships bar and then select **Select *design element* in Workflow**.

The state or transition that uses the design element is selected in the workflow.

To hide the Relationships bar:

- Click **Presentation** or **Properties** in the **View Mode** area on the **Design** tab of the Ribbon.

About Conditional Routing

Conditional routing lets designers create application workflows that automatically route an item to a particular state based on visually designed rules that evaluate item data. They are used in decisions to determine which outgoing transition is executed.

One or more incoming transitions lead from one or more source states to a decision, and two or more outgoing transitions lead from the decision to their respective target states. The rules are evaluated in the order you specify in the Rules Tab of the Decision Property Editor [page 329]. If a rule evaluates to "true," the associated transition is taken.

The last transition in the Property Editor list must be mapped to an "Otherwise" rule. This transition is taken if the last rule evaluates to "true" or if no rule evaluates to "true."

Note: For step-by-step instructions for using conditional routing, see Tutorial: Using Conditional Routing [page 274]. For examples of how conditional routing can be used, see Use Cases: Conditional Routing [page 271]. For ways to implement conditional routing most effectively, see Best Practices: Conditional Routing [page 272].

Key Benefits

- Conditional routing is an intuitive way to route items based on certain conditions. You no longer need to use actions and AppScripts to accomplish this.
- The decision in the workflow editor lets you see the routing options at a glance.
- A rules editor lets you build rule expressions graphically.
- The Property Editor for the decision displays the rule associated with each outgoing transition, and optionally shows detailed information about the rule and its associated transition. You can open the rules editor from this Property Editor to add and edit rules.
- Application variables make it easy to create and maintain rules.

Use Cases: Conditional Routing

The following use cases illustrate ways in which conditional routing can be used.

Expense Reporting

Employees use an application to submit expense reports. A decision named **Amount?** has an incoming **Evaluate** transition and three outgoing transitions: **Send to A/P**, **Get Mgr Approval**, and **Get VP Approval**. The workflow evaluates the rules that are associated with the **Amount?** decision.

- If the reimbursement amount is less than or equal to $100.00, no approval is required, so the **Send to A/P** transition is executed.
- If the reimbursement amount is more than $100.00 but less than or equal to $1000.00, a manager's approval is required, so the **Get Mgr Approval** transition is executed.

- If neither of the preceding rules evaluate to "true," then the reimbursement amount is more than $1000.00. In this case, a vice president's approval is required, so the **Get VP Approval** transition is executed.

New Hire Processing

Human resources uses a new hire application to request new computers. A decision named **Employee Type?** has an incoming **Evaluate** transition and two outgoing transitions: **Send to Facilities** and **Send to IT**. The workflow evaluates the rules that are associated with the **Employee Type?** decision.

- If the employment type is "Local," office space is needed for the employee, so the **Send to Facilities** transition is executed. After the office space is allocated, the **Send to IT** transition is executed so a computer can be ordered for the employee.
- If the employee type is not "Local," the employee must be "Remote." In this case a computer is needed, but not office space, so the **Send to IT** transition is executed.

Incident Management

When users submit into an incident management application, they are presented with a form where they enter field data and then select whether to post a new problem item to a problems application, or link it to an existing problem item.

A decision named **Post Type?** has the following transitions:

- An incoming **Post Problem** transition from the **Any** state
- An outgoing **Create** transition to the **Any** state
- An outgoing **Link** transition to the **Any** state

The workflow evaluates the rules associated with the **Post Type?** decision. If the user selects **New** from the drop-down list on the form, the **Create** transition is executed, and presents the user with the submit form for the problems application. If the user selects **Link**, the **Link** transition is executed, which presents a form that lets the user select the existing problem, and then invokes a quick transition that runs an orchestration workflow that links the two items.

The **Post Type?** decision has transitions from and to the **Any** state, because posting and linking can be done from every state in the workflow. (The **Any** state allows you to have one or more transition buttons in every state form in the workflow. For more information, see System-Provided States [page 294].)

Best Practices: Conditional Routing

This topic lists ways to implement conditional routing most effectively.

- Cover the entire range of values in your rules. For example, if an item should be routed based on a numeric value, and the user specifies a value that is not covered by a rule, it must take the transition associated with the **Otherwise** rule. This could result in unintended routing.
- Make rules mutually exclusive. For example, if an item in a defect management application should be routed based on severity, and the severity level can be 1 through 5, make sure each level is covered by only one rule. Do not assign 1 and 2 to one rule, and assign 2 and 3 to another rule.

- Use application variables. They offer the following advantages:
  - You can change the value of an application variable and all of the rules that use the application variable are automatically updated.
  - You can prevent gaps that can cause unintended routing. For example, in an expense reimbursement application, one rule covers reimbursements that are between $0.00 and $100.00, inclusive. Another rule covers reimbursements greater than $100.00 but less than $1000.00.

    If you change the upper limit of the first rule to $90.00, but forget to change the lower limit of the second rule accordingly, then a reimbursement of $95.00 must take the transition associated with the **Otherwise** rule. You can avoid this unintended routing if you instead assign 100 to an application variable, and use that variable in both rules.
- To prevent infinite loops:
  - Do not have two decisions, where each decision has an outgoing transition to the other decision.
  - Do not put a transition action on an outgoing transition from a decision that causes the incoming transition to that decision to be executed.
- Do not restrict the "Otherwise" transition from a decision step by role, group, or item type.
- A decision must have one incoming transition from the **Any** state before it can have an outgoing transition back to the **Any** state.

Tutorial: Using Conditional Routing

This tutorial shows you how to route an item in an application workflow based on the amount of an expense reimbursement request.

- If the amount is less than $100.00, no approval is required, so the item is routed directly to the **Accounts Payable** and **Processing** states, where the reimbursement is processed. After the reimbursement is processed, the item is closed.
- If the amount is between $100.00 and $1000.00, inclusive, a manager's approval is required, so the item is routed to the **Manager** state. The manager approves the request and transitions the item to the **Accounts Payable** state.
- If the amount is more than $1000.00, a vice president's approval is required, so the item is routed to the **Vice President** state. The vice president approves the request and transitions the item to the **Accounts Payable** state.

Note: This tutorial does not demonstrate all ways to configure decisions and creating rules and application variables. For complete information about these tasks, see Working with Decisions [page 327], Creating a Rule [page 340], and Creating an Application Variable [page 354].

Note: For basic instructions on creating a process app, see Chapter 6: Process App Tutorial [page 91].

To use conditional routing:

1. Create a process app from the Create New Process App Dialog Box [page 626] and name it `Expense Reimbursement`.
2. In the application workflow, add states and a decision as follows:
   a. Drag a **Decision** from the **Workflow Palette** to the workflow editor, and name it **Amount?**.
   b. Drag four "active" states from the **Workflow Palette** to the workflow editor and give them the following names:
      - **Accounts Payable**
      - **Manager**
      - **Vice President**
      - **Processing**
   c. Drag a "completed" state to the workflow and name it **Closed**.
3. Rename the **Branch** transition from the **Amount?** decision to `Send to A/P`, and connect it to the **Accounts Payable** state.
4. Rename the **Otherwise** transition from the **Amount?** decision to **Send to VP**, and connect it to the **Vice President** state.

5. Add "regular" transitions from the **Workflow Palette** as shown in the following table.

   Note: Quick transitions do not have forms associated with them and do not require user input. You can drag **Quick** from the **Transitions** section of the **Workflow Palette**, or make a regular transition a quick transition by selecting the **Quick transition (do not show a form)** check box on the **Options** tab or **Form** tab of the transition Property Editor. Outgoing "regular" transitions from a decision are configured as quick transitions by default.

Source	Target	Name	Quick Transition?
New	Amount?	Evaluate	Yes
Amount?	Manager	Send to Mgr	Yes (default)
Manager	Accounts Payable	Send to A/P	No
Vice President	Accounts Payable	Send to A/P	No
Accounts Payable	Processing	Process	No
Processing	Closed	Close	No

6. Add a *Numeric* field to the **Expense Reimbursement** table:
   a. Name the field **Amount**.
   b. On the **Options** tab of the field Property Editor, select **Fixed precision**, specify two digits to be displayed after the decimal point, and specify `$` as the prefix.
7. Select the **Amount?** decision in the workflow editor, and then select the **Rules** tab of the decision Property Editor.
8. Select the **Send to A/P** transition, and then click the arrow in the **Rule** column.
9. Select **(New rule)**. The rule editor opens.
10. On the **General** tab of the rule Property Editor, type `A/P Direct` in the **Name** box.
11. Drag an *Amount* field from the **Rule Palette** onto the **Rule: A/P Direct** block in the rule editor.
    a. Select the **>** conditional operator.
    b. Make sure **Value** is selected in the drop-down list, and type `0` in the box.
12. Drag an **AND** logical operator from the **Rule Palette** onto the **Amount** field.
13. Drag another *Amount* field onto the bottom line of the **AND** operator.

    Note: You can optionally skip the previous step and instead drag the other *Amount* field onto the existing *Amount* block. This automatically adds the **AND** logical operator to the expression.

a. Select the **<=** comparison operator.

b. Make sure **Value** is selected in the drop-down list, type `100.00` in the box, and then click **Convert to variable**.

Note: Designers can use the same application variable in multiple rules. They can also change the default value of an application variable, and all rules that reference that field will use the new value. For more information, see Introducing Application Variables [page 353].

14. Click **Edit application variables**.
15. In the application variable editor, rename **Amount Variable** to **Threshold**.
16. Click the **Navigate Backward** button  in the quick access toolbar at the top of the SBM Composer window two times.
17. On the **Rules** tab of the **Amount?** decision Property Editor, select the **Send to Mgr** transition, and then click the arrow in the **Rule** column.
18. Select **(New rule)**.
19. On the **General** tab of the rule Property Editor, type `Manager` in the **Name** box.
20. Drag an *Amount* field from the **Rule Palette** to the **Rule: Manager** block in the rule editor.
    a. Select the **>** comparison operator.
    b. Click **Variable** in the drop-down list at the right of the expression, and then select **Threshold**.
21. Drag an **AND** logical operator from the **Rule Palette** onto the **Amount** field.
22. Drag an *Amount* field onto the bottom line of the **AND** operator.
    a. Select the **<=** comparison operator.
    b. Make sure **Value** is selected in the drop-down list, and type `1000.00` in the box.
23. Click the **Navigate Backward** button  .
24. On the **Rules** tab of the **Amount?** decision Property Editor, select the **Send to VP** transition, and click the arrow in the **Rule** column.
25. Select **[Otherwise]**. One "otherwise" rule must be mapped to a transition. This transition is executed if the rule mapped to it evaluates to "true," or if no rule evaluates to "true." In this case, any amount over $1000.00 will use this rule and the **Send to VP** transition will be executed.
26. Validate, publish, and deploy the process app.

Creating a Workflow

You can create a new workflow in an *application* [page 679].

When you create a new *application workflow* [page 680], the only elements that exist in the new workflow are system fields; the **Submit**, **New**, **Email**, and **Deleted** states; the **Any** item; and the **Submit**, **Update**, and **Delete** transitions. You add states, transitions,

decisions, and other objects from the **Workflow Palette** at the right side of the workflow editor. Any sub-workflows derived from the new workflow inherit the states, transitions, decisions, swimlanes, and annotations you added.

To create a workflow:

1. Do one of the following:
   - In App Explorer, right-click the **Application Workflows** heading, and select **Add New Workflow**.
   - In the **New** area of the **Home** tab of the Ribbon, click **Element** and then **Workflow**.
2. Do one of the following:
   - In App Explorer, right-click the workflow you just added, select **Rename**, type the new name (32 characters or less), and press the Enter key.
   - On the **General** tab of the Property Editor, type the new name in the **Name** field, and then press the Tab or Enter key.

Creating a Sub-workflow

You can add a new sub-workflow to an existing workflow. The new sub-workflow appears as a child of its parent workflow in App Explorer.

To create a sub-workflow:

1. In App Explorer, right-click the workflow where you want to add the sub-workflow and select **Add New Sub-workflow**.
2. Do one of the following:
   - In the App Explorer, right-click the sub-workflow you just added, select **Rename**, type the new name, and press Enter.
   - With the sub-workflow selected, on the **General** tab of the Property Editor, type the new name in the **Name** field, and press the Tab or Enter key.

Opening a Workflow

You can open an individual workflow or view the list of existing workflows. The workflow list shows version number, creation date, and update date.

To view a list of all workflows:

- In App Explorer, do one of the following:
  - Expand the **Application Workflows** heading.
  - To open the list in a new tab, right-click the **Applications Workflows** heading, and select **Open in New Tab**.

To open a workflow or a sub-workflow:

- In App Explorer, do one of the following:
  - Click the workflow or sub-workflow.
  - Right-click the workflow or sub-workflow, and select **Open in New Tab**.
  - Click the **Application Workflows** heading.

    In the workflow editor, double-click the workflow or sub-workflow that you want to open.

To open the parent workflow of the open sub-workflow:

- With the sub-workflow open in the workflow editor, right-click on the background and select **Open Parent Workflow**.

Duplicating a Workflow

You can duplicate any workflow or sub-workflow. The duplicate is added to the **Application Workflows** heading. A duplicate sub-workflow is added as a child of the same parent workflow.

Duplicated workflows inherit the states, transitions, and decisions in the originating workflow.

Note:

- When you duplicate a workflow that contains sub-workflows, only the parent workflow is duplicated.
- The default name for a duplicated workflow is *workflow name workflow name* 2. The default name for a duplicated sub-workflow is *workflow name workflow name* Sub-workflow 2. If you duplicate the workflow or sub-workflow, the name is repeated, 2 is incremented to 3, and so on. For example, if you duplicate the workflow **Test**, the new duplicate is named **Test Test 2**. If you duplicate **Test** or **Test Test 2**, the new duplicate is named **Test Test Test 3**.

To duplicate a workflow:

- In App Explorer, right-click the workflow or sub-workflow, and select **Duplicate**.

  The duplicated workflow is added to App Explorer.

Deleting a Workflow

You can delete any workflow or sub-workflow from an *application* [page 679].

Note: You cannot delete a workflow that contains sub-workflows. You must first delete its sub-workflows.

To delete a workflow:

- In App Explorer, right-click the workflow or sub-workflow, and select **Delete**.

Changing the Order of Workflows in App Explorer

In App Explorer, you can change the order of workflows and sub-workflows by moving them up and down the tree.

Note: You cannot move a sub-workflow from one workflow to another.

To change the order of workflows in App Explorer:

1. In App Explorer, right-click the workflow or sub-workflow that you want to move.
2. Select **Move Up** or **Move Down**.

Modifying Application Workflow Properties

Modifying a workflow lets you configure and customize your *application* [page 679]. For example, a sub-workflow could include a transition that was inherited from a parent workflow but is not needed in the sub-workflow. You can edit the sub-workflow and disable the transition.

When an application workflow or sub-workflow is open in the workflow editor, you can modify its properties using the tabs of the Property Editor, which are described in the following sections.

Tip: If you click **Properties** in the **View Mode** area on the **Design** tab of the Ribbon, icons are displayed on states and transitions that indicate the properties that are set. If you click an icon, the applicable tab opens in the Property Editor.

- General Properties of an Application Workflow [page 279]
- Forms Properties of an Application Workflow [page 280]
- Field Privileges of an Application Workflow [page 281]
- Field Overrides for an Application Workflow [page 281]
- Dependencies for an Application Workflow [page 287]

General Properties of an Application Workflow

When an *application workflow* [page 680] or sub-workflow is open in the workflow editor, you can modify its general properties.

To modify the general properties of a workflow:

1. Open the workflow or sub-workflow.
2. In the workflow Property Editor, click the **General** tab. See General Tab of the Application Workflow Property Editor [page 280] for details.

General Tab of the Application Workflow Property Editor

Use this tab to modify the general properties of the selected *application workflow* [page 680] or sub-workflow.

Element	Description
Name	*(Optional)* Modify the name of the workflow or sub-workflow.
Description	*(Optional)* Modify the description of the workflow or sub-workflow.

Forms Properties of an Application Workflow

When an *application workflow* [page 680] or sub-workflow is open in the workflow editor, you can modify its default forms.

To modify the default forms for a workflow:

1. Open the workflow or sub-workflow.
2. In the workflow Property Editor, click the **Forms** tab. See Forms Tab of the Application Workflow Property Editor [page 280] for details.

Forms Tab of the Application Workflow Property Editor

When an *application workflow* [page 680] or sub-workflow is selected in the editor, you use this tab to manage its default state and transition forms.

Element	Description
Default state form	Select the form to be associated with new states added to this workflow. (You can change the form associated with an existing state at any time.)
Default *transition form* [page 700]	Select the form to be associated with new transitions added to this workflow. (You can change the form associated with an existing transition at any time.)
New	You can base a new custom form on the *quick form* [page 693] for the workflow or on an existing custom form, or you can start with an empty form. (You can also create a state or transition form by right-clicking the **Forms** heading in App Explorer.)
Preview	View a mockup of the corresponding form as it will appear to a user. You can use the controls at the top to view the form as it will be seen in a different workflow, state, or transition and by a user in a different *role* [page 695]. (The controls in the preview area have no effect.)
Edit	If a custom form is selected for the corresponding default form, click to open the form in the form editor.

Field Privileges of an Application Workflow

When an *application workflow* [page 680] or sub-workflow is open in the workflow editor, you can modify its field privileges.

To modify the field privileges of a workflow:

1. Open the workflow or sub-workflow.
2. In the workflow Property Editor, click the **Field Privileges** tab. See Field Privileges Tab of the Application Workflow, State, and Transition Property Editor [page 281] for details.

Field Privileges Tab of the Application Workflow, State, and Transition Property Editor

Use this tab to modify the field privileges of an *application workflow* [page 680], *state* [page 698], or *transition* [page 700].

Element	Description
Override field privileges	Select this option if you want to make changes to field privileges.
List of fields	The fields in this list are divided into sections, such as **Standard** and **Advanced**. • To change the field ordering in a section, drag and drop the fields. • To change the field privileges in each section, drag and drop fields between sections.
Fields in the *section-name* Section	When you select a field in the **Override field privileges** list, this section displays the permissions for that field. For example, it could indicate that fields in the **Advanced** section can be changed by the Administrator and Manager roles, can be read by the Developer *role* [page 695], and are invisible to the User role.

Tip: To sort the fields in a privilege section, right-click any of the fields, point to **Sort**, and then select and then select **Ascending** or **Descending.**

Field Overrides for an Application Workflow

When an *application workflow* [page 680] or sub-workflow is open in the workflow editor, you can override its field properties.

Note: For information about *overrides* [page 690], and for examples of overriding field properties, see Chapter 16: About Inheritance and Overrides [page 355].

To override the field properties for a workflow:

1. Open the workflow or sub-workflow.

2. In the workflow Property Editor, click the **Field Overrides** tab. See Field Overrides Tab of the Application Workflow and Transition Property Editor [page 282] for details.

Field Overrides Tab of the Application Workflow and Transition Property Editor

Use this tab to override the properties of the fields for a workflow or transition (except a **Delete** transition).

Note: For information about *overrides* [page 690], and for examples of overriding fields, see Chapter 16: About Inheritance and Overrides [page 355].

Element	Description
List of fields	Select a field whose properties you want to override. (Fields with overrides are displayed in boldface type.)  **Note:** You can sort the fields by clicking the **Arrange by** button and selecting a sort option.  **Tip:** To sort the list into two sections, showing fields that are overridden and those that are not, select the **Overridden** sort option.
Override field properties for *field-name*	Select this option for each field type whose properties you want to modify.

Attributes

Element	Description
Required	Makes the field a required attribute.
Read only	Prevents the field from being modified.

Element	Description
Allow mass update	Lets users simultaneously transition, update, or delete multiple primary items and to simultaneously update or delete multiple auxiliary items. For example, if several primary items have been "Deferred" from the system, you can activate all the items at the same time by performing a mass "Restart" transition. This activates all the items at the same time and places them in the same state. This option provides the flexibility of determining the specific fields that can be modified when users perform a *mass update* [page 689]. For example, if the **Additional Notes** field is selected for mass update, users can enter a value in the field before completing a mass update. This value overwrites any previous values stored in the field for items that are mass updated. If a value is not provided for the field during the mass update, the original values remain unchanged.
Default value	• For a text box field, type a default value. • For a list field, select whether the list is active, inactive, or automatically updated.

Style options

Element	Description
Allow searching	Select this option to enable "Value Find" on submit, transition, and update forms for the field. Value Find lets users enter search criteria, such as an entire word, a few letters, or an asterisk, and then click a **Find** button or press the Enter key to perform the search. Results appear in a drop-down list on the form, letting the user select one of the found values for the field.
Single drop-down list	Select this option to let the user select a value from a populated drop-down list.
Default value	• For a text box field, type a default value. • For a list field, select a state from the list.

Value options (transitions only)

Element	Description
Leave unchanged	Select this option to leave the value in the field unchanged during a transition.
Clear	Select this option to clear the value in the field during a transition.

Element	Description
Set to default	Select this option to set a default value for a *transition field* [page 700]. For *Text* fields with the **Journal** display option enabled, this setting applies only to appended text. Selecting this option displays the **Default value** field. See Date/Time Values [page 166] for a description of your options for populating this field. **CAUTION:** The "(Auto)" string is not supported as a default value for *Date/Time* fields.
Set to calculation	This option is only available for *Date/Time* and *Numeric* fields. This option allows you to calculate a value for these *field types* [page 686]. For example, you can calculate an amount owed based on a rate and the actual time to resolve. The calculation can be made read-only and placed in the **Hidden Fields** section. Selecting this option displays several additional controls that you use to set up date calculations. See Calculating Values for Date/Time and Numeric Fields [page 284] for a description of your options for populating this field.

Calculating Values for Date/Time and Numeric Fields

Field calculations for transitions provide a way to collect metrics using *Date/Time* and *Numeric* fields. For example, you could calculate how long Help Desk calls wait before a Support representative begins working the issue, or calculate interest values based on the current date.

The **Set to calculation** option appears when you select a *Date/Time* or *Numeric* field on the **Field Overrides** tab of the transition Property Editor.

To calculate values for *Date/Time* and *Numeric* fields:

1. In App Explorer, select the workflow containing the transition.
2. In the workflow editor, select the transition for which you want a *Date/Time* or *Numeric* field to perform a calculation.
3. In the transition Property Editor, select the **Field Overrides** tab, and select the **Override field properties for '*field name*'** option.
4. Optionally, click **Arranged by**, and select a different sort order for the list of fields.
5. In the field list, select the *Date/Time* or *Numeric* field for which you want to define the calculation to be performed.
6. In the **Value options** area, select the **Set to calculation** option.
7. Do one of the following:
   - Select **Field** from the **Operand 1** menu, and then select a field from the drop-down list or a keyword from the date/time tool. You cannot select the field being

edited or a field that could cause a recursive calculation. For a *Date/Time* field, see Date/Time Values [page 166] for a description of your options for populating this field.

CAUTION: The "(Auto)" string is not supported as a default value for *Date/Time* fields.

- Select **Value** from the **Operand 1** menu, and then enter a constant value in the adjacent text box.

Tip: Calculations do not work for system date fields (Submit Date/Time, Close Date/Time, Last State Change Date, Last Modified Date) in a transition calculation for which the date will be calculated. Instead, use the "Now" keyword, which gives you the same calculation.

8. Select an operator from the **Operator** drop-down list.

   The list contains only valid operators for this field and the current operand. If you change the operand, the operator list is updated with the appropriate choices. If you select an operator that is not valid with the operand, the operator changes, usually to **None**. (The **None** operator is useful if you want to transfer the value of another field to this field during the transition.)

9. If you select an operator other than **None**, you can specify a second operand from the **Operand 2** menu. This list offers the same options and constraints as the first.

10. From the **Perform calculation** list, specify whether the calculation should be performed before the *transition form* [page 700] opens, after it is submitted, or both.

11. From the **Empty operand fields** list, specify how the transition should treat empty operands:

    - **Are Invalid**: Select this option to require users to provide values for fields used as operands. The transition cannot be completed if values are not provided.
    - **Skip Calculation**: Select this option to let users complete the transition without providing values for fields used as operands. The calculation is simply skipped if values are not provided.
    - **Treat as Zero**: Select this option to perform the calculation using zeros in place of any unprovided values for fields used as operands.

12. For *Numeric* fields and *Date/Time* fields with the **Elapsed time** option enabled, select the **Add calculation to current value** check box to add the result of the calculation to the current value of the field. Use this option to increment the current value or to calculate the total of the calculation and the current value.

About Operand Fields and Operator Selection Lists

The following table lists the valid choices for the operand fields and operator selection lists for a *Numeric* or *Date/Time* field using the calculation feature. The valid choices are based on the type of *transition field* [page 700] being edited.

Field Type	1st Operand Constant	1st Operand Field Types	Operators	2nd Operand Constant	2nd Operand Field Types
Elapsed Time	Elapsed Time	Elapsed Time	None, +, –	Elapsed Time	Elapsed Time
Elapsed Time	Elapsed Time	Elapsed Time	*, Truncating /, Rounding /	Float	Numeric Int, Single Select, Summation
Elapsed Time	Elapsed Time	Elapsed Time	Truncating *, Rounding *, Truncating /, Rounding /	Float	Numeric Float
Elapsed Time	Date/Time	Date/Time	–	Date/Time	Date/Time, Date
Elapsed Time	*(not applicable)*	Date	–	Date/Time	Date/Time, Date
Date/ Time	Date/Time	Date/Time	None, +, –	Elapsed Time	Elapsed Time
Date/ Time	*(not applicable)*	Date	None, +	Elapsed Time	Elapsed Time, Time
Date/ Time	*(not applicable)*	Date	–	Elapsed Time	Elapsed Time
Date	Date	Date/Time, Date	None, Truncating +, Rounding +, Truncating –, Rounding –	Elapsed Time	Elapsed Time
Time	Time	Time	None, +, –	Elapsed Time	Elapsed Time
Time	*(not applicable)*	Date/Time	None (assigns time portion of date/time)	*(not applicable)*	*(not applicable)*
Numeric Int	Int	Numeric Int, Single Select, Summation	None, +, –, *, Truncating /, (integer math), Rounding /	Int	Numeric Int, Single Select, Summation

Field Type	1st Operand Constant	1st Operand Field Types	Operators	2nd Operand Constant	2nd Operand Field Types
Numeric Int	Int	Numeric Int, Single Select, Summation	Truncating +, Truncating –, Truncating *, Truncating /, Rounding +, Rounding –, Rounding *, Rounding /	Float	Numeric Float
Numeric Int	Float	Numeric Float	Truncating +, Truncating –, Truncating *, Truncating /, Rounding +, Rounding –, Rounding *, Rounding /	Int or Float	Numeric Int/ Float, Single Select, Summation
Numeric Float	Int or Float	Numeric Int/Float, Single Select, Summation	None, +, –, *, /	Int or Float	Numeric Int/ Float, Single Select, Summation
Numeric Float	Int or Float	Numeric Int/Float, Single Select, Summation	*	*(not applicable)*	Elapsed Time (converted to hours)

Dependencies for an Application Workflow

When an *application workflow* [page 680] or sub-workflow is open in the workflow editor, you can set up field dependencies.

To modify the field dependencies for a workflow:

1. Open the workflow or sub-workflow.
2. In the workflow Property Editor, click the **Dependencies** tab. See Dependencies Tab of the Application Workflow Property Editor [page 287] for details.

Dependencies Tab of the Application Workflow Property Editor

Use this tab to define dependencies for the workflow or sub-workflow in the editor.

Element	Description
Select independent field	Select the field for which you want to define a dependency.

Element	Description
Restrict dependent field... To these values	When a user selects a value in the specified independent field, the dependent field you specify here is automatically limited to the values you specify here. In the **To these values** column, you can use Ctrl+click, Shift+click, and Ctrl+A to highlight multiple values, then click any one of the highlighted check boxes to select all the highlighted values. Click **Check All** or **Uncheck All** to select or clear all values.
Change value to	Define a specific dependent field default value for each independent field value.

Application Workflow Editor

The workflow editor is where you design and modify application workflows. In the workflow editor you can:

- Add and modify objects such as states, transitions, decisions, swimlanes, and annotations.
- Add and modify design elements, such as forms and actions.
- Override inherited field properties.

You can use the following features in the workflow editor to help you work when you are designing your workflows:

- Zoom in to get a close-up view of your workflow, or zoom out to see more of it. The zoom range is between 10% and 500%.
- Resize the workflow to fit the workflow editor.
- Use the blue rectangle in the zoom preview (in the bottom right corner) to move around the workflow.
- Use the Relationships bar to explore the relationship between design elements and the workflow. For more information, see About the Relationships Bar [page 269].

To select a preset zoom level:

- In the **Zoom** area on the **Home** tab of the Ribbon, do one of the following:
  - Click **Zoom**, and select a zoom level.
  - Click **100%**.

To zoom in and out:

- Do one of the following:
  - Below the zoom preview (in the bottom right corner), click the **+** and **–** buttons on the slider, or move the slider to the right and left.

    The current zoom level is displayed to the left of the slider.

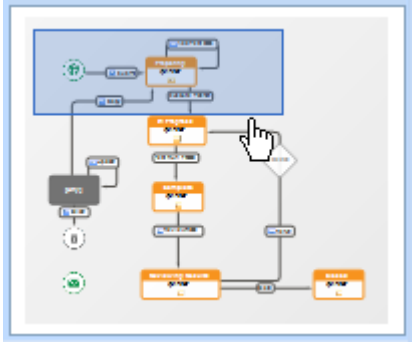
- With the pointer over the workflow editor, press the Ctrl key while you move the mouse backward and forward.

To fit the workflow to the window:

- In the **Zoom** area on the **Home** tab of the Ribbon, click **Zoom** and then select **Fit to Window**.

To move around in the workflow editor:

- Do one of the following:
  - In the zoom preview (in the bottom right corner), drag the blue rectangle to the section of the workflow that you want to view in the workflow editor window.



    As you drag the rectangle around in the thumbnail view, the section of the workflow that is visible in the editor changes. This is useful when you have zoomed in to a large workflow and want to move to another section without having to zoom out again. The blue rectangle indicates the portion of the workflow that is currently visible in the workflow editor—when you change the zoom setting, the rectangle resizes.

  - In the workflow editor, with the pointer over the background (white space), press and hold the right mouse button while you drag the workflow graphic within the editor pane.

Displaying Workflow Design Elements

You can toggle the display of certain workflow components. See Design Tab of the Ribbon [page 58] for details.

Arranging States and Transitions

In the workflow editor, you can choose the following layout options for states and transitions:

- **Auto Arrange**: Automatically arranges the states and transitions.
- **Re-Inherit From Parent**: Update the layout to that of the parent workflow.

To arrange states and transitions:

Do one of the following:

- In the **Layout** area on the **Design** tab of the Ribbon, click **Auto Arrange** or **Re-inherit From Parent**.

- Right-click the workflow background, and select **Auto Arrange**.
- Right-click the sub-workflow background, and select **Auto Arrange** or **Re-Inherit From Parent**.

Navigating through States and Transitions

In preview mode, you can navigate through the forms in your workflow. This lets you experience the user flow without having to deploy the process app and test it in the SBM User Workspace.

To navigate through states and transitions:

1. Select a state or a transition in the workflow.

   Restriction: You cannot preview the form for the **Submit**, **Email**, and **Any** states, and the **Delete** state and transition.

2. Do one of the following:
   - Right-click in the workflow, and then select **Preview State Form** or **Preview Transition Form**.
   - Click **Preview** on the **Form** tab of the state or transition Property Editor.
3. Click a transition control on the form that opens. The state or transition form associated with the state or transition opens. The state or transition is highlighted when you reach it in the workflow.

   Note: By default, transition controls are buttons that are automatically placed at the top of the form. You can replace these buttons with transition controls (buttons, hyperlinks, or images) that can be placed anywhere on the form. For more information, see Behavior Tab of the Control Property Editor [page 208].

4. Continue to click transition controls on forms as you navigate through the workflow.

   Note: This feature is not available when you select a form in App Explorer and then click **Preview** on the **Design** tab of the Ribbon. It is only available from the workflow, as described in this procedure.

Workflow Palette

In the workflow editor, you drag items from the **Workflow Palette** and arrange them in a process flow. You then use Property Editors to configure the items. This topic describes each section of the **Workflow Palette**.

Common Items

The following items are included in the **Common Items** section.

Item	Notes
Swimlane	See About Swimlanes [page 331] for information.
State	This item is identical to **Active** in the **States** section. See About States [page 294] for information.

Item	Notes
Decision	Two outgoing transitions are automatically added to the decision: **Otherwise** and **Branch**. Both transitions are required. See About Decisions [page 327] for more information.
Transition	This item is identical to **Regular** in the **Transitions** section. See About Transitions [page 304] for information.
Annotation	See Working with Annotations [page 335] for information.

States

The following items are included in the **States** section. After you drag a state to the workflow editor, you can rename the state and change its properties. For more information about states, see About States [page 294].

Item	Notes
Active	A generic active state.
Completed	A preconfigured inactive state.
Rejected	A preconfigured inactive state.
Pending	A preconfigured inactive state.

Transitions

The following items, which represent transition types, are included in the **Transitions** section. After you drag a transition to the workflow editor, you can rename the transition and change its properties. For more information, see About Transition Types [page 305].

- Regular
- Quick
- Post
- Subtask
- Publish
- Copy
- Update
- Delete
- External Post

Ownership of Items

SBM provides two types of *ownership* [page 690]: primary ownership and secondary ownership. Primary ownership means that a single user is responsible for an item while it resides in a particular state. Secondary ownership lets multiple users own an item in a state. With appropriate privileges, secondary owners can view, update, and transition items as needed.

To provide accountability for primary items, an owner is designated for each state in the workflow. Each state may have a *primary owner* [page 691], a primary owner and secondary owners, or secondary owners.

Considerations For Setting Primary Ownership Of Items

You define primary ownership for each state by selecting a single *User* field, such as *Manager*, *Engineer*, or *Tester*, in the Owner drop-down list on the General tab for each state. The selections for the *User* field determine who is available as a primary owner. You can specify *role* [page 695] selections for the field in SBM Composer. User and group selections for *User* fields are made in SBM System Administrator.

The selections from that *User* field are available to users, but they can only select a single user as a primary owner for each state in which the the item resides. This ensures that primary items always have one user who is primarily responsible for it while it resides in each state.

Consider the following when you set primary ownership of items:

- When you configure a workflow, define single *User* fields before assigning the Owner property for each state. Then, in each transition that moves items into the state, move the *User* field that populates the Owner property into the Standard section so that it is available to all users who can transition the item.
- In some workflows and projects, you may want to pre-select a specific owner for items in each state and hide the *Owner* field so that the system alone manages ownership. For example, the manager of a specific project needs to be the owner of items in the "Submit" and "Assigned" states. You can modify the project and select a default value for the *Owner* field, and then move the field to the Hidden section in the "Submit" and "Assigned" states. The *Owner* field automatically contains the value. Therefore, ownership does not need to be selected by the user.
- You can set the *Owner* field as read-only if you want to keep the field visible. This lets you explicitly see who the owner of the item is and prevents users from changing the owner.
- You can select "None" as the ownership value for a state. While this is an appropriate value in some cases, such as closed items, keep in mind that an item without an owner might be forgotten.

Considerations for Secondary Ownership

You can set up a workflow so that one or more users are secondarily responsible for items in a particular state. For example, you can set up a queue into which all new incidents are submitted and let users who are specified as secondary owners for new incidents transition items from the queue into their inbox. Or, you can designate a single user as a *secondary owner* [page 696] for a particular state, letting this user transition items from that state instead of the primary owner.

A *User*, *Multi-User*, or *Multi-Group* field can be used to populate the secondary ownership property. The selections for the *User*, *Multi-User*, or *Multi-Group* field determine which users and groups are available as secondary owners. You can also specify whether users can select individual users or groups as secondary owners.

Consider the following when you set secondary ownership of items:

- The *Secondary Owner* field is an optional *system field* [page 699] that must be added to a *primary table* [page 692] before you can configure secondary ownership.
- When you configure a workflow, define *User*, *Multi-User*, and *Multi-Group* fields before assigning the "Secondary Owner" property for each state. Then, in each transition that moves items into the state, make sure the appropriate *User*, *Multi-User*, and *Multi-Group* field is moved so that it appears in the Standard section and is available to all users.
- You can set up your workflow so that items in a particular state have a primary owner and secondary owners, or you can specify that items in a state have only secondary owners. For example, a manager may want to assign items to a group of several users, who can then decide who should take ownership of the items. Each member of this group can be specified as secondary owners since the group as a whole has responsibility for items in that state. When items are assigned to the group, each user in the group may see the item on his or her home page and may receive e-mail notifications about the items.
- You can also use a *Multi-Group* field to define secondary ownership, letting members of multiple groups set selections for the field to transition items as needed.
- Changing the **Selection Mode** to **Groups & Users** for the *Secondary Owner* field in SBM Composer enables the display of user and group names within the same field.

  Groups will only appear in the *Secondary Owner* field if the source field that determines secondary ownership contains group names and the **Selection Mode** for *Secondary Owner* is set to **Groups & Users**. For example, if you change a *Multi-User* field's **Selection Mode** to **Groups & Users**, and the *Multi-User* field sets the value for *Secondary Owner*, then any groups are that are selected in the *Multi-User* field will appear in *Secondary Owner* as well, as long as *Secondary Owner's* **Selection Mode** is also set to **Groups & Users**.

  Note: If you change the **Selection Mode** for either a *Multi-User* or *Secondary Owner* field back to **Individual Users** after populating groups in either field, performance may be impacted when the groups are unrolled in order to display large lists of individual users. Alternatively, you can improve performance by selecting **Groups & Users** for any *Secondary Owner* field that is auto-populated by a *Multi-Group* or *Multi-User* field that contains groups: the list of users that would otherwise appear in *Secondary Owner* are rolled up into groups after selecting **Groups & Users** in that case.

  When a user updates an existing item after making this change to *Secondary Owner*, the *Secondary Owner* field will be updated with the values from the source field that determines secondary ownership. The auto-populated values in *Secondary Owner* will be rolled up into groups only if groups were selected in the *Multi-User* or *Multi-Group* field that determines secondary ownership.
- You can select None as a secondary owner value for a state.

If you configured your workflow so that only secondary ownership is enabled for a state, the primary owner value displays as "None" in the *state change history* [page 697], change history, and the *Owner* field.

Managing States

This section contains information about managing states in application workflows.

- About States [page 294]
- Adding a State [page 296]
- Adding an Owner Field to a State [page 300]
- Adding a Secondary Owner to a State [page 301]
- Renaming a State [page 302]
- Moving and Resizing a State [page 302]
- Duplicating a State [page 303]
- Previewing and Opening a State Form [page 303]
- Deleting a State [page 304]

About States

A state is a position in an *application workflow* [page 680] where a *primary item* [page 691] resides. A state provides accountability as items are transitioned through the workflow process. While an item is in a given state, it has a *primary owner* [page 691] who is responsible for performing a specific task with the item before it can be transitioned to the next state. You can also set up an application workflow so that one or more users have secondary responsibility for items while they are in a particular state.

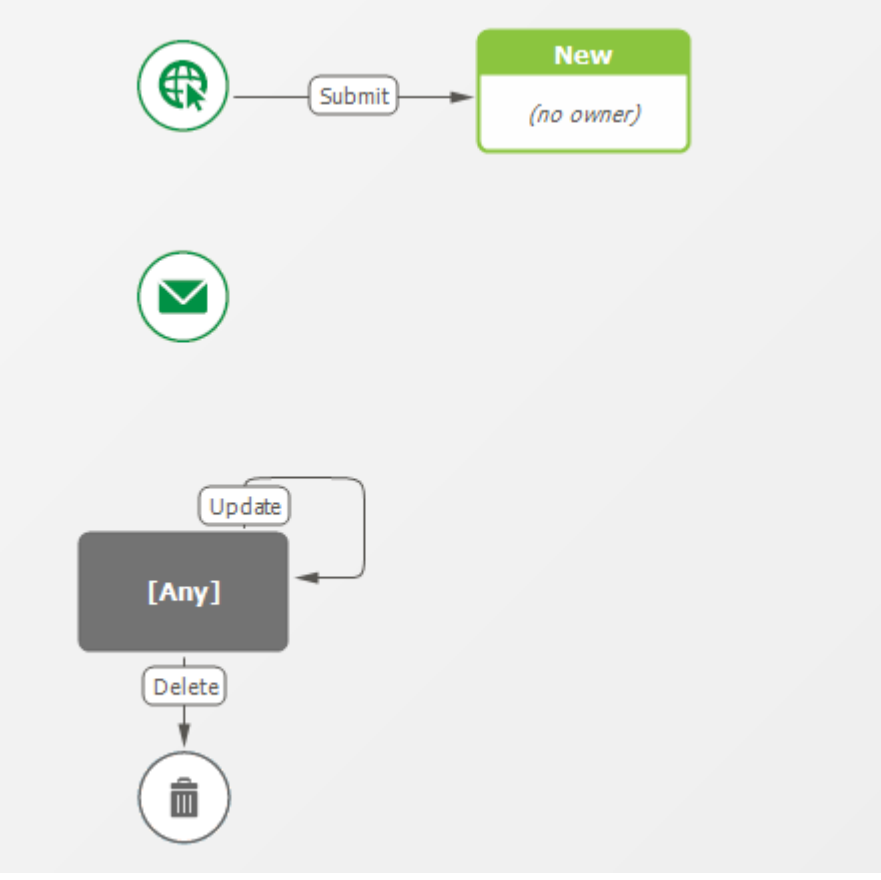
When a user completes a task associated with the primary item, the user can transition the item to the next state. For example, **Assigned**, **In Process**, **In QA**, and **Closed** are states in the workflow. When the owner of an assigned item begins work on the item, the owner transitions it to the **In Process** state. It cannot leave the **In Process** state until the owner transitions it to the **In QA** state. The item remains in this state until the new owner tests the work being tracked by the item and transitions it to the **Closed** state.

States can be *active* or *inactive*. In an active state, the owner of the state is to some degree engaged in a task. In an inactive state, activity on the item is suspended. In the previous example, **Assigned**, **In Process**, and **Testing** are active states, and **Closed** is an inactive state. The Workflow Palette [page 290] includes one active state and three preconfigured inactive states. All states can be renamed and reconfigured.

Note: There are four system-provided states included in every workflow that serve special purposes. For more information, see System-Provided States [page 294].

System-Provided States

All workflows contain four unique "states" that are not available from the **Workflow Palette**, but that can be viewed in the workflow editor. You cannot add these states to or delete them from the workflow, and their properties cannot be changed.



- **Submit** is represented by a globe with a mouse pointer. This is one of the the starting points in the workflow and corresponds to the user submitting an item in the SBM User Workspace.
- **Email** is represented by a circle with an envelope. This is the other starting point in the workflow and corresponds to the user submitting items through e-mail.
- **Any** represents every other state in the workflow in that it allows you to create one or more transitions that apply to every state in the workflow. By default, every state in the workflow has an **Update** transition button, which allows users with appropriate privileges to edit fields in the item without leaving the current state; and a **Delete** transition button, which allows users with appropriate privileges to delete an item from any state. You can add other transitions from the **Any** state; for example, a **Defer** transition that moves an item in any state to a **Deferred** state.

  You cannot have an outgoing transition from a regular state to the **Any** state. You can, however, have an outgoing transition from a decision to the **Any** state, if the only incoming transition to the decision is from the **Any** state.

  Tip: The **Any** state is provided so you do not have to create separate transitions from every state in the workflow. For example, if the **Any** state did not exist, and you wanted to have a **Delete** transition button in every state form, you would have to add outgoing **Delete** transitions from every state to the **Deleted** state.

- **Deleted** is represented by a trash can. It serves as the target state for **Delete** transition types. Items sent to this state are permanently deleted from the database.

Adding a State

You can add states to an *application workflow* [page 680] or sub-workflow. Primary items in your system are either active or inactive when they reside in each state.

To add a state:

1. To add an active state, do one of the following:
   - Drag **State** from the **Common Items** section of the **Workflow Palette**.
   - Drag **Active** from the **States** section of the **Workflow Palette**.

   Note: These two objects are identical.

2. To add an inactive state, drag **Completed**, **Rejected**, or **Pending** from the **States** section of the **Workflow Palette**.

   Note: These three objects are identical, except for their names and properties, which can be changed.

3. Type a name for the state, and press the Enter key.
4. To reposition the state, drag it to a new location in the editor.

Note: You can also double-click the state in the **Workflow Palette** to add the state to the workflow. The state is added to the right of a selected state, on the right side of a selected transition, or to the center of the workflow editor if no state or transition is selected.

Tip: A state that is added to a swimlane takes on the swimlane color. A state that is not added to a swimlane takes on the color that corresponds to the owner of the state.

Modifying State Properties

When you select a state in an *application workflow* [page 680] or sub-workflow in the workflow editor, you can modify its properties using the tabs of the state Property Editor, which are described in the following sections:

- General Properties for a State [page 297]
- Form Properties for a State [page 297]
- Field Privileges for a State [page 298]
- Actions for a State [page 299]
- Transition Order for a State [page 299]

General Properties for a State

When you select a state in an *application workflow* [page 680] or sub-workflow you can modify its general properties.

To modify the general properties of a state:

1. Open the workflow or sub-workflow containing the state.
2. Select the state.
3. In the state Property Editor, click the **General** tab. See General Tab of the State Property Editor [page 297] for details.

General Tab of the State Property Editor

This tab is displayed when you select a state in the workflow editor.

Element	Description
Type	*(Display only)* The state type.
Name	The name of the state. For clarity, state names should be unique within a workflow as it appears to a particular *role* [page 695]; but your design could include multiple states with the same name that appear for users in different roles.  **Note:** You cannot rename the system states: **[Submit]**, **[Email]**, **[Any]**, and **[Deleted]**.
Status	Select the state status (active or inactive).  **Note:** This field is not displayed for the "system" states.
Owner	Select the owner of the state.  **Note:** This field is not displayed for the system states.
Subtask	Select a *subtask* [page 698] for the state.  **Note:** This field is not displayed for the system states.
Description	*(Optional)* Type a description of the state.

Form Properties for a State

When an *application workflow* [page 680] or sub-workflow is open in the workflow editor, you can modify its default forms.

To modify a state form:

1. Open the workflow or sub-workflow containing the state.
2. Select the state.
3. In the state Property Editor, click the **Form** tab. See Form Tab of the State Property Editor [page 298] for details.

Form Tab of the State Property Editor

This tab is available when you select a non-system state in the workflow editor.

Note: For information about transition forms, see About Forms [page 195].

Element	Description
Form	Select the form to be associated with the selected state. You can select one of the following items: • **[Inherit from workflow]** The default form that was specified for the workflow on the **Forms** tab of the workflow Property Editor is used. • **[quick form]** A form that was generated by the system is used. • A custom form that was created is used. **Note:** For details about the differences between quick forms and custom forms, see Quick Forms and Custom Forms [page 196].
New	You can base a new custom form on the *quick form* [page 693] for the state or on an existing custom form, or you can start with an empty form. (You can also create a state or transition form by right-clicking the **Forms** heading in App Explorer.)
Preview	View a mockup of the state form as it will appear to a user. You can use the controls at the top to view the form as it will be seen in a different state and by a user in a different *role* [page 695]. You can click the transition buttons at the top of the form to experience the user flow without having to deploy the process app.
Edit	If a custom form is selected for the state, click to open the form in the form editor.

Field Privileges for a State

When you select a state in an *application workflow* [page 680] or sub-workflow you can modify its field privileges.

To modify a state field privileges:

1. Open the workflow or sub-workflow containing the state.

2. Select the state.
3. In the state Property Editor, click the **Field Privileges** tab. See Field Privileges Tab of the Application Workflow, State, and Transition Property Editor [page 281] for details.

Actions for a State

When you select a state in an *application workflow* [page 680] or sub-workflow, you can modify its associated actions.

To modify the actions for a state:

1. Open the workflow or sub-workflow containing the state.
2. Select the state.
3. In the state Property Editor, click the **Actions** tab. See Actions Tab of the State and Transition Property Editor [page 299] for details.

Actions Tab of the State and Transition Property Editor

This tab is available when you select a non-system state or a transition (except **Update** or **Delete**) in the workflow editor.

Element	Description
Actions	Lists actions defined for the selected state or transition.  **Tip:** To the extent possible, actions are performed in the order in which they are listed here. Select actions and use **Move up** and **Move down** to change that order.
New	Opens the **Action Wizard** to create a new *action* [page 679].
Edit, View	Opens the selected action for modification in the **Action Wizard**.  **Note:** If the workflow containing this state or transition is not checked out, this button is labeled **View**, and you cannot change the action definition.
Delete	Removes the selected action.
Move up, Move down	Cause the selected action to be performed earlier or later.  **Note:** Some action types (such as pre- and post-state Web service calls) are order-dependent. For those types, these controls are unavailable.

Transition Order for a State

You can define multiple transitions for a state. When you select a state in an *application workflow* [page 680] or sub-workflow you can modify the order of its transitions. This

order determines the sequence in which standard transition buttons will be displayed in the button bar on the form for the state.

Note: Transitions are listed in descending order. That is, transitions that appear first on the from appear higher in the list.

To modify the transition order for a state:

1. Open the workflow or sub-workflow containing the state.
2. Select the state.
3. In the state Property Editor, click the **Transition Order** tab. See Transition Order Tab of the State Property Editor [page 300] for details.

Transition Order Tab of the State Property Editor

This tab is available when you select a non-system state in the workflow editor.

Element	Description
Transition ordering	Lists the transitions defined for the selected state. Transition details are displayed. Such details include the form used for the transition, the target state, and actions and rules defined for the transition. **Note:** Transitions are listed in descending order. That is, transitions that are displayed first on the form appear higher in the list. Use **Move up** and **Move down** to change the order.
Move up	Raises the order of the selected transition.
Move down	Lowers the order of the selected transition.

Adding an Owner Field to a State

Use the **Add Owner Field to State** wizard to quickly establish *ownership* [page 690] for items in a state by assigning a user field to the **Owner** property of the state. The *role* [page 695] or roles associated with this field identify which users can own items in this state. There is one *Owner* system field. The value of this field changes as the item moves through the workflow, and its value is taken from the field you specify in the **Owner** property of the state.

You first identify the roles, and then either select or create a user type field that is constrained by those roles to use as the owner for the state. For example, the owner of a **Researching** state could be a *Research* field with an associated **Developer** role. Only users with the **Developer** role can be assigned to the field and can become owners of items in this state.

Note: For information about states, see About States [page 294]. For information about adding a *secondary owner* [page 696] to a state, see Adding a Secondary Owner to a State [page 301]. For information about user type fields, see Custom Field Types [page 135].

To add an owner to a state:

1. Select a regular state in an *application workflow* [page 680].
2. On the **General** tab of the state Property Editor, click the drop-down list next to the **Owner** field, and select **<Add Owner>**. Alternately, you can right-click the state in the application workflow and then select **Add Owner**.

   Note: The drop-down list next to the **Owner** field in the state Property Editor contains all valid potential owners for the state. If you want to use an existing field, you can select it from this drop-down list.

3. Follow the instructions on the wizard screens.

Select or Add Roles Screen

On this screen, specify the role or roles that define the users who can own items in the selected state. You can select existing roles or add a new role. If you are adding a new role, you can select an existing role to use as a template. The new role has the same privilege settings as the template role.

Field Selection Screen

On this screen, create a new user field or select an existing user field.

If you are creating a new field, specify its name and database name.

If you are selecting an existing field that corresponds to the selected roles, note that the number of available fields in the list is shown in the title bar of the screen. For example, **2 matching field matching field(s)** is shown if there are two fields in the drop-down list.

Adding a Secondary Owner to a State

Use the **Add Secondary Owner to State** wizard to quickly establish secondary *ownership* [page 690] for items in a state by assigning a *User*, *Multi-User*, or *Multi-Group* field to the **Secondary Owner** property of the state. The *role* [page 695] or roles associated with this field identify which users can be secondary owners of items in this state. There is one *Secondary Owner* [page 696] system field. The value of this field changes as the item moves through the workflow, and its value is taken from the field you specify in the **Secondary Owner** property of the state.

You first identify the roles, and then either select or create a user type field that is constrained by those roles to use as the secondary owner for the state. For example, you can set up a queue into which all new incidents are submitted. Users who are specified as secondary owners for new incidents can transition items from this queue into their inboxes. Alternately, you could designate a single user as a secondary owner for a state, allowing this user to transition items from that state instead of the *primary owner* [page 691].

Note: For information about states, see About States [page 294]. For information about adding an owner to a state, see Adding an Owner Field to a State [page 300]. For information about user type fields, see Custom Field Types [page 135].

To add an secondary owner to a state:

1. Make sure the *Secondary Owner* field has been added to the *primary table* [page 692]. The *Secondary Owner* field is an optional *system field* [page 699] that must be added to a primary table before you can configure secondary ownership.

2. Select a regular state in an *application workflow* [page 680].
3. On the **General** tab in the state Property Editor, click the drop-down list next to the **Secondary owner** field, and select **<Add Secondary Owner>**. Alternately, you can right-click the state in the application workflow and then select **Add Secondary Owner**.

   Note: The drop-down list next to the **Secondary Owner** field in the state Property Editor contains all valid potential owners for the state. If you want to use an existing field, you can select it from this drop-down list.
4. Follow the instructions on the wizard screens.

Select or Add Roles Screen

On this screen, specify the *role* [page 695] or roles that define the users who can be secondary owners of items in the selected state. You can select existing roles or add a new role. If you are adding a new role, you can select an existing role to use as a template. The new role has the same privilege settings as the template role.

Field Selection Screen

On this screen, create a new user field or select an existing user field.

If you are creating a new field, specify its name, database name, and field type. The field type can be *User*, *Multi-User*, or *Multi-Group*. For a description of the *field types* [page 686], see Custom Field Types [page 135].

If you are selecting an existing field that corresponds to the selected roles, note that the number of available fields in the list is shown in the title bar of the screen. For example, **2 matching field(s)** is shown if there are two fields in the drop-down list.

Renaming a State

You can rename a state in an *application workflow* [page 680] or sub-workflow.

To rename a state:

- Do one of the following:
  - Right-click the state, select **Rename**, type the new name, and press the Enter key.
  - Select the state, and on the **General** tab of the state Property Editor, modify the name in the **Name** field.

Moving and Resizing a State

In the workflow editor, you can move a state anywhere in the same workflow or sub-workflow. You can also resize a state manually or adjust it so that it automatically fits the title inside the box.

To move a state:

- Drag the state to its new location in the workflow.

To resize a state:

1. Select the state.

2. Select a point on the border of the state, and drag it to resize the state.

To resize a state to fit the title:

- Right-click the state, and then select **Size to Text**.

Duplicating a State

You can duplicate any state in an *application workflow* [page 680] or sub-workflow.

The default name for a duplicated state is *state name* 2. If you duplicate the state again, 2 is incremented to 3, and so on. For example, if you duplicate the state Unit Test, the duplicated workflow is named **Unit Test 2**. If you duplicate Unit Test 2, it is named **Unit Test 3**.

To duplicate a state:

- In the workflow editor, right-click the state, and select **Duplicate**.

  A duplicate (copy) of the state is created in the workflow.

  Note: The duplicate is unreachable until you connect it with a transition *from* another state. Use the **Unreachable Paths** option in the **Show/ Hide** area of the **Design** tab of the Ribbon to change the appearance of unreachable states and transitions.

Previewing and Opening a State Form

You can preview the form for a state. The preview opens in a separate window.

Unless the state inherits its form from the workflow or simply uses its *quick form* [page 693], you can also open the form for a state in the form editor.

To preview a state form:

- Do one of the following:
  - Right-click the state, and select **Preview State Form**.
  - Select the state. On the **Form** tab of the state Property Editor, click **Preview**. Alternatively, you can press F5 while in the form editor.

To edit a state form:

- Do one of the following:
  - Right-click the state, and select **Open State Form**.
  - Select the state. On the **Form** tab of the state Property Editor, click **Edit**.

  Note: The **Open State Form** command does not appear on the context (right-click) menu if the selected state inherits its form from the workflow or if the state uses its "quick form."

Tip: You click the transition buttons at the top of the state form to experience the user flow without having to deploy the process app. For more information, see Navigating through States and Transitions [page 290].

Deleting a State

You can delete most states from a workflow or sub-workflow.

Note: The **Submit**, **Email**, **Any**, and **Deleted** states cannot be deleted, even if they were renamed.

To delete a state:

- Do one of the following:
  - Right-click the state, and select **Delete**.
  - Select the state, and press the Delete key.

Managing Transitions

This section contains information about managing transitions in application workflows.

About Transitions

A transition moves an item from state to state. Most transitions are available as buttons on state forms. Users click a transition button to initiate the transition. In many cases, users can enter data about an item on a transition form before executing the transition.

Transitions can be customized in many ways. For example:

- You can specify that a transition applies only to certain item types, such as bugs and problem reports.
- You can specify that only members of particular groups can use the transition.
- The buttons that users click in the SBM User Workspace to execute a transition are automatically added to the top of a transition or state form. You can add custom

transition controls (buttons, hyperlinks or images) to a custom state or transition form. These controls can be used in addition to or instead of the standard buttons. For more information, see Custom Transition Controls [page 197] and Behavior Tab of the Control Property Editor [page 208]. For use cases and instructions, see Custom Transition Control Tutorials [page 260]

Inheritance Rules for Transitions

Every transition contains a set of properties that define how the transition acts, such the originating and receiving states, and the fields needed for data collection from users during the transition. These types of transition properties are inherited throughout the application workflow hierarchy. This lets you configure transition properties once rather than for each workflow. For example, in an application workflow, you can create transitions that are needed throughout all the workflows. The individual transition properties, like the **From** and **To** properties, are set once in the parent workflow and inherited in any sub-workflows and duplicated workflows.

About Transition Types

This topic describes the transition types you can add to an application workflow or sub-workflow.

Regular

Regular transitions let users transition items from one state to another.

Quick

Quick transitions transition items from one state to another without showing a form to users. Outgoing transitions from decisions are automatically configured as quick transitions, because the workflow logic, not the user, determines the next state in the workflow.

Post

Post transitions lets users submit into a different project a new item from the item they are currently working with. The posted item receives a new Item ID and follows the workflow of the project it is posted into. A link is created between the original item and the new item.

Post transitions can perform the following actions:

- Post a new item to another project using the same *primary table* [page 692]. By default, items posted within the same primary table contain all the fields values of the original items, except for the *Submitter* and *Submit Date/Time* fields. These fields are populated with the user who posts an item and the date the item is posted. You can, however, change this default field mapping.
- Post a new item to a project using a different primary table. Items posted across tables by default contain the **Title** and **Description** of the original item. You can, however, map compatible *field types* [page 686] between tables.
- Post a new item to an *auxiliary table* [page 680]. Items posted across tables by default contain the **Title** and **Description** of the original item. You can, however, map compatible field types between tables.

Examples

- **Posting Within a Single Primary Table:** An engineering team and a documentation team post items into separate projects and each of these projects follows a unique workflow. The engineering team could receive an enhancement request that affects documentation, so a new item can be posted to the documentation workflow from the original item. Each team can work on their respective tasks regarding the items and move them through both workflows independent of each other.

  Note: A **Post** transition used within a single primary table differs from a **Copy** transition in that a **Post** transition always creates a new item in the selected project, whereas a **Copy** transition copies an item in its current state and places it in another project. The copied item resides in the new project in the state it was copied from, along with all the data provided from the original item (except its *state change history* [page 697]). For example, an item in the **Resolved** state that is copied to the new project remains in the **Resolved** state in both projects.

- **Posting Across Primary Tables:** A Support Center representative receives a call about a problem and submits an incident into a project associated with a support workflow, which uses the Incidents table. The representative posts an issue into an engineering project, which is associated with a workflow using the Issues table. Each team can work on its respective tasks regarding the items and move them through both workflows independent of each other. See Creating a Post Transition Between Two Primary Tables [page 324] for details on setting up this example.

  Note: Triggers can be used with **Post** transitions to transition linked items when one or the other reaches a specific state in a workflow.

- **Posting from a Primary Table to an Auxiliary Table:** A sales team uses a process app to track prospects. Items containing information about prospects are submitted into a project associated with a workflow using a Customer Prospects primary table. When a prospect becomes a customer, a sales representative can post the item to the Contacts table. You can set up the **Post** transition to map fields gathered during the prospect process to those in the Contacts table, eliminating the need to gather contact information twice.

Note: You can use an action on a **Post** transition action to cause a second transition to execute after the workflow reaches the transition on which the action is defined. For details, see Tutorial: Submitting Multiple Primary Items [page 389].

Subtask

A *subtask* [page 698] is a *primary item* [page 691] that is associated with a principal task and that is displayed differently from the principal task. Subtasks are typically used when a set of smaller tasks needs to be worked on before a larger task, or principal task, can continue its process. A **Subtask** transition lets users create a new primary item in a specified project. You can specify that the new item be linked to the source (original), or principal, item and that subtasks are transitioned according to values in a specific *Binary/Trinary* or *Single Selection* field. For example, when all subtasks are completed, the *principal item* [page 692] could be transitioned to a completed state.

Subtask transitions are useful for setting up an entire set of new tasks at once. For example, at the beginning of a large project, a user can create a principal task and use a **Subtask** transition to create subtasks from that larger task.

After adding a **Subtask** transition to your workflow, you can define actions that transition subtasks and principal items. See Chapter 19: Working with Transition Actions [page 381] for details.

Viewing Subtasks in the SBM User Workspace

Users who are granted the **View Principal and Subtasks** privilege and who selected the **Subtasks** display preference can view the **Subtasks** section in the **Item Details** frame of the SBM User Workspace. The **Subtasks** section is visible only if an item is a subtask of another item or has one or more subtasks associated with it. In addition, users can access subtasks only in projects for which they have "view" privileges.

Tip: Subtasks appear as links in the **Subtasks** section of the SBM User Workspace. By default, these links contain the Item ID and Title of the item. You can modify this display using the value display format setting using the Property Editor for the primary table used by the subtask workflow.

Examples

- **Posting a new item into another project using the same primary table:** Subtasks created within the same primary table contain all fields values of the source (original) items, except for the *Submitter* and *Submit Date/Time* fields. These fields are populated with the name of the user who creates a subtask and the date it is created. You can, however, change this default field mapping. See Mapping Fields Within and Across Tables [page 326] for details.

- **Posting a new item into a project using a different primary table:** By default, values for the *Title* and *Description* fields are posted to the new item when a subtask is created in another primary table. See Mapping Fields Within and Across Tables [page 326] for details.

- **Posting a new item into another project using the same primary table:** Subtasks created within the same primary table contain all fields values of the source (original) items, except for the *Submitter* and *Submit Date/Time* fields. These fields are populated with the name of the user who creates a subtask and the date it is created. You can, however, change this default field mapping. See Mapping Fields Within and Across Tables [page 326] for details.

- **Posting a new item into a project using a different primary table:** By default, values for the *Title* and *Description* fields are posted to the new item when a subtask is created in another primary table. See Mapping Fields Within and Across Tables [page 326] for details.

Note: You can use an action on a **Subtask** transition action to cause a second transition to execute after the workflow reaches the transition on which the action is defined. For details, see Tutorial: Defining Subtask-Driven Actions [page 383].

Publish

The **Publish** transition lets users quickly publish problems and resolutions pertaining to primary items to the Knowledge Base that is part of the SBM User Workspace. When users execute the **Publish** transition, Submit forms for the Problems and Resolutions tables automatically open. By default, the problem contains the **Title** and **Description** from the item. Users can modify this information and select other values for the problem and resolution, such as which folder the problem will be stored in and its visibility.

Considerations

- The **Publish** transition type is available for any workflow, but problems can be posted only to the Problems table, and resolutions can be posted only to the Resolutions table.
- Publish transitions are available only to users who have been granted privileges to submit items into the Problems table. In addition, users who have privileges to submit items into the Problems table but not the Resolutions table will not receive a Submit form for the Resolutions table after submitting a problem.
- Users who publish items to the Knowledge Base must be granted the **Add Items to Folders** privilege for the folders to which you want them to publish items. This privilege is granted on the **Folder** privileges sub-tab when you add or edit a user or group account in SBM System Administrator.
- You can choose to specify **Publish** transitions as "quick transitions," letting users bypass the *Transition form* [page 700] for the workflow of the original item and to immediately submit a problem to the Problems table.
- You can map additional fields for a **Publish** transition.

Copy

The **Copy** transition lets users copy primary items and place them in another location in the project hierarchy within the same table.

Considerations

- When users click the **Copy** transition button, instead of the transition moving the item to the next state in the current project, a copy of the item is created in the specified project.
- The project to which items will be copied must be set up to allow new items to be submitted. Verify that that the **Allow New Items to be Submitted** check box is selected on the **General** tab of the **Add Project** or **Edit Project** dialog box in SBM System Administrator. Then make sure that users who will execute the transition have privileges to submit items into the project to which items will be copied.
- The project hierarchy does not open for users for those transitions that are configured to copy to a specific project. If you select the **Select at Runtime** option for the **Copy** transition in SBM System Administrator, the project hierarchy is displayed for users to select the project to which to copy an item.
- A new item ID is provided for the copied item.
- **Copy** transitions copy all field values from the source item to the new item, except for the *Submit Date/Time* field. This field is populated with the date the item is copied. If you want specific field values set during the **Copy** transition, use the transition field overrides in the destination project in SBM System Administrator to set or clear the affected fields.
- By default, the submitter for the copied item is the same user who submitted the original item. To set the submitter for the copied item as the user who executed the **Copy** transtion, edit the **Copy** transition for the project you are submitting the copied item into, and then edit the *Submitter* field. To do so, in SBM System Administrator, on the **Attributes** tab for the transition, select **Allow Override**, and

then select the **Set Value to Default Value** option. Select the **Options** tab, and and then select **(Current User)** from the **Default Value** drop-down list. Click **OK** to save your changes.

- **Copy** transitions always have a "To State" of "Same."
- Ensure that **Copy** transitions do not change the project, state, or both in such a way as to make the item fall outside the workflow of the target project. To avoid this, only allow copying of items between projects that share inherited workflow properties. This lets all states and transitions available to the source item's project to be available in the newly copied item's final project.
- You can configure a project to only allow items to be submitted from the **Copy** transition, instead of being submitted manually by users through a **Submit** transition. To do so, in SBM Composer, clear the **Default submit transition** check box on the **Options** tab of the Property Editor for the **Submit** transition. Alternatively, in SBM System Administrator, select the **No Button on Form** check box on the **General** tab of the **Edit Transition** or **Add Transition** dialog box.

Update

Update transitions are unusual in that they do not move items from state to state. Instead, **Update** transitions let users with the appropriate privileges update data in an item at a particular state in the workflow. You can set the "from" state for **Update** transitions, but the "to" state is always the same state. This transition is automatically included on the **Any** item in new workflows, but you can disable that default **Update** transition and create others that are available only for selected states.

Update transitions are controlled by the "update" privileges in the Item privilege category. Users who are granted these privileges can update individual items using the **Update** transition, or modify multiple items using the Rich Editable Grid.

Important:

The **Update Only** mode in the Editable Grid attempts to update items by first trying to use the default **Update** transition. If this transition is disabled, then it uses the first available **Update** transition named *Update*. If neither of these options are available or if the user does not have update privileges, the update will fail.

Regular transitions named *Update* are not used in **Update Only** mode, but these transitions can be used when users select the **All Transitions** option for editing items in the grid.

Delete

Delete transitions do not move items from state to state. They let users with the appropriate privileges delete an item at a particular state in the workflow. You can set the "from" state for **Delete** transitions, but the "to" state is always **Deleted**. This transition is automatically included on the **Any** state in new workflows, but you can disable that default **Delete** transition and create others that are available only for selected states or in particular sub-workflows.

Note: Field overrides and triggers do not apply to custom **Delete** transitions.

External Post

External Post transitions let users transition items from one state to another, as well as generate and send an e-mail message to an external database requesting that the item be posted. You select the external database to which you want the item to be added.

Adding a Transition

You can add transitions between states in an *application workflow* [page 680] or sub-workflow.

To add any transition (except the Copy and Update transitions) between states:

1. From the **Workflow Palette**, drag a transition type onto the state in which you want the new transition to begin.
2. Release the mouse button.
3. Click the state in which you want the new transition to end.
4. Type a name for the transition, and then press the Enter key.

Note: You can also double-click a transition in the **Workflow Palette** to add the transition to the workflow. The transition is placed between the two most recently selected states, and points from the first state that was selected to the second state. If no states were selected, no transition is added. For the **Delete** transition, only one state needs to have been selected, and the transition is automatically drawn from that state to the **Delete** state.

To add a Copy or Update transition to a state:

1. From the **Workflow Palette**, drag the **Copy** or **Update** transition onto the state.
2. Type a name for the transition, and then press the Enter key.

Note: You can also double-click a **Copy** or **Update** transition in the **Workflow Palette** to add the transition to the workflow. The transition is added to the selected step. For these transitions, only one state needs to have been selected.

Note: Transition buttons are automatically added to the top of the form in the SBM User Workspace. Users click these buttons to execute transitons. You can replace the buttons with button controls (buttons, hyperlinks, or images) that can be placed anywhere on the form. For more information, see Behavior Tab of the Control Property Editor [page 208].

Modifying Transition Properties

When you select a transition in an *application workflow* [page 680] or sub-workflow in the workflow editor, you can modify its properties using the tabs of the transition Property Editor, which are described in the following sections:

- General Properties of a Transition [page 311]
- Options for a Transition [page 312]
- Post Options for a Transition [page 315]
- Form Properties for a Transition [page 319]

- Field Privileges of a Transition [page 320]
- Field Overrides for a Transition [page 321]
- Actions for a Transition [page 321]
- Restrictions by Type for a Transition [page 321]
- Restrictions by Role for a Transition [page 322]

General Properties of a Transition

When you select a transition in an *application workflow* [page 680] or sub-workflow, you can modify its general properties.

To modify the general properties of a transition:

1. Open the workflow or sub-workflow containing the transition.
2. Select the transition.
3. In the transition Property Editor, click the **General** tab. See General Tab of the Transition Property Editor [page 311] for details.

General Tab of the Transition Property Editor

Use this tab to modify the general properties of a transition.

Element	Description
Name	The name of the transition. For most transitions, the name appears as a button on the item details page that users can click to transition items to another state. Certain transition names do not appear as button labels. For example, names for **Submit** transitions are not used on the **Submit** button in the item in the SBM User Workspace. However, the *state change history* [page 697] and *change history* [page 681] reflect the name you provide here.  **Note:** You can replace a transition button with a button, hyperlink, or image control that can be placed anywhere on the form. For button and hypertext controls, you either use the transition name as the control label, or type a different name. For more information, see Behavior Tab of the Control Property Editor [page 208].
Type	Select a transition type from the list. You set the transition type by dragging the correct type from the **Workflow Palette**, but you can change it here. See About Transition Types [page 305].

Element	Description
From	The state where the transition begins. You set the **From** state when you drag the transition from the **Workflow Palette**, and you can change it by dragging the starting end of the transition to a different state. Every workflow includes two special-purpose states of interest: • **E-Mail:** Used to define a **Submit** transition for e-mail submissions from outside users or from another SBM database through cross-database posting. • **Any:** Used to define a transition that is available from every state in a workflow. For example, you could define a **Post** transition that lets users post an item to another project, no matter which state the original item is in.
To	The state where the transition ends. You set the **To** state when you drag the transition from the **Workflow Palette**, but (except for **Copy** and **Update** transitions) you can change it by dragging the ending (arrowhead) end of the transition to a different state.
Status	Specifies whether the transition is enabled or disabled. You can tailor your *application* [page 679] by enabling or disabling transitions for child workflows and for projects. **Tip:** To simplify maintenance of your process, create transitions at the highest workflow level, and then disable or enable transitions as they apply to child workflows or to projects. • **Enabled:** Enables the transition for the selected workflow or project. If you are setting the status for a transition for a child workflow or for a project, select this option to use transition properties set for this transition rather than inherited properties. • **Disabled:** Disables the transition for the selected workflow or project. **Note:** You cannot disable the **Otherwise** transition from a decision. • **Inherited:** Assigns the status in a child workflow to be the same as in the parent workflow. This option is selected by default for transitions in a child workflow.
Description	*(Optional)* A description of the purpose of the transition.

Options for a Transition

When you select a transition in an *application workflow* [page 680] or sub-workflow you can modify its options.

To modify the options for a transition:

1. Open the workflow or sub-workflow containing the transition.
2. Select the transition.
3. In the transition Property Editor, click the **Options** tab. See Options Tab of the Transition Property Editor [page 313] for details.

Options Tab of the Transition Property Editor

Use this tab to modify various options for the transition.

Options

Element	Description
Quick transition [page 693] (do not show a form)	Bypasses the *transition form* [page 700]. This option lets users quickly transition items out of a particular state if fields do not need to be modified. You can also use this option to bypass the transition form for the original item for **Post** or **Subtask** transitions. The form opens if there are required fields specified for the transition. If you select this option, you cannot select the **Required attachment**, **Show "New Note" field**, and **Require "New Note" entry** options.
Default submit transition	Specifies a transition from a **Submit** state for items that a user submits manually. This option should not be used for transitions invoked indirectly, such as **Post** and **External Post** transitions that submit items automatically. By default, the first transition from a **Submit** state has this option selected, and other transitions have this option cleared.
Hide button on form	Hides the transition button on the state form. This option applies to all transitions except those that start from the **Submit** state. This option lets you hide transition buttons for transitions that are invoked indirectly. Examples of transitions that are invoked indirectly are those that use triggers or **Subtask** transitions.
Show "New Note" field	Inserts a **New Note** field during the selected transition. The field appears after the **Standard Field** section and lets users add a note to the item, automatically recording their name and a date/time stamp when the note was created.
Require "New Note" entry	Specifies the **New Note** field as requiring an entry. Users must enter a note to transition an item successfully.

Attachment

Element	Description
Required attachment	If attachments are not required for the transition, select **<None>** . Otherwise, select the type of attachment to be required.

Authentication

Element	Description
Required, Not required, Inherited	**Important:** If transition authentication is used, it is recommended that the SBM Server use HTTPS protocol. Select **Required** if users should supply their login IDs for the transition. The transition fails if a user does not provide the correct login ID and password or attempts to provide the login ID and password of another user. On successful authentication, the login ID and password are recorded as an electronic signature. If you select **Required**, you can also select an authentication option from the *DateTime* field to update list to record the time the transition was performed. The transition is stored in the transitions section of the **Change History** section for each item in the SBM User Workspace.

Element	Description
When selecting a *Date/Time* field for transition authentication: • Only *Date/Time* fields that are set to display the date and time or only the date are available for this option. • System fields cannot be used for transition authentication. • Deleted fields cannot be used for transition authentication. • Consider naming the *Date/Time* field so that it is clear that is used for transition authentication. Examples include "Authentication Time" and "Electronic Signature Recorded At." • The *Date/Time* field specified for transition authentication is always populated when the transition is executed. To prevent users from changing the date and time, consider moving the field to the **Hidden Fields** section or another section controlled by privileges. You can also set the field as read-only. • *Change history* [page 681] for transition authentication is not recorded on **Submit**, **Copy**, or **Delete** transitions. When setting authentication options for transitions: • If your system uses NT Challenge Response, passwords are checked against internal SBM System Administrator passwords. Users should synchronize their SBM System Administrator passwords and their network passwords, unless they choose to specify a unique password for authenticating SBM System Administrator transitions. • If your system uses LDAP authentication, LDAP handles password verification. • Authentication settings apply only to transitions that are executed manually by users in the SBM User Workspace. Automatic transitions that require authentication will fail. Use care when setting authentication options for transitions that are executed as part of actions, by e-mail submission, or by API programs. • You can override authentication options for child workflows and for projects. For best results, set authentication options at a parent workflow, and then enable or disable them for child workflows and projects as needed.	

Post Options for a Transition

When you select a transition in an *application workflow* [page 680] or sub-workflow, you can modify its post options.

To modify the post options for a transition:

1. Open the workflow or sub-workflow containing the transition.
2. Select the transition.
3. In the transition Property Editor, click the **Post Options** tab. See Post Options Tab of the Transition Property Editor [page 316] for details.

Post Options Tab of the Transition Property Editor

Use this tab to modify the post options of a **Post**, **External Post**, **Publish**, **Subtask**, or **Copy** transition. You cannot set post options for an **Update**, **Delete**, or **Regular** transition.

Post Settings

Element	Description
Post table	Select the table to which you want items to be posted. For **Post** transitions, you can select a *primary table* [page 692] or an *auxiliary table* [page 680]. For **Copy** and **Subtask** transitions, you must select a primary table.
Project	*(Display only)* By default, users with appropriate privileges are presented with a list of projects to submit the post item or subtask into. You can limit the list of projects by specifying a project or list of projects for the transition in SBM System Administrator or by granting privileges to specific projects to which users can post items.
Set new item in this item's field	This drop-down list contains *Single Relational* and *Multi-Relational* fields in the table for the current item that are related to the specified post table. Select a field that will contain the Item ID and Title of the posted item in the original item. For example, if you create a **Post** transition that lets users post from the Issues table to the Incidents table, you can create a *Multi-Relational* field to the Incidents table and select that field from the **Set new item in this item's field** drop-down list. When users post items to the Incidents table, the Incidents field in the original item contains information about the posted item. If you select a *Multi-Relational* field for this option, all items posted from the original item are selected as values for the field. If you select a *Single Relational* field, the last posted item is selected as the value.
Set this item in new item's field	This drop-down list contains *Single Relational* and *Multi-Relational* fields for the current primary table that are available in the receiving table. For example, if the transition lets users post items from the Issues table into the Incidents table, an Issues *Single-Relational* or *Multi-Relational* field contained in the Incidents table is listed here. Select a field that will contain the Item ID and Title of the original item in the posted item. For best results, select a *Single Relational* field for this option, because the only valid value is the originating item that created the posted item.  **Tip:** In the SBM User Workspace, a relational field icon appears next to fields populated using the **Post**, **Copy**, **Publish**, or **Subtask** transition type. Users with privileges to view items in the relational field table can click the icon to open the item in a pop-up window.

Element	Description
Use submit transition	Each non-default item in this drop-down list consists of a workflow name and a submit transition. From this list, you can select a specific submit transition that can be used for **Post** and **Subtask** transitions. This lets you set different properties for this transition. For example, you could require users to populate certain fields when they submit directly into a project, but require them to populate a different set of fields when they use a Subtask transition. This field also lets you specify a single project into which posted items and subtask items can be submitted, when more than one project is available. For example, if a process app contains a principal workflow and two subtask workflows, the submit transition from each of the subtask workflows appears in this list. Select the transition for the project into which this item should be submitted. (A project is created for each workflow when the process app is deployed.)
When finished, show...	Select **New item** to display the new item created by the transition after the transition is complete. Select **Original item** to return users to the item from which they execute the transition after the transition is complete.

Element	Description
Item link type *(Post, Subtask, Publish, and Copy transitions)*	Select one of the following: • **<None>**: Creates the transition without an item link type. • **1-Way, original to other, no trigger**: Creates a link from the original item to the new item without triggering a transition on the linked item. • **1-Way, original to other, with trigger**: Creates a link from the original item to the other item, and fires the trigger specified on the **Actions** tab on the original item. • **1-Way, other to original, no trigger**: Creates a link from the new item to the original item without triggering a transition on the linked item. • **1-Way, other to original, with trigger**: Creates a link from the new item to the original item, and fires the trigger specified on the **Actions** tab on the new item. • **2-Way, both trigger each other**: Creates links in both the original and new items, and fires triggers on both items. • **2-Way, no triggers**: Creates links in both the original and new items without triggering a transition on the linked item. • **2-Way, original triggers other**: Creates links in both the original and the new item, and fires the trigger specified on the **Actions** tab on the new item. • **2-Way, other triggers original**: Creates links in both the original and new items, and fires the trigger specified on the **Actions** tab on the original item.
Database name *(External Post transitions only)*	Select an external database into which the posted item will be added.

Mapping

Element	Description
This item's field	Lists fields in the primary table.

Element	Description
Mapped to	For each field in the primary table, select the name of the field to which the field for this item should be mapped. (For an **External Post** transition, the field is in the specified external database.) For **Post** and **Subtask** transitions, you can select the field from a drop-down list.

This table lists the available field mappings.

Source field type	Allowed receiving field type
Binary/Trinary	Text
Date/Time	Date/Time or Text
Folder	Text
Multi-Relational	Text
Multi-Selection	Text
Numeric	Numeric or Text
Single Relational	Text
Single Selection	Single Selection or Text
Sub-Relational	Text
Summation	Text
Text	Text (Memo)

Form Properties for a Transition

When you select a transition in an *application workflow* [page 680] or sub-workflow, you can modify its general properties.

To modify the general properties for a transition:

1. Open the workflow or sub-workflow containing the transition.
2. Select the transition.
3. In the transition Property Editor, click the **Form** tab. See Form Tab of the Transition Property Editor [page 320] for details.

Form Tab of the Transition Property Editor

Use this tab to select, create, edit, and preview the form for the transition selected in the workflow editor.

Note: For information about transition forms, see About Forms [page 195].

Element	Description
Form	Select the form to be associated with the selected transition. You can select one of the following items: • **[Inherit from workflow]** The default form that was specified for the workflow on the **Forms** tab of the workflow Property Editor is used. • **[quick form]** A form that was generated by the system is used. • A custom form that was created is used. **Note:** For details about the differences between quick forms and custom forms, see Quick Forms and Custom Forms [page 196].
Quick transition (do not show a form)	Bypasses the *transition form* [page 700]. This option lets users quickly transition items out of a particular state if fields do not need to be modified. You can also use this option to bypass the transition form for the original item for **Post** or **Subtask** transitions. The form opens if there are required fields specified for the transition. If you select this option, you cannot select the **Required attachment**, **Show "New Note" field**, and **Require "New Note" entry** options.
New	You can base a new custom form on the *quick form* [page 693] for the transition or on an existing custom form, or you can start with an empty form. (You can also create a state or transition form by right-clicking the **Forms** heading in App Explorer.)
Preview	View a mockup of the *transition form* [page 700] as it will appear to a user. You can use the controls at the top to view the form as it will be seen in a different state and by a user in a different *role* [page 695]. You can click the transition buttons or controls to experience the user flow without having to deploy the process app.
Edit	If a custom form is selected for the transition, click to open the form in the form editor.

Field Privileges of a Transition

When you select a transition in an *application workflow* [page 680] or sub-workflow, you can modify its field privileges.

To modify a transition's field privileges:

1. Open the workflow or sub-workflow containing the transition.

2. Select the transition.
3. In the transition Property Editor, click the **Field Privileges** tab. See Field Privileges Tab of the Application Workflow, State, and Transition Property Editor [page 281] for details.

Field Overrides for a Transition

When you select a transition in an *application workflow* [page 680] or sub-workflow, you can override its field properties.

Note: For information about *overrides* [page 690], and for examples of overriding fields, see Chapter 16: About Inheritance and Overrides [page 355].

To override a transition's field properties:

1. Open the workflow or sub-workflow that contains the transition.
2. Select the transition.
3. In the transition Property Editor, click the **Field Overrides** tab. See Field Overrides Tab of the Application Workflow and Transition Property Editor [page 282] for details.

Actions for a Transition

When you select a transition in an *application workflow* [page 680] or sub-workflow, you can modify its associated actions.

To modify the actions for a transition:

1. Open the workflow or sub-workflow containing the transition.
2. Select the transition.
3. In the transition Property Editor, click the **Actions** tab. See Actions Tab of the State and Transition Property Editor [page 299] for details.

Restrictions by Type for a Transition

When you select a transition in an *application workflow* [page 680] or sub-workflow, you can restrict the item types that can use it.

To restrict a transition by *item type* [page 688]:

1. Open the workflow or sub-workflow containing the transition.
2. Select the transition.
3. In the transition Property Editor, click the **Restrict By Type** tab. See Restrict By Type Tab of the Transition Property Editor [page 322] for details.

Restrict By Type Tab of the Transition Property Editor

Use this tab to restrict the item types that can use a transition.

Note: You cannot restrict **Update** or **Delete** transitions by *item type* [page 688], and cannot restrict the **Otherwise** transition from a decision.

Element	Description
Item Types	Select the item types that are allowed to use the transition. The item types listed are the values defined (on the **Options** tab of the Property Editor) for the *Item Type* field in the *primary table* [page 692] for the *application* [page 679].
Edit Item Types	For convenience, you can click this link to add, remove, or modify the values in the *Item Type* field for the primary table. Click in the quick access tool bar to return to the current view of the **Restrict By Type** tab.

Restrictions by Role for a Transition

When you select a transition in an *application workflow* [page 680] or sub-workflow, you can restrict the roles that can use it.

To restrict a transition by *role* [page 695]:

1. Open the workflow or sub-workflow containing the transition.
2. Select the transition.
3. In the transition Property Editor, click the **Restrict By Role** tab. See Restrict By Role Tab of the Transition Property Editor [page 322] for details.

Restrict By Role Tab of the Transition Property Editor

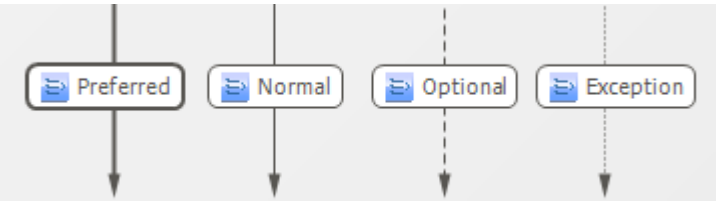
On the **Restrict by Role** tab for a **Post**, **External Post**, **Publish**, **Subtask**, or **Copy** transition, select any roles for which you want to restrict access to the transition.

Note: You cannot restrict **Update**, **Delete**, or **Regular** transitions by *role* [page 695], and cannot restrict the **Otherwise** transition from a decision.

Important: In SBM System Administrator, you can also specify groups who will be able to access the transition. In SBM System Administrator, any role-based restrictions that you set here in the transition Property Editor appear in a read-only state, and if any role-based restrictions exist, the administrator cannot clear the Restricted check box.

About Transition Styles

You can use transition styles to provide a visual reminder of the purpose, use, or importance of a transition. Transition styles have no effect on transition behavior or form layout.



To modify the style of a transition:

- Do one of the following:
  - Right-click the transition, select **Transition Style**, and then select the style.
  - Select the transition, and then click the style in the **Transition Style** area of the Ribbon.

Note: You can change the default transition styles. To do so, click the **Styles** heading in App Explorer, and then make changes in the styles editor.

Displaying Transition Labels

You can toggle the display of labels for all transitions. When labels are displayed, you can toggle the display of labels for individual transitions.

To toggle the display of all transition labels:

- In the **Show/Hide** area of the **Design** tab of the Ribbon, select or clear the **Labels** option.

To toggle the display of the label for an individual transition:

1. Right-click the transition.
2. On the menu that opens, select or clear **Show Label**.

Renaming a Transition

You can rename a transition in an *application workflow* [page 680] or sub-workflow.

To rename a transition:

- Do one of the following:
  - Right-click the transition, select **Rename**, type the new name, and press the Enter key.
  - Select the transition, and on the **General** tab of the transition Property Editor, modify the name in the **Name** field.

Adjusting a Transition

You can adjust a transition in the following ways:

- Move it (or part of it) in almost any direction.
- Add corners to route it around states and transitions.

- Move its starting or ending point to a different state.

  Note: A **Copy** or **Update** transition that starts on any state always ends on that same state.

- Use waypoints (small white circles on a selected transition) to bend a transition into an angle.

Previewing and Opening a Transition Form

You can preview the form for a *transition form* [page 700]. The preview opens in a separate window.

Unless the transition inherits its form from the workflow or simply uses its *quick form* [page 693], you can also open the transition form in the form editor.

To preview a transition form:

- Do one of the following:
  - Right-click the transition, and select **Preview Transition Form**.
  - Select the transition, and on the **Form** tab of the transition Property Editor, click **Preview**. Alternatively, you can press F5 while in the form editor.

To edit a transition form:

- Do one of the following:
  - Right-click the transition, and select **Open Transition Form**.
  - Select the transition, and on the **Form** tab of the transition Property Editor, click **Edit**.

  Note: The **Open Transition Form** command does not appear on the context (right-click) menu if the selected transition inherits its form from the workflow or if the transition uses its quick form.

Tip: You can click the transition controls on the transition form to experience the user flow without having to deploy the process app. For more information, see Navigating through States and Transitions [page 290].

Deleting a Transition

You can delete any transition from a workflow or a sub-workflow.

To delete a transition:

- Do one of the following:
  - Right-click the transition, and select **Delete**.
  - Select the transition, and press the Delete key.

Creating a Post Transition Between Two Primary Tables

This example describes how to create a **Post** transition that posts from one *primary table* [page 692] to another, without changing the state in the active workflow. It uses primary

tables named Issues and Incidents in an example process app containing Issue Management and Incident Management applications.

To create a Post transition between two primary tables:

1. In the App Explorer, select the workflow (such as Externally Submitted) in which you want to create the **Post** transition.
2. In the workflow editor, drag a **Post** transition from the **Workflow Palette** onto the workflow, releasing the mouse button over the state from which you want the post to occur.
3. Click on the same state again. This lets users post an item to the Issues table and leave the original item in the **New** state.
4. Type a name (such as `Post to Engineering`) for the transition. This name typically appears to the user as a button on the state form, so it is helpful to provide a name that conveys the type and purpose of the transition.
5. On the **Options** tab of the transition Property Editor, select **Quick Transition** to automatically transition the original item. This lets users quickly post the new item. This option should be selected only if there are no required fields for this transition or if you do not want users to provide data for the original item before posting it.
6. On the **Post Options** tab of the transition Property Editor:
   a. From the **Post table** drop-down list, select the table (such as Issues) associated with the *application* [page 679] to which you want to post items.

      Tip: For a **Post** or **Subtask** transition, you can select a primary table only if a dependency on the corresponding application is defined in the current process app.

   b. If your workflow contains *relational fields* [page 694] to the table and in the table selected in the **Post table** drop-down list, you can select these fields from the **Set new item in this item's field** and **Set this item in new item's field** drop-down lists. This lets you create a field relationship between the two tables.

      Tip: You can create a **Submit** transition in the receiving project that can be used by **Post** and **Subtask** transitions. You can configure field properties as needed for this transition.

   c. For **When finished, show**, select **Original item** to return users to the item from which they execute the **Post** transition, after the transition is complete. You would select **New item** to display the new item instead.

   d. From the **Item link type** drop-down list, select the type of link (such as "2-Way, no triggers") you want to add between the original and posted item. By default, no link is created.
7. From the Composer menu, click **Save** (or click the **Save locally** icon in the Quick Access toolbar) to save your workflow changes.

By default, values for the *Title* and *Description* fields are posted to the new item when users post to another primary table. See Mapping Fields Within and Across Tables [page 326] for information on posting additional field values.

Mapping Fields Within and Across Tables

The **Post**, **Publish**, and **Subtask** transition types let users post items within a *primary table* [page 692] or to other tables. Depending on the destination of the posted item, certain field values are automatically copied to the newly posted item. You can change this default field mapping for items posted within a primary table or to another primary or *auxiliary table* [page 680].

- **Mapping fields within a primary table:** By default, when users post items within a primary table, all field values of the original items, except for the *Submitter* and *Submit Date/Time* fields, are posted to the new item. The *Submitter* and *Submit Date/Time* fields are populated with the user who posts the item and the date the item is posted. Any additional field mappings you define for the transition also apply.

- **Mapping fields across tables:** By default, values for the *Title* and *Description* fields are posted across tables. If a value is not provided for the *Description* field for the original item, the *Description* field for the posted item is populated with the title of the original item. In addition, the default mappings are only used if you have not created any additional field mappings. If you create any field mappings for this transition, you will need to also supply mappings for the *Title* and *Description* fields.

Field Mapping Considerations for Posted Items

Consider the following information when you map fields for **Post**, **Publish**, and **Subtask** transitions:

- Selections for *Single Selection*, *Multi-Selection*, *Multi-User*, and *Multi-Group* fields should have identical names when mapped. If a selection is not available for the field in the receiving project or table, **None** is selected as the value in the posted item. If a selection has been disabled or deleted in the receiving project or table, **disabled** is displayed, and users must select a valid value before completing the transition.

- Default values for *Single Selection*, *Multi-Selection*, *Multi-User*, and *Multi-Group* fields in the receiving project or table are overwritten by values from the source item. For example, a **Post** transition is used to post items from the Issues table to the Incidents table, and the *Issue Type* field is mapped to the *Incident Type* field, which has a default value of **Problem Report**. If a user selects **Bug Report** for the **Issue Type** field before posting an item from the Issues table to the Incidents table, the **Problem Report** default value is ignored and is replaced by **Bug Report**.

- Fields must be mapped to fields of the same type, with the following exceptions:
  - All *field types* [page 686] map to *Text* fields.
  - *Sub-Relational* fields map only to *Text* fields. Because users do not specify data for *Sub-Relational* fields, data cannot be mapped to *Sub-Relational* fields but can be mapped from them as *Text* fields. The *Text* field of the new item is set to the display value of the *auxiliary item* [page 680] which is defined by the value display format setting for the *auxiliary table* [page 680].
  - *Summation* fields cannot be mapped for posted items.

- When mapping fields for **Publish** transitions, the mapping applies only to the Problems table, not to the Resolutions table.

Mapping Fields for a Post, Publish, or Subtask Transition

To map fields when you edit a workflow in which a Post, Publish, or Subtask transition exists:

1. In App Explorer, select the workflow that contains the **Post**, **Publish**, or **Subtask** transition for which you want to map fields.
2. In the workflow editor, select the transition.
3. On the **Post Options** tab of the transition Property Editor, scroll down to the **Mapping** area.
4. Under **This item's field**, select the field to be mapped to a field in another project or table.
5. Under **Mapped to** in the same row, use the drop-down list to select the field you want to map to (from the fields in the table you are posting to).

Working with Decisions

This section contains information about working with decisions in application workflows.

About Decisions

Decisions enable *conditional routing* [page 682] in application workflows or sub-workflows. The **Decision** is included in the **Workflow Palette**, and is added to a workflow by a drag-and-drop operation, the same way states and transitions are added to a workflow.

A decision has one or more incoming transitions and two or more outgoing transitions. An incoming transition connects a source state to the decision, and the outgoing transitions connect the decision to target states. When you add a decision to a workflow, two outgoing transitions are automatically added: **Otherwise** and **Branch**.

The decision evaluates rules that are associated with each outgoing transition, and the transition associated with the first rule that evaluates as "true" is executed. If a role is restricted from using an outgoing transition, the workflow evaluates the rule associated with the first non-restricted outgoing transition. If no outgoing transition is allowed for a role, then the transition associated with the "Otherwise" rule is executed. For information about rules, see Introducing Rules [page 339].

If a decision fails to complete when the application runs, the item that entered the step will remain in the source state. This prevents changes that were saved to the item by completed actions or by the user in a transition form from being lost. (Decision failures are caused by a user abandoning or cancelling a transition form or by an error.)

Restriction:

- You cannot use the **Post**, **Subtask**, or **Copy** transition type for incoming transitions to a decision. You cannot use the **Copy** transition type for outgoing transitions from a decision.
- Using a "Transition", "Orchestration Workflow (continue executing)," or "Trigger" action type on an incoming transition to a decision is not currently supported.
- You cannot restrict the **Otherwise** transition from a decision by role or type.
- There are some field comparison limitations in expressions used to create decision rules. For details, see Rule Operators [page 346].
- You can have an outgoing transition from a decision to the **Any** state, but only if there is only one incoming transition from the **Any** state to the decision.

Adding a Decision

You can add decisions to an *application workflow* [page 680] or sub-workflow.

To add a decision:

1. From the **Workflow Palette**, drag the **Decision** item to the workflow editor.
2. Type a name for the decision, and press the Enter key.
3. To reposition the decision, drag it to a new location in the editor.

Note: You can also double-click the decision in the **Workflow Palette** to add the decision to the workflow. The decision is added to the right of a selected state, on the right side of a selected transition, or to the center of the workflow editor if no state or transition is selected.

Modifying Decision Properties

When you select a decision in an *application workflow* [page 680] or sub-workflow in the workflow editor, you can modify its properties using the tabs of the step Property Editor, which are described in the following sections:

- General Tab of the Decision Property Editor [page 329]
- Rules Tab of the Decision Property Editor [page 329]

General Tab of the Decision Property Editor

This tab is displayed when you select a decision in the workflow editor.

Element	Description
Name	Type or change the decision name, if necessary.
Description	*(Optional)* Type a description of the decision.

Rules Tab of the Decision Property Editor

This tab is available when you select a decision in the workflow editor.

Note: For information about rules, see Introducing Rules [page 339].

Element	Description
Rule	Click a row in the **Rule** column to open the rule menu. All rules that are defined for the application are listed at the bottom of the menu. To map an existing rule to the transition, select the rule from the menu. To create a new rule for this transition, select **(New rule)**. The rule editor opens. One transition must be mapped to the "Otherwise" rule. This transition is executed if none of the other rules evaluates as "true."  **Important:** The rules are evaluated in the order in which they are listed here, so it is critical that the order is correct. (See **Move up, Move down** later in this table.)
Transition	The outgoing transitions from this **Decision**.
To state	Shows the target state for this transition.
Status	Shows whether the transition is enabled or disabled. Disabled transitions are highlighted in a light gray color on the workflow editor.
Hide disabled transitions	Removes disabled transitions from the **Transition** column.
Override transition ordering	*Sub-workflows only:* Enables the **Move up** and **Move down** buttons so you can change the transition order in a sub-workflow without changing the order in the parent workflow.
Show rule summaries	Expands the area under each row and provides a textual summary of the rule.

Element	Description
Show transition summaries	Expands the area under each row and shows the form the transition uses, the state the workflow moves to after the transition is executed, and other information about the transtition (if any).
Open rule	Opens the rule editor for the selected rule.
Move up, Move down	Changes the order in which rules are evaluated.  **Note:** After you map the "Otherwise" rule to a transition, it is moved to the bottom of the list, because that transition is taken if no other rule evaluates as "true."

Renaming a Decision

You can rename a decision in an *application workflow* [page 680] or sub-workflow.

To rename a decision:

- Do one of the following:
  - Right-click the step in the workflow editor, select **Rename**, type the new name, and press the Enter key.
  - Select the step in the workflow editor, and on the **General** tab of the step Property Editor, modify the name in the **Name** field.

Deleting a Decision

You can delete any decision from a workflow or a sub-workflow.

To delete a decision:

- Do one of the following:
  - Right-click the decision in the workflow editor, and select **Delete**.
  - Select the step in the workflow editor, and press the Delete key.

Working with Swimlanes

This section contains information about working with swimlanes in application workflows.

- About Swimlanes [page 331]
- Adding a Swimlane [page 332]
- Selecting a Swimlane [page 332]
- Renaming a Swimlane [page 332]
- Deleting a Swimlane [page 332]
- Moving States Between Swimlanes [page 333]

About Swimlanes

Swimlanes are a way to visually group states in an application workflow, so others can quickly understand the workflow design. A swimlane typically represents either the activities performed by the same role or participant, or a distinct phase of the process defined by the workflow.

For example, you could put all states that are associated with software development tasks into one swimlane. Another swimlane could include states that are associated with testing tasks. The **Any**, **Deleted**, and **Email** states that have no owner could be moved into an "unassigned" swimlane below or to the right of the other swimlanes.

Note the following points about swimlanes:

- Swimlanes are displayed in the workflow as horizontal or vertical bands. You can toggle between a horizontal or vertical orientation.
- The left and right edges of horizontal swimlanes, and the top and bottom edges of vertical swimlanes automatically cover the entire workflow.
- You can tailor the appearance of swimlanes by modifying their colors, labels, boundary lines, and so on.
- A swimlane encloses the states that are in its boundaries when you add the swimlane to the workflow. You then drag states among swimlanes to depict the process in the best way.
- A sub-workflow inherits swimlanes from its parent workflow.
- Swimlanes are included in the workflow diagrams that users can view in the SBM User Workspace.

Key Benefits

- Swimlanes are an efficient way to organize the activities or phases encompassed by an application workflow.
- Other designers and users can understand the workflow design at a glance.
- Numerous formatting options let you tailor the appearance of swimlanes to your needs.

Adding a Swimlane

You can add any number of swimlanes to an application workflow. The workflow editor expands as new swimlanes are added.

Important: Decide whether you want horizonal or vertical swimlanes before you arrange states and transitions within them. States and transitions do not automatically adjust to the new orientation, so you will otherwise need to rearrange them manually. For information about changing swimlane orientation, see Changing Swimlane Orientation [page 334].

To add a swimlane:

1. Drag a **Swimlane** item from the **Common Items** section of the **Workflow Palette**.
2. Alternatively, do one of the following:
   - Right-click an empty area of the workflow, and then select **Add New Swimlane**.
   - Right-click the label of an existing swimlane, and then select **Add New Swimlane**.

If no swimlanes exist, a horizontal swimlane is placed at the top of the workflow or a vertical swimlane is placed to the left of the workflow. If swimlanes already exist, the new swimlane is placed directly below the bottom swimlane or to the right of the rightmost swimlane.

Selecting a Swimlane

You need to select a swimlane before you can change its appearance, resize it, rename it, delete it, and so on. There are four ways to select a swimlane:

- Select the swimlane label.
- Select the swimlane in the drop-down list in the workflow or swimlane Property Editor.
- Select a swimlane, and then use the Tab key to select other swimlanes.
- Drag a rectangle around the swimlane label.

Renaming a Swimlane

You can rename swimlanes.

To rename a swimlane:

1. Click the swimlane label.
2. Right-click and select **Rename.**
3. Type the new name in the box that opens.

Deleting a Swimlane

When you delete a swimlane, the steps and transitions within it are moved to the swimlane that is above or to the left of it.

To delete a swimlane:

1. Click the label of a swimlane.
2. Right-click and select **Delete**.

Moving States Between Swimlanes

You can drag a state from one swimlane to another. States and transitions can also be placed on the line between two swimlanes, or in an area of the workflow not enclosed by a swimlane.

Changing Swimlane Styles

There is a default swimlane style in the styles editor. To view the style settings, click the **Styles** heading in App Explorer, and then select **Swimlane Style**. You can modify the default style in the styles editor, and add additional styles. For more information, see Customizing Styles [page 258].

The styles in the styles editor are included as items in the **Style** area on the **Appearance** tab of the Ribbon. When you select a style, its settings are applied to all swimlanes in the workflow.

Important: When you add a new swimlane, the background color and line settings you specified in the style editor are overridden. To apply these settings to new swimlanes, select the swimlane, and then select the desired style on the **Style** area on the **Appearance** tab.

The styles you modify or create in the styles editor initially affect all swimlanes in a workflow. You change the style of one or more swimlanes in a workflow while maintaining the default style for the other swimlanes.

To change the default style of a swimlane:

1. Click the swimlane label to select it.
2. Click the **Appearance** tab in the Ribbon.
3. To change the swimlane border, in the **Line** area of the Ribbon, select a style and width.
4. To change the text formatting in the swimlane label, in the **Font** area of the Ribbon, select the options you want to change.
5. If you want add or change the image displayed in the background of the image, do the following in the **Background** area of the Ribbon:
   a. Select or add a new image.
   b. Specify whether you want the image to repeat vertically, horizontally, or both.

Note: If you want to remove an image, select **(No image)**.

6. To change the alignment of the label text, select an icon in the **Alignment** area of the Ribbon.

Changing Swimlane Orientation

Swimlanes can run horizontally or vertically in an application workflow.

Important: Decide whether you want horizonal or vertical swimlanes before you arrange states and transitions within them. States and transitions do not automatically adjust to the new orientation, so you will otherwise need to rearrange them manually.

To change swimlane orientation:

1. Click the **Design** tab of the Ribbon.
2. In the **Swimlanes** area, click **Horizontal** or **Vertical**.

Changing Swimlane Order

You can change the order of swimlanes in an application workflow.

To change swimlane order:

1. Click the label of a swimlane.
2. Do one of the following:
   - Right-click and then select **Move Up** or **Move Down**.
   - On your keyboard, press the Ctrl key and then click the up arrow or down arrow key.

The swimlanes and the states and transitions within them are moved as you reorder them.

Resizing Swimlanes

Swimlanes can be resized. The workflow editor expands or contracts as needed to accommodate the new sizes.

The way you resize a swimlane determines whether the states and transitions in the swimlane move to an adjacent swimlane if necessary, or remain in place. The following table describes ways to resize a swimlane to achieve either result.

Result/Methods
Content of swimlane can move to an adjacent swimlane after the swimlane is resized. **Method 1:** 1. Select the swimlane you want to resize. 2. Press Shift+Up to decrease the size, or Shift+Down to increase the size. **Method 2:** 1. Position the mouse pointer over the line separating the swimlanes. 2. Drag the line to adjust the size of the swimlane.

Result/Methods
Content of swimlane does not move to an adjacent swimlane after the swimlane is resized. **Method 1:** 1. Right-click and position the mouse pointer over the line separating the swimlanes. 2. Drag the line to adjust the size of the swimlane. **Method 2:** 1. Press Ctrl and position the mouse pointer over the line separating the swimlanes. 2. Drag the line to adjust the size of the swimlane.

Working with Annotations

Annotations are notes that you can add to a workflow or sub-workflow. This section contains information about managing annotations in application workflows.

- Adding an Annotation [page 335]
- Customizing Annotations [page 336]
- Deleting an Annotation [page 337]

Adding an Annotation

You can add as many annotations as you need and place them anywhere in an application workflow or sub-workflow.

To add an annotation to a state:

- Do one of the following:
  - Drag **Annotation** from the **Common Items** section of the Workflow Palette [page 290], and drop it on the target state, which becomes highlighted.
  - Right-click the state, and select **Add Annotation**.

To add an annotation to a transition:

- Do one of the following:
  - Drag **Annotation** from the **Common Items** section of the Workflow Palette, and drop it on the target transition, which becomes highlighted.
  - Right-click the transition, and select **Add Annotation**.

To add an annotation to a workflow:

- Do one of the following:
  - Drag **Annotation** from the **Common Items** section of the **Workflow Palette**, and drop it on the workflow.

- Right-click the workflow background, and select **Add Annotation**.

Note: After you add one annotation to a workflow, additional annotations are placed directly on top of the first annotation. Drag the additional annotations to separate them from the first one.

Customizing Annotations

This topic describes how you can customize an annotation.

To move an annotation:

- Drag the annotation to another location.

To resize an annotation:

- Drag any of the points on its border.

To add a connector line:

1. Select the annotation.
2. In the annotation Property Editor, select **Draw connector line**.

To apply a system annotation style:

1. Select the **Styles** heading in App Explorer.
2. In the styles editor, under **Annotation Styles**, select a defined style.

To modify the background color:

1. Select the annotation.
2. In the **Background** area of the styles editor, click the paint bucket icon.
3. Do one of the following:
   - Select one of the "theme colors" or "standard colors."
   - Select the default color.
   - Select **Transparent**.
   - Click **More Colors**, select a color from the **Standard** or **Custom** tab, and click **OK**.

To add or remove a background image:

1. Select the annotation.
2. In the **Background** area of the styles editor, select one of the following options:
   - **(No image)**: Remove the current background image from the annotation.
   - **New image**: Locate an image, and click **Open** to make it available for selection.
   - ***Image filename***: Select a background image you have already "imported."

To modify the border style and width:

1. Select the annotation.
2. In the **Line** area of the styles editor, select the border style and width.

To edit the text:

1. Do one of the following:
   - Double-click the annotation.
   - Right-click the annotation, and select **Edit Text**.
2. Edit the text, and then press the Enter key to record your changes.

To modify the text styling:

1. Select the annotation.
2. In the **Font** area of styles editor, select the font name, size, and styling.

General Tab of the Annotation Property Editor

Use this tab to toggle display of the arrow pointing to the state that was selected when the annotation was added.

If no state was selected when the annotation was added, the arrow points to the nearest corner of the workflow background.

Note: If you move the annotation, the arrow shifts so that it always points to the nearest corner of the state or workflow background.

Element	Description
Draw connector line	Toggle display of the arrow pointing to the associated state, if any, or to the nearest corner of the workflow background.

Deleting an Annotation

You can delete any annotation from a workflow or sub-workflow.

To delete an annotation:

- Do one of the following:
  - Right-click the annotation, and select **Delete**.
  - Select the annotation, and press the Delete key.

Chapter 14: Defining Rules

The following topics describe how to define rules:

Introducing Rules

Rules are used in the context of *conditional routing* [page 682]. A rule is an expression or set of expressions that is evaluated when an application workflow reaches a decision. The workflow executes the outgoing transition associated with the first rule that evaluates as "true." If no rule evaluates as "true," the transition mapped to the "Otherwise" rule is executed. The "Otherwise" rule must be selected for one transition.

Important: The rule order is critical. You set the order on the Rules Tab of the Decision Property Editor [page 329].

Rules contain evaluation criteria that includes fields from the primary table, comparison operators appropriate for the field type, and values to compare. A value can be converted to an application variable. See About Conditional Routing [page 271] for the advantages of using application variables.

For conditional routing use cases, see Use Cases: Conditional Routing [page 271].

Rule Editor

The rule editor is primarily used to construct and edit rules that determine how to route an item from a decision in an application workflow.

Rule Palette

The **Rule Palette** includes the following types of controls:

- **Logical Operators** include AND and OR search operators. These operators are used to specify search operators for fields and to group expressions to set a sequence for evaluating rules.
- **Fields** include the fields that can be evaluated. These fields are in the *primary table* [page 692]. If you add or remove fields from the primary table, they are added or removed from this section.

Rule Expression

In the rule editor, drag fields and logical operators from the **Rule Palette** to the **Rule: *rule name*** block. Then specify the value, field, or application variable you want to

compare to complete the logic of the rule. The **Rule summary** block contains a read-only query string that represents the rule as you create it. The name of the rule is displayed on the tab in the rules editor.

Note: Only compatible field types can be evaluated. See Field-to-Field Comparisons [page 345] for a list of the field types that can be evaluated against the field you dragged from the **Rule Palette**.

Creating a Rule

To create a rule:

1. To create a rule for a decision, perform the following steps:

   a. Click the **Workflow Design** filter in App Explorer.

   b. Right-click the **Rules** heading, and select **Add New Rule**.

   Note: Alternatively, you can right-click in the rules list on the **Rules** tab of Rule Editor [page 339] and select **Add New Rule**, or on the **Rules** tab of the decision Property Editor, click the arrow next to the rule in the **Rule** column, and select **Create new rule**.

   Note: Alternatively, you can click **Element** in the area of the Ribbon, and select **Rule**.

   The two blocks described in the following table are displayed in the rule editor.

Section	Description
Rule: *rule name*	Lets you create expressions that define the rule.
Rule summary	Contains a read-only expression string that represents the rule as you create it.

2. Build an expression that the workflow will use to evaluate the rule. For instructions, see Creating Expressions for Rules [page 342].

3. On the **General** tab of the rule Property Editor, type a name and optional description for the rule.

Modifying Rule Properties

To modify rule properties:

- Select the rule in App Explorer (or click the **Rules** heading, and then double-click the rule in the list).

  The Property Editor below the rule editor lets you view and edit the name and description of the selected rule.

  Tip: If the Property Editor is not visible, select **Property Editor** in the **Common Views** area on the **Home** tab of the Ribbon.

General Tab of the Rule Property Editor

The following table describes the information and options that are displayed on the **General** tab of the rule Property Editor.

Element	Description
Name	The name of the rule that was provided when the rule was created.
Description	An optional description of the rule.

Creating Expressions for Rules

Rules provide comparison criteria that determine how an item is routed in an application workflow. A rule consists of one or more field expressions that are joined by logical operators. A field expression consists of a field, followed by a comparison operator, followed by a literal value, another field, or an application variable.

For example, to create a rule that evaluates whether the value of a *Numeric* field is equal to or more than 500 but less than 1000, create the following expression:

Amount >= 500

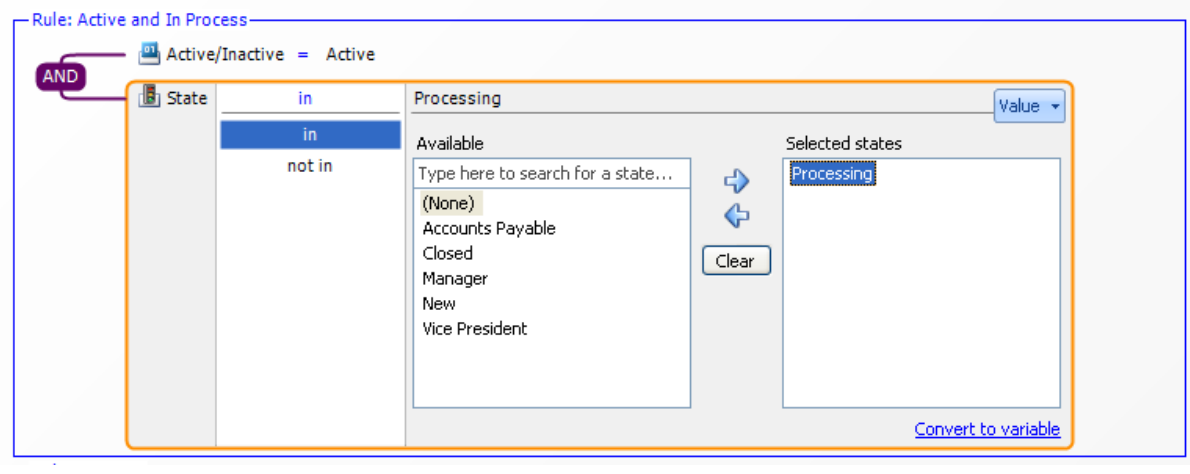
AND

Amount < 1000

The expression contains three panes:

- The left pane contains the field you want to evaluate.
- The middle pane contains the available comparison operators for the field type.
- The right pane contains a field value editor that lets you specify the value, field value, or application variable value that will be compared to the field.

For example:



To create expressions for rules:

1. Drag a field or logical operator from the **Rule Palette** and drop it in the **Rule: *name*** block.
2. If you added a logical operator, drag a field from the **Rule Palette** onto each branch of the operator. For a description of logical operators, see Rule Operators [page 346].
3. Comparison operators for most field types can be changed. Optionally, click the field and select another comparison operator from the popup window that opens. For a description of comparison operators, see Rule Operators [page 346].
4. Do one of the following to complete the expression.

- If you want to evaluate the value of the field, select **Value** from the drop-down list at the top right corner of the rule, and then select or type the value or values. For example, if you are working with an *Item Type* field, the types that are available as a selection for the field are displayed in the menu that opens. You can select a type from the list. Field values are described in Custom Fields [page 134].

- If you want to compare the value of this field to the value of another field, select **Field** from the drop-down list at the top-right corner of the rule, and then select the field.

  Note: Only compatible field types are available. For a list, see Field-to-Field Comparisons [page 345].

- If you want to compare the value of this field to the value of an application variable, select **Variable** from the drop-down list on the top right corner of the rule panel, and then select the variable.

  Note: Only those application variables that are compatible with the field are available.

5. To combine expressions and establish the order in which they are evaluated, drag an AND or OR logical operator from the **Rule Palette** onto the expression. This inserts the logical operator in the place where the expression was and moves the expression to the first child of the new operator. For more information about using the drag-and-drop operation in various scenarios, see Drag-and-Drop Behavior [page 193]. For examples of rule logic, see Rule Logic Examples [page 349].

   Tip: You can drag a field onto another field. This creates a new comparison expression that is "AND'd" with the existing one.

You can view a string that represents the rule in the **Rule summary** block.

Field Options and Values

The type of field that you use in an expression determines the options and values that are available to you. The following list describes common features of fields.

- **Current user** – All *User*, *Multi-User*, and *Multi-Group* fields have a special "Current user" field value. The expression evaluates to "true" when the user appears in that field. For example, add the *Submitter system field* [page 699] from the **Fields** section of the **Rule Palette**. Select the **in** operator, and then select the **Current user** check box.

  This expression will evaluate as true only if the "Current User" submitted the issue. In *Multi-Group* fields, the expression will evaluate to "true" if the current user is a member of one or more groups associated with the field.

- **None** – With certain field types, you can select the **None** check box. This selection evaluates all items meeting the rest of the rule criteria that do not have a value for the selected field.

- **Clear** – Most fields have a **Clear** button. Use this button to clear selected values or to return to the default value of **Undefined**.

- **Value, Field, Variable** – All fields have a drop-down list containing these selections. You can compare the field value to a specific value, the value of another field, or the value of an application variable. Fields and variables that are not supported in a given expression are not included as selections. (For a list of supported field types, see Field-to-Field Comparisons [page 345].)

Note: See Rule Operators [page 346] for a list of restrictions on comparisons used in expressions.

The following table lists field types and describes the options that are available to them when **Value** is selected in the drop-down list.

Field Type	Options
Selection/ User	If you can select values for a field, you can search for specific values or select multiple values to add to your condition. If you select a selection or user field, the values that are available as selections for the field are available in the **Available values** column. Use the right arrow button to move them to the **Selected values** column. In some cases, a text box contains **Type here to search for a value**. You can type a few letters in this box to filter the list of values. When you do this, the only values that are displayed are those that contain the letters you typed.
Date/ Time	To specify a date relative to the current date, select an option, such as **Start of Last Week** on the **Special value** tab. If you want to specify a certain number of days before or after the current date, select the **Now** checkbox on the **Special value** tab, select **+** or **-**, and type or select the number of days. For example, if you want to evaluate whether a Help Desk incident was submitted in the last 30 days, select the **≤** operator, select the **Now** check box, select **–**, and then type 30 in the **days** combo box. The expression becomes **Date ≤ 30 days ago**. To specify an exact date and time, click a date and select a time on the **Exact value** tab, and then click **Accept**.
Non-Selection	If a field does not let you select a specific value, type a value in the text box. For example, if you select a *Text* field, such as the *Description* field, type characters to search for. *Date/Time*, *Numeric*, and *Summation* fields also let you specify a value. In some cases, the text box contains **Enter a wildcard pattern**. For information about using wildcards, see Wildcards [page 191].
Binary	If you select a *Binary* field, such as the *Active/Inactive* field, select one value for your rule criteria.
Trinary	If you select a *Trinary* field, select one, two, or three values for your rule criteria.

Field-to-Field Comparisons

The following table shows the SBM field types that can be compared to each other in a rule expression.

Note: See Creating Expressions for Rules [page 342] for more information about field-to-field comparisons.

Field Type	Field Subtype	Compared Field Type
Binary/ Trinary	Binary, Trinary	Binary/Trinary with same subtype
Date/Time	Date and Time, Date Only, Elapsed Time, Time of Day	Date/Time with same subtype
Multi-Group	N/A	Multi-Group
Multi-User	N/A	Multi-User with same selection mode on **Options** tab of field Property Editor (**Individual users** or **Groups & users**)
Numeric	Fixed, Integer, Float	Numeric or Summation
Sub-Relational	N/A	Selected **Sub-field** on **Options** tab of field Property Editor
Summation	N/A	Summation or any Numeric field type
Text	Fixed Length, Memo, Journal	Text with any subtype
User	N/A	User, Multi-User, Multi-Group
Folder	N/A	Not supported
Multi-Relational	N/A	Not supported
Multi-Selection	N/A	Not supported
Project	N/A	Not supported
Single Relational	N/A	Not supported
Single Selection	N/A	Not supported

Field Type	Field Subtype	Compared Field Type
State	N/A	Not supported

Rule Operators

Logical operators let you set a sequence for evaluating conditions. *Comparison* operators let you select a function to compare the value of a field to a specific value, the value of another field, or the value of an application variable.

Restriction:

- In field-to-field comparisons, you can only select fields that are suitable to a given field type. For a list of supported field types, see Field-to-Field Comparisons [page 345].
- You can only select comparison operators that are suitable for a given field type. For example, you can select a **>** operator for a *Numeric* field, but not for a *Multi-User* field. For a list of supported operators, see Field Comparison Operators [page 348].

Logical Operators

The following table describes the AND and OR logical operators.

Operator	Description
AND	Drag **AND** from the **Logical Operators** section of the **Rule Palette** onto the expression. A rule using this operator evaluates as "true" when all expressions it joins evaluate to "true." For example, the condition "Amount > 100 " AND "Amount <= 500" evaluates whether the *Amount* field value is between 101 and 500.
OR	Drag **OR** from the **Logical Operators** section in the **Rule Palette** onto the condition. A rule using this operator evaluates as "true" when any expression it joins evaluates to "true." For example, the conditions "Owner in Joe Manager" OR "State in New" evaluates whether the item is in the "New" state *or* is owned by Joe Manager.

Comparison Operators

The following table describes the comparison operators.

Operator	Description
= (equal to)	An expression using this operator will evaluate to "true" if the value of the field on the left is the same as the value on the right.

Operator	Description
> (greater than)	An expression using this operator will evaluate to "true" if the value of the field on the left is greater than the value on the right.
< (less than)	An expression using this operator will evaluate to "true" if the value of the field on the left is less than the value on the right.
>= (greater than or equal to)	An expression using this operator will evaluate to "true" if the value of the field on the left is greater than or equal to the value on the right.
<= (less than or equal to)	An expression using this operator will evaluate to "true" if the value of the field on the left is less than or equal to the value on the right.
<> (not equal to)	An expression using this operator will evaluate to "true" if the value of the field on the left is not the same as the value on the left.
in	An expression using this operator will evaluate to "true" if the field on the left contains one or more of the values specified on the right.
not in	An expression using this operator will evaluate to "true" if the field on the left does not contain any of the values specified on the right.
contains all	An expression using this operator will evaluate to "true" if the field on the left contains all of the values specified on the right.
contains any	An expression using this operator will evaluate to "true" if the field on the left contains any of the values specified on the right.
does not contain all	An expression using this operator will evaluate to "true" if the field on the left does not contain all of the values specified on the right.
does not contain any	An expression using this operator will evaluate to "true" if the *Multi-Selection* or *Multi-User* field on the left does not contain any of the values specified on the right.
contains	An expression using this operator will evaluate to "true" if the field contains the specified text.
like	An expression using this operator will evaluate to "true" if the first field on the left matches the specified wildcard expression.
not contains	An expression using this operator will evaluate to "true" if the *Text* field on the left does not contain the specified text.

Operator	Description
not like	An expression using this operator will evaluate to "true" if the field on the left does not match the specified wildcard expression.
like (zero-filled)	An expression using this operator will evaluate to "true" if the field on the left matches the specified wildcard expression, including leading zeroes. For example, if there are leading zeroes on an Item ID number, they do not have to be specified for the expression to evaluate as "true."

Field Comparison Operators

The following table lists the comparison operators that are available for field-to-value comparisons and field-to-field comparisons in rule expressions.

Field Type	Field-to-Value Operators	Field-to-Field Operators
Binary/ Trinary	Binary: =; Trinary: in, not in	=, <>
Date/Time	=, <>, >, >=, <, <=	Same as "Field-to-Value Operators"
Multi-Group	contains any, does not contain any	Same as "Field-to-Value Operators"
Multi-Selection	contains all, does not contain all, contains any, does not contain any	Not supported
Multi-User	contains any, does not contain any	Same as "Field-to-Value Operators"
Numeric	=, <>, >, >=, <, <=	Same as "Field-to-Value Operators"
Single Selection	in, not in	Not supported
State	in, not in	Not supported
Sub-Relational	All operators applicable to the referenced *Sub-Relational* field type.	Same as "Field-to-Value Operators"
Summation	=, <>, >, >=, <, <=	Same as "Field-to-Value Operators"
Text	contains, not contains, like, not like	=, <>

Field Type	Field-to-Value Operators	Field-to-Field Operators
User	in, not in	Same as "Field-to-Value Operators"

Wildcards

You can use wildcard characters expressions for rules if **Enter a wildcard pattern** is in the text box.

The following guidelines apply to wildcard characters:

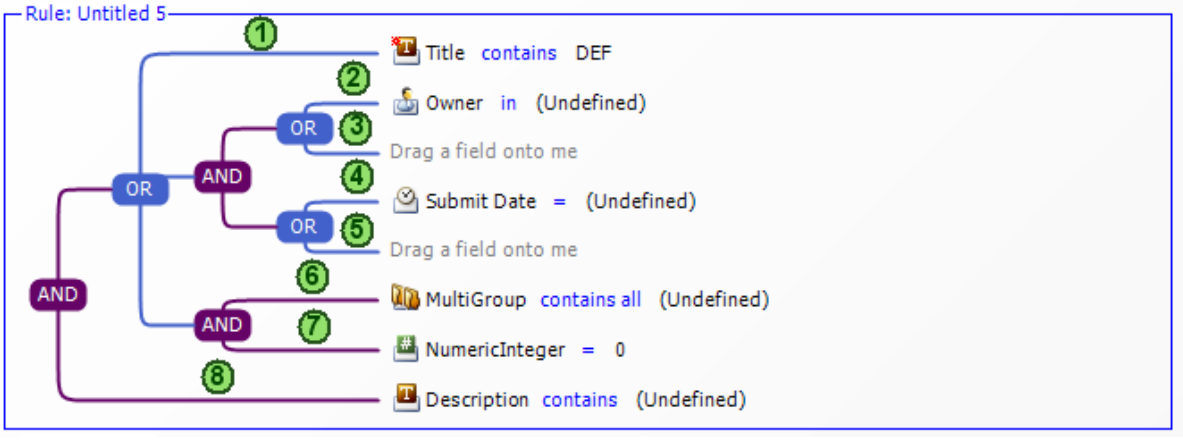
- Asterisks (*) and percent signs (%) serve as wildcard characters. A wildcard character matches zero or more consecutive characters.
- Underscores (_) match a single character.
- Wildcard characters are automatically applied to the beginning and end of your criteria. Items containing all of your criteria are returned.
- You can override automatic wildcards by including at least one wildcard in your criteria. Wildcard characters can be placed anywhere in the box and could return different results depending on the location of the wildcard character. For example, *ed returns all items ending in ed, while ed* returns all items beginning with ed.
- To find all items, type a wildcard character (* or %) or leave the box empty.
- Leading and trailing spaces are removed.

Rule Logic Examples

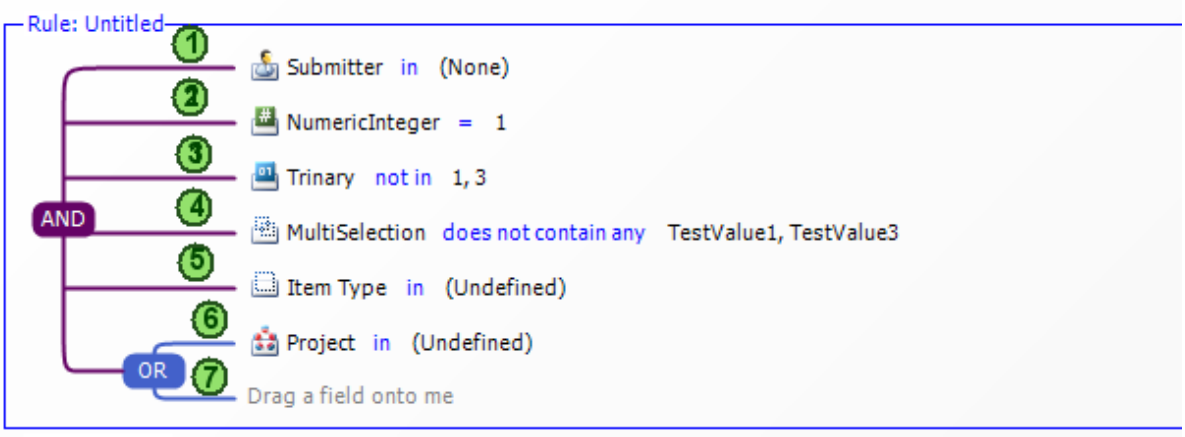
The following examples illustrate how rules are constructed.

Note: AND and OR operators always have at least two child nodes. The nodes are either fields or **Drag a field onto me** placeholders.

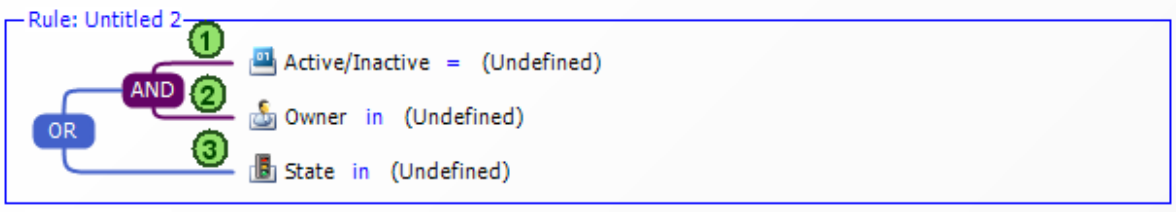
(1 or ((2 or 3) and (4 or 5)) or (6 and 7)) and 8



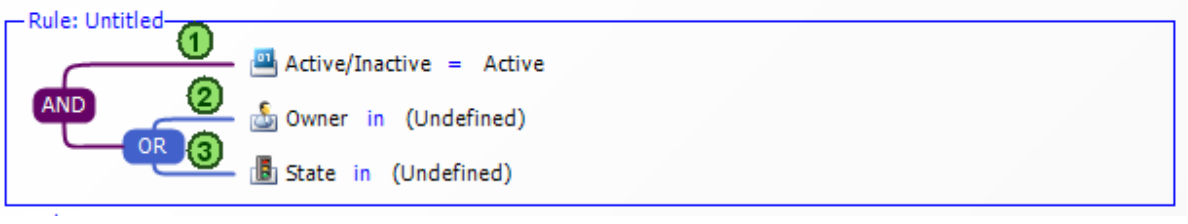
1 and 2 and 3 and 4 and 5 and (6 or 7)



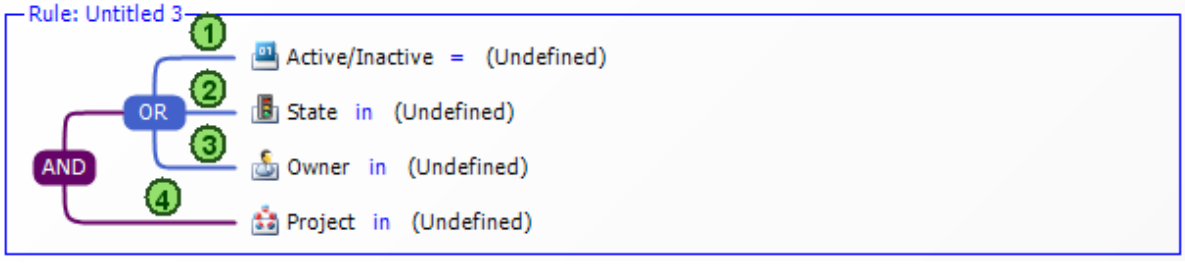
(1 and 2) or 3



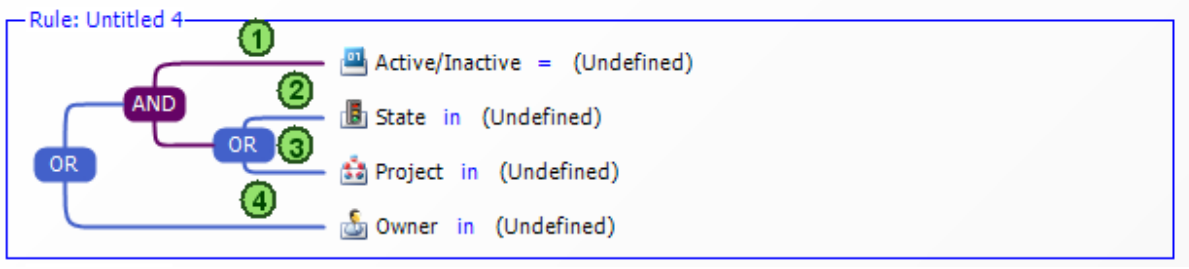
1 and (2 or 3)



(1 or 2 or 3) and 4



(1 and (2 or 3)) or 4



Drag-and-Drop Behavior

The following points describe how to use the drag-and-drop operation to add fields and operators to a rule expression.

Drag-and-Drop Operation	Behavior
AND to AND	To add a new expression to an AND group, drag a field from the palette onto another field in the group. The new field is added above the field onto which you dragged.
OR to AND	To add an OR group to an AND group, drag the **OR** operator to one of the lines extending from the AND group. If there is a field on the top line, the OR group is added to the bottom line of the AND group. If no field is on the top line, the OR group is added to the top line of the AND group.
OR to OR	To add a new expression to an OR group, drag a field from the palette onto another field in the group. The new field is added above the field onto which you dragged.
AND to OR	To add an AND group to an OR group, drag the **AND** operator to one of the lines extending from the OR group. If there is a field on the top line, the AND group is added to the bottom line of the OR group. If no field is on the top line, the AND group is added to the top line of the OR group.
Existing Field	To move an existing field within an AND or OR group, drag it to another field. The existing field is placed above the field onto which you dragged it.

Chapter 15: Defining Application Variables

The following topics describe how to add and modify application variables:

Introducing Application Variables

An application variable stores a part of a rule expression. Each application variable is associated with a single primary table field. In the Rule Editor [page 339], you drag application variables from the **Rule Palette** to the rule expression.

Application variables provide the following benefits:

- Designers can assign a default value to an application variable once, and then reuse the application variable in multiple rules in the process app.
- Designers can change the default value of an application variable, and all rules that reference that variable will use the new default value.

Restriction: You cannot associate an auxiliary table field with an application variable.

Note: For information about rules, see Introducing Rules [page 339]

Application Variable Editor

Use the application variable editor to add application variables and specify the default values of new or existing application variables.

To open the application variable editor, click the **All Items** filter in App Explorer, and select the **Application Variables** heading.

The left pane of the editor contains a list of the application variables and their current default values. The right pane of the editor is where you assign or modify the default values.

Element	Description
Name	The name of the application variable.
Description	A description of the application variable.

Element	Description
Field	The field associated with the application variable.
Data Type	Displays the data type of the selected field.
Value	The default value that will be stored in the application variable. If you want to change the value, type it in the box below the value. To change what you typed, click **Clear**. A hint in the box shows you the format of the value (based on the field type and its options), or whether you can enter wildcard characters. See Wildcards [page 349] for wildcard guidelines.

Creating an Application Variable

To create an application variable:

1. Do one of the following:
   a. Right-click the left pane of the Application Variable Editor [page 353] and then select **Add New Field Application Variable**.
   b. In the Rule Editor [page 339], click the field for which you want to create the variable, and then click **Convert to variable**.
2. Complete the fields in the right pane of the Application Variable Editor [page 353] to specify a default value for the variable.
3. Click in an empty area of the right pane to refresh the left pane of the application variable editor.

Note: The application variables are ordered by value under headings in the left pane. For example, all application variables with an "Undefined" value are listed under the **(Undefined)** heading.

Chapter 16: About Inheritance and Overrides

Application workflows are defined in SBM Composer and determine how work efforts, or primary items, are tracked as they move from state to state. Projects are configured in SBM System Administrator, and let you organize groups of primary items in a way that makes sense for your workflow. For example, you can create a project for each team working on a product, or for each version of the product. You can then configure the project by enabling roles for the project or assigning privileges to users and groups.

Sub-workflows are children of parent workflows, and sub-projects are children of parent projects. The following sections describe how sub-workflows inherit elements from parent workflows, how sub-projects inherit elements from parent projects, and how projects and sub-projects inherit elements from the workflows that are assigned to them. It also describes how you can override settings for the inherited elements, making them independent of the parent workflow, parent project, or assigned workflow.

Inheritance

Inheritance rules [page 687] make it easy to configure your projects. Inheritance lets you create an *application workflow* [page 680] in SBM Composer that can be used by all projects. You can assign the workflow to a parent project in SBM System Administrator and by default, all sub-projects of the parent project automatically use the same workflow. This lets you quickly create a project hierarchy, without having to create a workflow for each project. When a workflow is assigned to a project, the elements of that workflow (such as fields, states, and transitions) are also assigned to the project. These elements and their properties are inherited throughout the project hierarchy.

Overrides

You use *overrides* [page 690] to change inherited elements of workflows and projects. In SBM System Administrator, you can use overrides to customize projects by making changes to specific properties of fields, states, and transitions in projects or sub-projects. You can also change the workflow that is assigned to a project or sub-project. In SBM Composer, you can use overrides to change inherited elements in sub-workflows.

The following list contains examples of using overrides.

- You have an *application* [page 679] that handles time-off requests. One sub-workflow in the application is used for vacation time, and another is used for sick time. The vacation sub-workflow needs a **Vacation Time Used** field, and the sick time sub-workflow needs a **Sick Time Used** field. You specify different forms for the

states and transitions in each of the two sub-workflows to accommodate the different fields.

- A sub-workflow could include a transition that was inherited from its parent workflow but is not needed in the sub-workflow. You can edit the sub-workflow and disable the transition.
- You have a workflow that handles requests that come into the Support organization, and make **Mark** (the Support manager) the default value for the **Support Lead** field. After you deploy the process app, you could create sub-projects for each product that the organization supports, and change the default value for the **Support Lead** field to the support lead for each product.
- You want to make sure that users always provide a value for a **Customer Name** field in an Issue and Defect Management project that customers use to report issues. You can set the field as required for that project, but not for a sub-project that employees use to report issues.
- You want to assign a manager named Susan as a default value for a **Manager** field in a Human Resources sub-project. This ensures that **Susan** is always selected when users update items in the project. In a Finance sub-project, you would assign the Finance manager as the default value, and so on.

You can override the following workflow attributes in SBM Composer:

- Transition forms
- State forms
- Field properties
- Field privileges for transitions
- Field privileges for states
- Transition ordering for sub-workflows

The following table shows the override tasks you can perform in SBM System Administrator.

Task Type	Tasks
Project	You can override the following project attributes: • Default state forms • Default transition forms • Default project, state, and *transition field* [page 700] ordering when quick forms are used • Field attributes for projects and transition fields • Display options for fields in projects and for transition fields (except *Binary/Trinary*, *Date/Time*, *Numeric*, and *Text* fields) • Dependent field selections for independent *Single Selection* fields • Dependent field selections for *User* fields
State	You can override the following state attributes: • Inherited form for states in projects • Default field ordering for states in projects when quick forms are used • Transition button ordering for states in projects
Transition	You can override the following transition attributes: • Transition authentication settings for transitions in projects • Inherited form for transitions in projects • Project settings for Post, Subtask, and Copy transitions • Default field ordering for transitions in projects when quick forms are used • Calculations for *Date/Time* and *Numeric* fields in projects • Transition button ordering • Default values for transition fields

Note: For information about how to apply overrides, see the *SBM Composer Guide* or online Help, and the *SBM System Administrator Guide* or online Help.

Behavior and Usage

This section contains important information about using inheritance and *overrides* [page 690].

- Each *application* [page 679] can have multiple workflows, which may or may not be related through inheritance. Workflows generally define behavior, and projects generally control access to SBM items.
- A sub-workflow inherits the fields, states, transitions, and forms in its parent workflow. You typically define core behavior in a parent workflow and then use overrides to define more specialized behavior in its sub-workflows.
- A duplicated (also known as sibling) workflow inherits the fields, states, transitions, and forms in the original workflow.
- A project inherits the settings established for the *application workflow* [page 680] to which it is assigned.
- The hierarchical arrangement of projects supports inheritance. For example, you can apply group or user privileges for a parent project, and the sub-projects inherit those settings. This reduces the effort it takes to maintain privileges. You can override settings at any level in the hierarchy.
- The *workflow hierarchy structure* [page 701] is a representation of workflows and sub-workflows, and is designed in SBM Composer. In SBM System Administrator, the *Base Workflow* [page 681] serves as the root of your system's workflow hierarchy. It cannot be edited, deleted, or assigned to any project.
- The *Base Project* [page 681] serves as the root of your system's project hierarchy. The Base Project cannot be deleted, but it can be renamed to suit your company's needs. When a new project is created, it is always derived from an existing project or from the Base Project. Any new project is a sub-project of the Base Project. The Base Project cannot be assigned a workflow or contain primary items; it can only contain sub-projects.
- You can move projects in the hierarchy; however, if you make a sub-project a parent project, it stops inheriting from the parent project.
- Field selections for *User*, *Multi-User*, and *Multi-Group* fields in workflows and auxiliary tables are defined in SBM System Administrator, because selections for these fields are application configuration elements that are not available in SBM Composer. Roles assigned to the fields in SBM Composer are listed as "Acting as: <role name>." You cannot enable or disable these selections in SBM System Administrator.
- Each workflow and project contains a list of default fields. For workflows, the default fields list is determined by the *primary table* [page 692] on which the workflow is based. For projects, the default fields list is determined by the workflow to which the project is assigned. Each field contains a set of properties defining how the field looks and acts to users in the SBM User Workspace. You can modify fields in workflows to override these inherited field properties as needed. You make most of these modifications in SBM Composer; however, some modifications can be made in SBM System Administrator.
- When you delete a project, all primary items associated with that project are also deleted, along with the *change history* [page 681] for those items. Overrides and settings associated with the project, such as field properties and selections are deleted, as well as user and group privileges associated with the project.

- The *mass update* [page 689] feature lets users simultaneously transition, update, or delete multiple primary items and simultaneously update or delete multiple *auxiliary table* [page 680] items. You set the mass update option in SBM Composer. In SBM System Administrator, you can override this setting for default fields in projects and for transition fields.

Remember: Projects inherit from workflows. However, changes you make in a workflow are not reflected in the SBM User Workspace if the project does not inherit from the workflow (that is, if it has settings that override the workflow). A transition override at the project level has the final authority regarding what users see in the SBM User Workspace. For example, if you set a default value for a *Manager* selection field in a workflow, that value might not appear in the SBM User Workspace if the project has its own override for that field, or if the transition in a project has its own override for the field.

Best Practices

This section contains suggestions for getting the best results when you use inheritance and *overrides* [page 690] in your workflows and projects.

- Because overrides can be difficult to maintain, it is best to use them only when absolutely necessary.
- You can override authentication options for transitions in sub-workflows and projects. For best results, set authentication options for a parent workflow, and then enable or disable them for sub-workflows and projects as needed.
- Copy transitions copy all field values from the source item to the next item (except for the *Submit Date* field, which is populated with the date the item was copied). If you want specific field values set during the Copy transition, use the *transition field* [page 700] overrides in the destination project to set or clear the affected fields.

Chapter 17: Working with References

This section contains the following information:

About References

You create references to make an association in your process app to a table, field, or other design element in an application contained in another process app. You cannot create a reference to such a design element unless you first create a reference to the application that contains the design element. This topic contains examples of typical ways in which you can use references.

Example: Using a Post Transition

A tester submits a ticket to engineering, requesting an enhancement to a feature. This ticket is assigned to an engineer, who starts making code changes. The engineer realizes that the enhancement requires documentation changes, so he posts it to the documentation department. A new ticket is created for documentation, and a writer is assigned to it.

In this example, one process app handles requests to the engineering department. The other process app handles requests to the documentation department. In the *application workflow* [page 680] for the engineering process app, you use a **Post** transition to create a reference between the referencing process app (engineering) to the referenced process app (documentation).

Example: Using a Subtask Transition

A company has a new hire, and the IT department needs to prepare for the employee's first day of work. The preparation includes assigning a cubicle for the employee. The hiring manager submits an issue to the employee onboarding application, and the issue is assigned to the IT manager. The IT manager assigns the issue to a system administrator, who posts it to the facilities department to handle the cubicle assignment. After the cubicle is assigned, the issue is transitioned to the **Cubicle Assigned** state and reassigned to the system administrator.

In this example, one process app handles the overall employee onboarding process. Another process app handles the facility department processes. In the application workflow for the employee onboarding process app, you use a **Subtask** transition from a **Create Network Account** state to a **Cubicle Assigned** state. The **Subtask** transition creates a reference between the referencing process app (employee onboarding) to the referenced process app (facilities).

Example: Using a Relational Field

A customer calls Technical Support to report that a feature does not work as expected. The technical support representative submits an issue to a defect management application.

The engineering manager realizes that the solution is complex, and schedules it to be implemented during the next release. In the meantime, the technical support engineer submits an issue to a knowledge base application. This issue describes the problem and provides a temporary solution.

In this example, the issue defect management application and knowledge base applications are in separate process apps. A *Single-Relational* field called "Associated Issue" is on the submit form in the knowledge base application. This field references the Issues table in the defect management application. When the technical support engineer submits the knowledge base issue, he selects the customer-reported issue from a list of issues.

Example: Using the Global Process App

A customer support representative is working with Acme Corporation, who reported a problem with a product. The customer support representative submits an issue to the incident management application on behalf of Acme Corporation. On the *transition form* [page 700], the customer support representative selects **Acme Corporation** from the **Requestor** drop-down list.

In this example, the one process app handles the incident management process. The *Global Process App* [page 687] contains a Company *auxiliary table* [page 680], which includes a *Company* field. You create a reference to the Global Process App by mapping the *Requestor* field in the incident management process app to the Company auxiliary table in the Global Process App.

Important: You can create references to fields in Global Process App tables, but you cannot create references from tables in the Global Process App.

About Design Numbers

Every time you create a new process app from AppCentral™ or create a new process app based on a blueprint (`.msd`) file, the process app gets a unique internal identity. This way, if two designers in the same *environment* [page 684] create two process apps from the

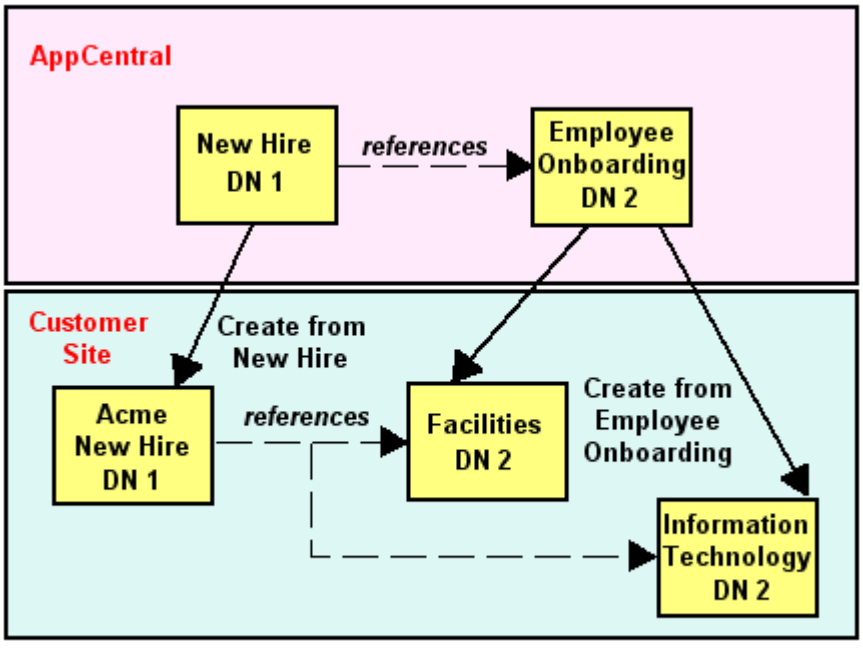
same process app, they will not overwrite each other when their process apps are deployed.

Note: To create a new process app, click **New** on the Composer menu. In the **Create New Process App** dialog box, select a process app from the **AppCentral™** area or click **Browse** from the **Available Templates** area and then navigate to the blueprint file you want to use.

When they are first created, SBM Composer assigns each *application* [page 679] and *orchestration* [page 690] in the process apps a *design number* that represents the *design* identity of the process apps. The design number is the same design number as the applications and orchestrations in the original AppCentral™ process app or blueprint file. This is because the two new process apps share the same purpose as the original process app.

Normally, you do not need to pay attention to the design number. However, it becomes important when an application in one process app references an application in another process app. In this case, the reference contains not only the internal identity of the referenced application, but also the design number of the referenced application. This permits the reference to continue to be valid, even though the internal identity was changed when the process app was created from AppCentral™ or the blueprint file. When there is ambiguity about which application was referenced, SBM Composer uses design numbers to present you with a choice of those applications the referencing application was designed to work with.

Suppose you downloaded the New Hire and Employee Onboarding process apps from AppCentral™. The New Hire process app includes the New Hire application and the Employee Onboarding process app includes the Employee Onboarding application. The New Hire application references the Employee Onboarding application.



DN = Design Number

You create a new process app based on the New Hire process app and call it Acme New Hire, and create a new process app based on the Employee Onboarding process app and

call it Facilities. Someone else creates a process app based on Employee Onboarding and calls it Information Technology. The new process apps include the Acme New Hire, Facilities, and Information Technology applications, respectively. The Acme New Hire application references both the Facilities and the Information Technology applications. The Facilities and Information Technology applications could be modified, but they still represent the same general function.

You now need to resolve a reference from the Acme New Hire application. When the original New Hire process app was designed, the New Hire application referenced the Employee Onboarding application. You need to know which applications in your SBM Application Repository are likely candidates to resolve the reference.

Although the Facilities and Information Technology applications are separate applications with unique internal identities, they share a common design number. This is because the internal identifiers that identify an application and other design elements is completely changed when a process app is downloaded from AppCentral™, but the design number of an application remains the same. In the **Resolve Reference to Application** dialog box, applications with the same design number as the referenced application are displayed in bold. This makes it easier for you to select compatible applications. For more information about the **Resolve Reference to Application** dialog box, see Resolve Reference to Application Dialog Box [page 641].

If process apps with the same design number diverge in purpose and content such that they cannot still be considered related, you can reset the design numbers of their applications. To do this, select the process app in App Explorer, and then in the process app editor, click the **Design numbers** tab, and then click **Reset design numbers**.

Defining a Reference

To define a reference:

1. In App Explorer, right-click the **References** heading or the name of an existing reference, and select **Add Application Reference**. The **Add Application Reference** dialog box opens.
2. Select **Local Cache** or **SBM Application Repository** to indicate whether the process app you want to reference is stored in your computer's file system or checked in to the SBM Application Repository from SBM Composer.
3. Select the *process app* [page 692] that contains the *application* [page 679] that you want to reference, and then click **Add**.

   Note: To define a reference, the referenced process app must have been checked in. For example, if you want to create a new process app and create a reference to the *Global Process App* [page 687], you must open the Global Process App in SBM Composer and then check it back in before you can see it in the SBM Application Repository and reference it. You must also have the "View" privilege (set in Application Administrator) for the process app you want to reference if the process app is in the SBM Application Repository, not in the *Local Cache* [page 688].

   CAUTION: If you add a reference to an application in the open process app, the application becomes inaccessible.

4. Create references to the referenced application. For example:

- For a **Post** transition, on the **Post Options** tab of the transition Property Editor, select the referenced application in the **Post application** list. If there are multiple tables in the referenced application, select the table you want to reference in the **Post table** list.
- For a *Multi-Relational* or *Single Relational* field, on the **Options** tab of the field Property Editor, select the referenced application in the **Application** list. If there are multiple tables in the referenced application, select the table you want to reference in the **Table** list.

Viewing References

To view a list of *application* [page 679] references:

1. Do one of the following:
   - Click **+** next to the **References** heading to expand the list in App Explorer.
   - Select the **References** heading to display the references list on the **References** tab to the right.
2. If you want to see details about a referenced application, double-click the application in either list to open the application editor.

Refreshing References

If a design element in a referenced application changes after the reference is defined, you need to "refresh" the reference definition in your process app. Subsequent validation shows whether the changed design element will still serve the intended purpose in your process app. Refreshing a reference has an effect similar to that of the **Get Latest of All** command.

Refreshing a Reference Definition

To refresh a single reference definition:

1. In App Explorer, under the **References** heading, right-click the *application* [page 679] name.
2. Select **Refresh**.

Refreshing All Reference Definitions

To refresh all reference definitions:

In App Explorer, right-click the **References** heading, and select **Refresh All References**.

Note: You can use the **Refresh All References** command if there is at least one reference to a process app in the SBM Application Repository.

Removing References

You cannot remove an application reference if it is used by any *design element* [page 683] references.

To remove an *application* [page 679] reference:

1. In App Explorer, under the **References** heading, right-click the application name.
2. Select **Remove**.

The application reference is removed if it is not used by any design element references. If it is used by design element references, a dialog box opens that lists all affected design elements. Eliminate all such design element references, and then remove the application reference.

Examining References

To examine a referenced *application* [page 679]:

In App Explorer, under the **References** heading, select the referenced application. Information about the selected application is displayed in the application editor.

Modifying a Referenced Application

To modify a referenced *application* [page 679]:

1. In App Explorer, right-click the **References** heading, and then select **Open Related Process App**.
2. Respond to prompts regarding saving changes to the open process app and checking in design elements.

About Resolving References

This topic describes some use cases for resolving references.

Important: When you *import* a set of process apps (by pointing to **Import and Export** from the Composer menu and then selecting **Import from File**), the process apps keep the same internal identifiers and the references will be preserved. When you *create* a set of process apps (using **New** from the Composer menu), the process apps get new internal identifiers and you must resolve the references explicitly.

Use Case: Two Process Apps from AppCentral

Some process apps from AppCentral™ are designed to work together. Such process apps contain references to each other, such as a **Post** transition that creates an item in a target *process app* [page 692], *application* [page 679], or *workflow* [page 701] when the transition fires.

You use the **New** command in the Composer menu to create two new process apps based on two process apps from AppCentral™. When you do this, all internal identifiers in the process apps are regenerated. This is necessary because each process app can be downloaded multiple times to be modified for different purposes. The process apps and their parts must have unique internal identifiers so they can appear as unique process apps when they are deployed.

When the internal identifiers in the process app are regenerated, the references between the process apps become unresolved. This is because the referencing process app contains an internal identifier to something in the referenced process app, but all of the internal identifiers in the referenced process app changed.

To make resolving references easier, a different type of identifier, the design number, is applied to all referenceable parts. The design number makes it possible for SBM Composer to provide a list of parts that have the same design number as the original referenced part. For example, at the *application* [page 679] level, the applications that have the same design number as the original application are listed. Such applications are displayed in bold in the Resolve Reference to Application Dialog Box [page 641].

Note: For information about design numbers, see About Design Numbers [page 362].

Use Case: Two Existing Process Apps

You have an existing process app with a reference to a field in another existing process app. The field in the referenced process app was deleted, so the reference becomes invalid. When you try to deploy the referencing process app, you get an error message about the missing field.

When you double-click the error message, the Property Editor for the missing field is displayed. In the Property Editor, you can select another field to use as a reference.

Use Case: An Existing Process App and a Process App from AppCentral

You have an existing process app and want it to work with a process app you download from AppCentral™. For example, both process apps handle change requests, but the process app from AppCentral™ has a reference to a field in a Customer table in another process app that it was designed to work with. You need to change your existing process app so it can work with the downloaded process app and access customer data. The process app from AppCentral™ has references that need to be resolved to the existing process app. However, unlike Use Case 1, the existing process app has no design numbers in common with the process app from AppCentral™.

To make resolving references easier in this situation, the reference contains the last known name of the referenced item. For example, the downloaded process app contains a reference to a customer table. You can see the name of the table in the Property Editor for a *Relational* field, *Sub-Relational* field, or **Post** transition. You can resolve the reference to an existing customer table in your database, or to a new customer table that you create.

Use Case 4: An Updated Process App in AppCentral

Some time ago, you created two process apps based on process apps in AppCentral™ that were designed to work together. Serena updated one of the process apps, and you want to get the latest version. You delete the outdated process app from the SBM Application Repository because the updated process app has the same name, and you cannot check in a process app with the same name as an existing process app. Then you create a process app based on the updated process app.

Because the updated process app has new internal identifiers, the references between your process apps become unresolved. However, the design numbers of referenceable parts in original process app are maintained in the updated process app. The applications with the same design numbers are displayed in bold in the Resolve Reference to Application Dialog Box [page 641], making it easier to resolve the references.

Use Case 5: Ignoring Unresolved References

You create a new process app based on a process app in AppCentral™. This process app is meant to work with another process app in AppCentral™, so it contains unresolved references.

You do not need the other process app now, so you decide to ignore the unresolved references. To do this, right-click the unresolved reference in App Explorer and select **Ignore Reference**. The reference and any design elements in the referenced application are marked through, and when you validate the process app, no unresolved reference warnings are displayed in the Message List.

Resolving References

This topic explains how to resolve an unresolved reference to a *design element* [page 683] in another process app. When you validate a process app, a list of unresolved references, if any, is displayed in the Message List at the bottom of the SBM Composer window.

Tip: In many cases, the process app containing a referenced *application* [page 679] is missing from the SBM Application Repository (that is, the referenced application existed when the reference was created, but was later deleted). After you resolve the application reference, the references to other design elements in the referenced application are usually resolved automatically.

Note: You might not need to resolve references. See Ignoring Unresolved References [page 369] for more information.

Unresolved references are displayed with a red X. The following illustration shows an unresolved application reference as it appears in App Explorer.

To resolve an unresolved reference:

1. After validating the process app, double-click the message in the Message List or right-click the message and select **Go to Location**. SBM Composer takes you to the location where you can resolve the reference. For example, for an unresolved application reference, SBM Composer takes you to the application editor. For an unresolved table reference, SBM Composer takes you to the Property Editor for the field in the referenced table.
2. If this is an unresolved reference to an application, perform the following steps:
   a. Right-click the reference in App Explorer and select **Resolve**, or click the message at the top of the application editor. The **Resolve Reference to Application** dialog box opens.
   b. Complete the dialog box as described in Resolve Reference to Application Dialog Box [page 641].
   c. Validate the process app again.

3. Resolve any remaining resolved references as appropriate. For example, if a referenced field is missing from a table, either restore the field in the table, or in the Property Editor for the field, select another application and field that resolves the reference.

Changing a Reference Resolution

You can select a resolved application reference and change it to another *application* [page 679]. When you do this, all element references (such as *relational fields* [page 694], **Post** transition options, and **Subtask** transition options) are automatically resolved.

To change a reference:

1. Right-click the reference under the **References** heading in App Explorer—**All Items** or App Explorer—**References**, and select **Change**. The **Change Reference** dialog box opens.
2. Complete the dialog box as described in Change Reference Dialog Box [page 622].

Ignoring Unresolved References

You might not need to resolve a reference to something in another process app. For example:

- You do not need the other process app.
- Your process app runs correctly, even though it is missing some unneeded functionality.
- You want to see if the process app is suitable before you make the reference.

To ignore an unresolved reference:

1. Right-click the reference in App Explorer and select **Ignore Reference**. The reference is marked through and grayed out.
2. All references to design elements in the referenced *application* [page 679] should also be ignored. Click **Validate** and make sure that all messages have cleared from the Message List.

Chapter 18: About Actions

You can define an *action* [page 679] that executes when an initiating item is transitioned from one state to another in your workflow.

The following table describes the action types.

Action Type	Description
Transition	Causes a transition to be executed when the workflow reaches a transition or state on which the action is defined. For example, you could use a transition action to transition the subtasks for an item based on the value of a *Binary/Trinary* or *Single Selection* field. For more information, see Chapter 19: Working with Transition Actions [page 381].
Trigger	A trigger action can automatically transition associated primary items as they move through a workflow. Associated items can be relational field selections, principal or subtask relationships, or item link attachments. For example, you could have two items that are related but stored in separate projects. The two items can be linked to each other and transitioned together as they move through the workflow. When one item is transitioned, any other items that are linked to the current item are also transitioned.
Script	Causes a script to be run when the workflow reaches a transition or state on which the action is defined. For example, you could have a script that copies the value of a *Title* field to a *Description* field when a Submit transition is executed.
Web Service	Causes a Web service to be invoked when the workflow reaches a transition or state on which the action is defined. For example, you could use a Web service operation to create items in an auxiliary table when the workflow reaches a transition or state on which the action is defined.
Orchestration Workflow	Causes an orchestration workflow to be invoked when the workflow reaches a transition or state on which an action is defined. For example, you could invoke an orchestration workflow that creates a set of items based on the values a user selects from a *Multi-Relational* field. The action is defined on the transition that is executed after the user selects the items.

Restriction: Do not use "Transition," "Orchestration Workflow (continue executing)," or "Trigger" transition types on an incoming transition to a decision.

This section contains the following information:

- Considerations for Using Actions [page 373]
- Action Wizard [page 374]

Considerations for Using Actions

Consider the following when you set up actions:

- Actions can be performed on items transitioned manually by users, by other actions, or by mass updates.
- Actions must be defined at the level at which a state or transition was created and cannot be overridden for a sub-workflow or project.
- All actions must conform to the workflow. In other words, you cannot use an *action* [page 679] to execute a transition that is not already defined in the workflow.
- Actions can be defined for all transition types, but only **Regular**, **Copy**, **Post**, and **Subtask** transitions can be executed as a result of an action.
- To ensure that actions execute properly, make sure there are no required fields for the transition that will execute as a result of an action. If necessary, provide default values for data in fields required for any transition that contains an action.
- You can define multiple actions for a transition. To the extent possible, actions are executed in the order in which they are listed on the **Actions** tab of the Property Editor for the selected transition or state. Use **Move up** and **Move down** to reorder actions as needed.
- If actions are used to execute multiple transitions on an item, *Item Type* [page 688] restrictions may prevent subsequent transitions from executing.
- If you have multiple transition actions, the first one that evaluates as true prevents the evaluation of further actions.
- If you have an orchestration action, it must be last (is not orderable with the other actions), and the *orchestration workflow* [page 690] is not called if one of the previous actions is true. If the orchestration workflow must always run, the setup must be redesigned to account for that (for example, have the orchestration action on the transition, and find a way to move the transition actions to the next state).
- When a transition action is a **Post** or **Subtask** transition and you do not specify a post-item project and only one valid project is available, the submit defaults to the valid project. Otherwise, the submit does not occur and an error is generated.
- All actions are processed at the end of the transition process.
- If an action fails, an error is recorded in the Event Viewer and displayed to users in the **Item Details** frame for the item that initiated the action.
- Do not use "transition," "asynchronous orchestration workflow," or "trigger" transition types on an incoming transition to a decision.
- The SBM Application Engine executes Web service functions, SBM AppScripts, transition attribute scripts, transition actions and state actions, and events, in the following order:

  1. Web service function for the pre-transition context
  2. SBM AppScript for the pre-transition context

3. Transition attribute scripts for the pre-transition context
4. Transition executed by users
5. SBM AppScript for the post-transition context
6. Transition attribute scripts for the post-transition context
7. Web service function for the post-transition context
8. SBM AppScript for the post-state context
9. Web service function for the post-state context
10. SBM AppScript for the pre-state context
11. Web service function for the pre-state context
12. Transition completed and recorded in the database
13. Transition actions
14. Events are emitted
15. Subtasks and posted items are submitted
16. State actions are performed

Action Wizard

Use the **Action Wizard** to define the *action* [page 679] that you want to associate with the selected state or transition. You can set up some action types to be executed before or after a transition, upon entry to a state, or upon exit from a state.

For a state or transition, available actions include invocation of a synchronous *orchestration workflow* [page 690] (with reply), transition of a linked item, execution of a script, and execution of a Web service method. For a transition, asynchronous orchestration workflows (without reply) and triggers are also available.

On each page of the wizard, refine the action definition.

- For an asynchronous orchestration workflow (without reply), a trigger, and a transition, the definition includes which item will be affected, the conditions under which the action will occur, and the orchestration workflow to be invoked or the transition or trigger to be executed.
- For a synchronous orchestration workflow (with reply), a script, or a Web service, the definition includes which item will be affected, when the execution occurs (before or after a transition, for example), and the actual workflow, script, or Web service method to be executed.
- Some aspects of the definition (such as orchestration workflow type, field names, values, and relationships) must be specified. Click links in the description to make your selections.

Click **Next** and **Back** to modify your choices until you are satisfied with the action definition. Click **Finish** to save the definition.

Selecting the Action Type

On this screen, select the type of *action* [page 679] that should be performed for the selected state or transition. For states and transitions, you can select an *orchestration workflow* [page 690], script, transition, or Web service. For transitions, you can also select a trigger.

Note: If you are editing an existing action, you cannot change its action type.

Action Type	Description
Orchestration Workflow	The orchestration workflow that should be invoked as a result of this action. For transitions, you can also specify whether the *application workflow* [page 680] should continue processing as soon as the selected orchestration workflow is invoked, or wait for a reply from the orchestration workflow before continuing. For states, the application workflow always waits for a reply from the selected orchestration workflow before processing continues.
Script	The defined script that should be executed as a result of this action.
Transition	The transition that should be executed as a result of this action. Some combination of **Regular**, **Copy**, **Post**, and **Subtask** transitions will be available, depending on the affected items you select later in the **Action Wizard**.
Trigger	The trigger that should be executed as a result of this action. Triggers can only be executed for transitions.
Web Service	The defined Web service that should be invoked as a result of this action.

Selecting the Affected Item

On this screen, select the item to be affected by this action.

Action Type	Affected Item
Orchestration workflow ("continue executing")	**Items that reference this item using relational field:** Invokes the *orchestration workflow* [page 690] on items in which the current item is selected as a value. **Referenced items in relational field:** Invokes the orchestration workflow on the *principal item* [page 692] of a *subtask* [page 698] when the specified condition is met. **This item:** Invokes the orchestration workflow on the current or initiating item.

Action Type	Affected Item
Transition	**Items that reference this item using relational field:** Defines actions for the parent in parent-child transitions. This enables users to establish a parent-child relationship between multiple items and enables you to define an *action* [page 679] that executes transitions on items in which the current item is selected as a value. The current item can be selected as a value in multiple items. The action is applied to all items that meet the criteria. **Referenced items in relational field:** Defines actions for child items in parent-child transitions. This lets users define a parent-child relationship between items and lets you define an action that executes transitions on items selected as values in the field. **This item:** Defines subsequent actions for the initiating item. For most transition types, **This item** refers to the initiating item. For **Copy** transitions, **This item** refers to a copied item created as a result of the **Copy** transition. If you are setting action attributes for a state, this item is the only option available to you, and refers to either the principal item or the item affected by a trigger.
Trigger	**Items that reference this item using relational field:** Executes the trigger on items in which the current item is selected as a value. Select a primary *Multi-Relational* field to execute the trigger on items that have the current item selected as a value in this field when the condition for the action is met. **Principal item:** Executes the trigger on the principal item of a subtask when the specified condition is met. This option is not available for **Copy** transitions. **Referenced items in relational field:** Executes the trigger on items selected as values in the selected primary *Single Relational* or primary *Multi-Relational* field. Select a field to execute the trigger on items that are selected as a value in that field. **Subtask item** (triggers): Executes the trigger on subtasks when the specified condition is met. This option is not available for Copy transitions. **This item:** For Regular transition types, executes the trigger on the current or initiating item. For Copy transition types, executes the trigger on the copy that was created as a result of the transition. **Triggerable linked items:** Executes the trigger on linked items that have enabled triggers.

Selecting the Timing

On this screen, specify the timing of the *action* [page 679].

Action Type	Timing
Orchestration workflow ("wait for reply")	**After:** Invokes the *orchestration workflow* [page 690] on exit from the selected state or after the selected transition is executed. **Before:** Invokes the orchestration workflow on entry to the selected state or before the selected transition is executed.
Script	**After:** Runs the script on exit from the selected state or after the selected transition is executed. **Before:** Runs the script on entry to the selected state or before the selected transition is executed.
Web Service	**After:** Invokes the Web service method on exit from the selected state or after the selected transition is executed. **Before:** Invokes the Web service method on entry to the selected state or before the selected transition occurs.

Note: For a transition or state, you can define only one **After** and one **Before** action. If you defined one or the other for the selected state or transition, that choice is disabled. Click the disabled choice to see where it is used.

Selecting the Condition

On this screen, select the condition that causes this *action* [page 679] to be executed. The listed conditions depend on the affected item and the selected action type.

Sibling's Field Value

Performs an action on items where the current item is selected as a value in a primary *Multi-Relational* field. **All are**, **none are**, **half are**, and **most are** are available as operators. (**Most are** means "more than half.") The value of this field is evaluated for other selected items in the *Multi-Relational* field in which the current item is a selection.

Affected Item	Action Types
Items that reference this item using relational field	Orchestration workflow ("continue executing"), Transition

Field Items Value

Performs an action on items that are selected as values in a primary *Relational* field. The *Binary/Trinary* field is contained in the same table as the selected primary *Relational* field. If the primary *Relational* field is a *Single Relational* field, **is** and **is not** are available as operators. If the primary *Relational* field is a *Multi-Relational* field, **all are**, **none are**, **half are**, and **most are** are available as operators. (**Most are** means "more than half.")

Affected Item	Action Types
Referenced items in relational field	Orchestration workflow ("continue executing"), Transition

Sibling Tasks

Performs the action on the affected item, which is assumed to be a *subtask* [page 698], evaluating the status of sibling subtasks as defined by their value in the *Binary/Trinary* or *Single Selection* field used to determine *subtask status* [page 698]. Three values are available: **In Progress**, **Completed**, and **Rejected**. You can set the action to execute when all, none, half, or most subtasks meet one of two or three specified values. (Most means "more than half".)

Affected Items	Action Types
This item	Orchestration workflow ("continue executing"), Transition, Trigger
Principal item	Trigger
Subtask item	Trigger

Subtask's Status

Performs the action based on the value of the *Binary/Trinary* or *Single Selection* field used to determine subtask status. Three values are available: **In Progress**, **Completed**, and **Rejected**. You can set the action to execute when all, none, half, or most subtasks meet the specified value. (Most means "more than half.")

Affected Items	Action Types
This item	Orchestration workflow ("continue executing"), Transition, Trigger
Subtask item	Trigger

Trigger Received

States only: Lets you enable a trigger for a state. You must also set the trigger to fire during one or more transitions.

Affected Item	Action Type
This item	Trigger

Unconditionally

The action is always performed on items satisfying the condition.

Affected Items	Action Types
Referenced items in relational field	Orchestration workflow ("continue executing"), Transition
This item	Orchestration workflow ("continue executing"), Transition, Trigger
Principal item	Trigger
Subtask item	Trigger
Triggerable linked items	Trigger

Related Topics

Selecting the Action Type [page 375]

Selecting the Affected Item [page 375]

Selecting the Timing [page 376]

Selecting the Action [page 379]

Chapter 18: About Actions [page 371]

Selecting the Action

On this screen, select the action to be taken:

- The *orchestration workflow* [page 690] or Web service method to be invoked
- The AppScript, trigger, or transition to be executed

If the workflow or method you need is not listed, you can create it from this screen.

Chapter 19: Working with Transition Actions

Transition actions are typically used to coordinate items such that transitioning one item causes another item to also transition, according to a defined rule. Transition actions are typically subtask actions or parent-child actions.

Note: For general information about actions and for details about the **Action Wizard** you use to create the defined rules, see Chapter 18: About Actions [page 371].

Subtask Actions

A *subtask* [page 698] is a *primary item* [page 691] that is associated with a principal task. Subtasks are typically used when a set of smaller tasks needs to be worked on before a larger task, or principal task, can continue its process. Principal and subtask items are displayed in the **Subtasks** section in the SBM User Workspace for users who have the appropriate user privileges and preferences. Subtasks can be created in two ways:

- **Subtask transitions.** A subtask transition lets users create a primary item in a specified project. You can specify whether the new item should be linked to the original (principal) item. Subtask transitions are useful for setting up a set of new tasks at once. For example, a user can create a principal task and use a Subtask transition in the workflow to create subtasks from that larger task.
- **Linking principal tasks and subtasks in the SBM User Workspace.** Users who have the appropriate privileges can link and unlink items through the **Actions** drop-down list to create a principal/subtask relationship. This can be used to create a relationship between existing items.

After the principal/subtask relationship is established, you can define actions to move items through the workflow. You can define actions that transition the *principal item* [page 692] or subtask items unconditionally or based on the value in the *Binary/Trinary* or *Single Selection* field used to determine *subtask status* [page 698], or the value of any *Binary/Trinary* or *Single Selection* field. See Tutorial: Defining Subtask-Driven Actions [page 383] for an example.

Parent-Child Actions

Parent-child actions [page 690] differ from subtask actions in that they use values in primary *Relational* fields to execute transitions. The *action* [page 679] can be executed unconditionally or based on the values for a *Binary/Trinary* or *Single Selection* field in either the original item or the item in the relational table.

Instead of linking items together in the SBM User Workspace or through Subtask transitions, users select items as values in primary *Relational* fields to establish a parent-child relationship. For example, you can execute an action on items in the Incidents table that are selected as values in a primary *Relational* field in the Issues table. Parent-child actions are similar to the functionality provided with subtask actions, but use values in primary *Relational* fields to drive actions.

The benefit of using this type of action is that it lets users run reports based on the parent-child relationships and it executes transitions on all items meeting the action definition. Parent-child relationships can be created in two ways:

- **Child-driven actions.** This type of action executes a transition on a parent item, based on the status of its children. For example, if a parent item has two child items, the status of the children determines when the action is executed for the parent. See Tutorial: Defining Child-Driven Actions [page 386] for an example.
- **Parent-driven actions.** This type of action executes a transition on child items, based on the status of the parent. For example, if a parent item has two child items, the status of the parent determines when the action is executed for the child items. See Tutorial: Defining Parent-Driven Actions [page 388] for an example.

This section contains the following information:

Tutorial: Basing an Action on a Single Selection Field

You can create an action that executes a specific transition when a user selects a value in a field. For example, when a user selects a value from a *Category* field, the item is transitioned to a specific state based on that value. If users select **New Order** from the *Category* field, the **Place New Order** transition is executed and the item is sent to an **Order Placed** state.

Prerequisites:

The following must be set up before you perform this procedure:

- *Application workflow* [page 680] renamed to `Single Select`.
- *Single Selection* field named "Category" with **New Order**, **Service**, and **Account Management** values
- **Assigned** state
- **Assign** transition that leads from the **New** state to the **Assigned** state
- **Order Placed** state
- **Place New Order** transition that leads from the **Assigned** state to the **Order Placed** state
- **Closed** inactive state
- **Close** transition that leads from the **Order Placed** state to the **Closed** state

To set up an action that is based on a *Single Selection* field:

1. In the application workflow editor, right-click the **Assign** transition, and select **Show Actions**.
2. On the **Actions** tab of the workflow Property Editor, click **New**. The **Action Wizard** opens.
3. Select **Transition** and then click **Next**.
4. Select **This item** and then click **Next**.
5. Select **Field Value**.
6. Click **specified** and select **Category**. The **value** link changes to one of the values for the *Category* field.
7. If that value is not **New Order**, click the value and then select **New Order**.
8. Click **Next**.
9. Select **SingleSelect:Place New Order(Assigned->Order Placed)**.
10. Click **Finish**.
11. Deploy the process app.
12. Test the process app.
    a. In the SBM User Workspace, submit an item into the Single Select project.
    b. In the **Title** box, type a name for the item.
    c. In the **Category** list, select **New Order**.
    d. Click the **Assign** transition button.

    The **Place Order** transition is automatically executed.

Tutorial: Defining Subtask-Driven Actions

This example describes how to add a **Subtask** transition that posts an item from one application to another. After you add the **Subtask** transition, you can define actions that transition principal tasks based on the subtask's values in a *Binary/Trinary* or *Single Selection* field used to determine subtask status. For example, in a Human Resources (HR) workflow that requires an Information Technology (IT) department to purchase a computer for a new employee, you can create a **Subtask** transition that submits a task into the IT workflow for the purchase. After IT closes its task in its workflow, the task in the HR workflow is automatically closed.

Prerequisites:

The following must be set up before you perform this procedure:

- Two applications, named `New Hires` and `IT Tickets`
- One *application workflow* [page 680] for each *application* [page 679], named `New Hires` and `Asset Management`, respectively.
- *Binary/Trinary* field named `Subtask Status` in the **IT Tickets** table, with **In Progress** and **Completed** values.
- In the **New Hires** workflow:
  - **Waiting for IT** "active" state
  - **Send to IT** "subtask" transition that leads from the **New** state to the **Waiting for IT** state.
  - **Closed** "completed" state
  - **Resolved by IT** "regular" transition that leads from the **Waiting for IT** state to the **Closed** state.
- In the **Asset Management** workflow:
  - **In Progress** "active" state
  - **Assign** "regular" transition that leads from the **New** state to the **In Progress** state
  - **Closed** "completed" state
  - **Close** "regular" transition that leads from the **In Progress** state to the **Closed** "completed" state

Note: A process app that defines subtask-driven actions requires at least two applications. The applications can be in the same process app, or can be in another process app, if the current process app contains references to them. See About References [page 361] for more information.

To set up a subtask-driven *action* [page 679]:

1. Configure the **Send to IT** transition in the **New Hires** workflow.
   a. On the **Options** tab of the transition Property Editor, select the **Quick Transition** check box.
   b. On the **Post Options** tab of the transition Property Editor:
      - In the **Post Application** list, select **IT Tickets**.
      - In the **Post Table** list, select **IT Tickets**.
      - In the **Use submit transition** list, select **Asset Management : Submit**.
      - For **When finished, show**, select **New Item**.

- In the **Item Link Type** list, select **2-Way, no triggers**.

2. Set the *subtask status* [page 698] in the **Asset Management** workflow.
   a. Select the **Completed** state. This is the state to which the subtasks are transitioned.
   b. On the **General** tab of the state Property Editor, in the **Subtask** list, select **Subtask Status**.
   c. Select the **Close** transition.
   d. On the **Field Overrides** tab of the transition Property Editor, select the *Subtask Status* field, select the **Override field properties** check box, select **Set to Default**, and then select **Completed** in the **Default Value** list.
   e. Still on the **Field Overrides** tab, make sure that there are no required fields for this transition, or set default values for fields that require values.
3. Create an action in the **New Hires** workflow that will transition principal items, based on the value of the *Subtask Status* field.
   a. Select the **Waiting for IT** state. This is the state in which principal items remain while they wait for subtasks to complete.
   b. On the **Actions** tab of the state Property Editor, click **New**.
   c. In the **Action Wizard** that opens, select **Transition**, and then click **Next**.
   d. Select **This item**, and then click **Next**.
   e. Select **Subtask's Status**.
   f. Select **all are**, select **Completed**, and then click **Next**.
   g. Select the **New Hires:Resolved by IT (Waiting for IT->Close)** transition, and then click **Finish**.
   h. Click the **Resolved by IT** transition.
   i. On the **Field Overrides** tab of the transition Property Editor, make sure there are no restrictions, such as *Item Type* [page 688] restrictions, on the transition.
4. Test the process app.

a. In the SBM User Workspace, submit an issue into the New Hires project.

b. Click the **Send to IT** transition button.

c. Submit an issue into the IT Tickets project from the window that opens.

d. Click the **Assign** transition button.

e. Click the **Close** transition button. The IT Tickets subtask item is in the **Closed** state.

f. Click the *principal item* [page 692] in the **Subtasks** or **Attachments** section. The New Hires principal item opens, and is in the **Closed** state.

Note: In the SBM User Workspace, users are presented with a list of projects into which they have privileges to submit items. If only one project is valid for submission, the submit form opens for that project. If multiple projects are valid for submission, you can specify a single project by selecting a value in the **Use submit transition** field in the transition Property Editor. For more information, see Post Options Tab of the Transition Property Editor [page 316]. In addition, in SBM System Administrator, administrators can configure transitions in projects to select a specific project into which all items are submitted.

Tutorial: Defining Child-Driven Actions

This example explains how to define an action that transitions a parent item when its child items are completed. It includes adding fields to drive the action; defining a transition action that creates child items, and defining the transition action that closes a parent item, based on the status of its children.

Prerequisites:

The following must be set up before you perform this procedure:

- *Multi-Relational* field named `Children`
- *Single Relational* field named `Parent`
- *Application workflow* [page 680] named `Parent`:
  - **In Progress** state
  - **Assign** transition that leads from the **New** state to the **In Progress** state
  - **Child Created** state
  - **Create Child** Post transition that leads from the **In Progress** state to the **Child Created** state.
  - **Closed** state
  - **Close** transition that leads from the **Child Created** state to the **Closed** state.
- Application workflow named `Child`:
  - **In Progress** state
  - **Assign** transition that leads from the **New** state to the **In Progress** state
  - **Close** transition that leads from the **In Progress** state to the **Closed** state

To set up a child-driven *action* [page 679]:

1. Configure the **Create Child** transition in the **Parent** workflow.
   a. In the **New Item in Original Original Item's Field** list, select **Children**.
   b. In the **Set Original Item in New Item's Field** list, select **Parent**.
   c. In the **Use Submit Transition** list, select **Child : Submit**.
   d. In the **Item Link Type** list, select **2-Way, no triggers**.
2. Create an action that will close the parent item when the status of every child item is **Completed**.
   a. In the **Child** workflow, right-click the **Close** transition, and select **Show Actions**.
   b. On the **Actions** tab of the transition Property Editor, click **New**.
   c. In the **Action Wizard** that opens, select **Transition**, and then click **Next**.
   d. Select **Items that reference this item using relational field**.
   e. Click the **specified** link, select **Children**, and then click **Next**.

f. Select **Sibling's Field Value**.

g. Click the **specified** link and then select **Active/Inactive**.

h. Make sure the next link reads **all are**.

i. Click the **Active** link, select **Inactive**, and then click **Next**.

j. Select **Parent:Close (Child Created->Closed)**, and then click **Finish**.

3. Deploy the process app.

4. Test the process app.

a. In the SBM User Workspace, submit an item into the Parent project.

b. Click the **Assign** transition button.

c. Click the **Create Child** transition button.

d. Submit an item into the Child project from the window that opens.

e. Click the **Assign** transition button.

f. Click the **Close** transition button.

g. Select all items in the **Children** field and then click **OK**.

h. Click the parent item in the **Attachments** section. The parent item opens, and is in the **Closed** state. This is because when the children moved to the **Closed** state, the value of the *Active/Inactive* field becomes **Inactive**, and the action rule specifies that the parent item should close when the children items are all *Inactive*.

Tutorial: Defining Parent-Driven Actions

This example explains how to define an action that closes a child item when its parent is completed.

Prerequisites:

The following must be set up before you perform this procedure:

- *Single Relational* field called `Child`
- *Application workflow* [page 680] named `Parent Workflow`:
  - **In Progress** state
  - **Assign** transtion that leads from the **New** state to the **In Progress** state
  - **Closed** state
  - **Close** transtion that leads from the **In Progress** state to the **Closed** state
- Application workflow named `Child Workflow` with the same steps and transitions as the **Parent Workflow**

To set up a parent-driven *action* [page 679]:

1. Create an action that will close the child item when the value of the *Active/Inactive* field in the parent item is **Inactive**.
   a. In the **Parent** workflow, right-click the **Close** transition, and select **Show Actions**.
   b. On the **Actions** tab of the transition Property Editor, click **New**.
   c. In the **Action Wizard** that opens, select **Transition**, and then click **Next**.
   d. Select **Referenced items in relational field**.
   e. Click the **specified** link and then select **Child**.
   f. Select **Field value**.
   g. Click the **specified** link and select **Active/Inactive**.
   h. Make sure the next link reads **is**.
   i. Click the **Active** link, select **Inactive**, and then click **Next**.
   j. Select **Child:Close (In Progress->Close)**, and then click **Finish**.
2. Deploy the process app.
3. Test the process app.
   a. In the SBM User Workspace, submit an item into the **Parent Workflow**.
   b. Click **Assign**.
   c. Submit an item into the **Child Workflow**.
   d. Click **Assign**.
   e. In the **Parent Workflow**, select the child item in the **Child** field, and then click **Close**. In the **Child** field in the **Standard Fields** section, the child item has **(Inactive)** after its name. This means the item was closed.

Tutorial: Submitting Multiple Primary Items

This topic shows how to submit multiple primary items based on a single transition action. You will create a workflow and configure it so users can create two items, each one in a separate project, when a third item is submitted. In this example, the three items are linked together, but no parent/child or principal/subtask relationship is established, and the items move independently through their workflows.

Prerequisites:

The following must be set up before you perform this procedure:

- *Application workflow* [page 680] named `Software Issues`
  - **Submit Defect** Regular transition that leads from the **Submit** state to the **New** state
  - **Submit Enhancement** Regular transition that leads from the **Submit** state to the **New** state
- Application workflow named `Product Development`
  - **Defect Post** Post transition that leads from the **Any** state back to itself
  - **Enhancement Post** Post transition that leads from the **Any** state back to itself

Consider the following information when you use actions to submit multiple primary items.

- The example in this topic uses **Post** transitions to submit multiple items. You can also use **Copy** or **Subtask** transitions.
- The example is defined within a single *application* [page 679]. You can use **Subtask** or **Post** transitions to set up multiple submittals across applications as needed. Make sure users have submit privileges into the projects specified in the **Subtask** or **Post** transitions.
- When you implement multiple submittals in your system, consider how you want data to be mapped. By default, some field values automatically map to the new items, depending on the type of transition you use to submit new items, and whether you are submitting items into the same *primary table* [page 692] or to a different primary table. You can specify your own mapping or use the default mapping. You can also define default field values for the **Copy**, **Post**, or **Subtask** transitions.

To submit multiple primary items:

1. Configure the transitions in the **Software Issues** workflow:
   a. Click the **Submit Defect** transition.
   b. In the transition Property Editor, on the **Options** tab, select the **Quick Transition** check box.
   c. Click the **Submit Enhancement** transition.
   d. In the transition Property Editor, on the **Options** tab, select the **Quick Transition** check box.
   e. Save the process app.
2. Configure the transitions in the **Product Development** workflow:
   a. Click the **Defect Post** transition.

b. In the transition Property Editor, on the **Options** tab, select the **Quick Transition** and **Hide Button on Form** check boxes.

c. On the **Post Options** tab, in the **Use submit transition** list, select **Software Issues : Submit Defect**.

d. Still on the **Post Options** tab, In the **Item Link Type** list, select **2-Way, no triggers**.

e. Click the **Post Enhancement** transition.

f. In the transition Property Editor, on the **Options** tab, select the **Quick Transition** and **Hide Button on Form** check boxes.

g. On the **Post Options** tab, in the **Use submit transition** list, select **Software Issues : Submit Enhancement**.

h. Still on the **Post Options** tab, in the **Item Link Type** list, select **2-Way, no triggers**.

3. Create actions to allow multiple submittals:

a. In the **Product Development** workflow, right-click the **Submit** transition, and then select **Show Actions**.

b. In the transition Property Editor, on the **Actions** tab, click **New**. The **Action Wizard** opens.

c. Select **Transition**, and then click **Next**.

d. Select **This Item**, and then click **Next**.

e. Select **Unconditionally**, and then click **Next**.

f. Select **Product Development:Defect Post([Any])->([Any])** and then click **Finish**.

g. In the transition Property Editor, on the **Actions** tab, click **New** again.

h. In the **Action Wizard**, repeat steps c, d, and e.

i. Select **Product Development:Enhancement Post([Any])->([Any])** and then click **Finish**.

4. Deploy the process app.

5. In SBM System Administrator, add two subprojects to the Software Issues project.

a. On the **Projects** tab, select **Software Issues Project**.

b. Click **Add**.

c. In the **Add Project** dialog box, type **Defects** in the **Project Name** box.

d. Repeat steps b and c, except type **Enhancements** in the **Project Name** box.

6. Still in SBM System Administrator, edit the **Post** transitions in the Product Development project.

a. On the **Projects** tab, select **Product Development Project**.

b. Click **Edit**.

c. In the **Edit Project** dialog box, click the **Transitions** tab.

d. Select the **Defect Post** transition, and then click **Edit**.

e. In the **Edit Transition** dialog box, select the **Override** check box above the **Post Item Project** box.

f. Select **Defects** under **Software Issues Project**, and then click **OK**.

g. Repeat steps d, e, and f, except select the **Enhancement Post** transition, and select **Enhancements** under **Software Issues Project**.

7. Test the process app:

a. In the SBM User Workspace, submit an issue into the Product Development project. The **Attachments** and **Change History** sections indicate that when you submitted issue 000225, issues 000226 and 000227 were automatically submitted. Issue 000226 was submitted into the Defects project, and issue 000227 was submitted into the Enhancements project.

Chapter 20: Working with Scripts

This section contains the following information:

About SBM AppScript

SBM supports a scripting language called SBM AppScript, which offers a degree of power and flexibility beyond that available through the administration interfaces. You can associate scripts that implement custom features with transitions, notifications, and the self-registration form. You can also set up scripts that run when a user visits a special SBM URL.

SBM AppScript is modeled after VBScript 4.0 and contains extensions to support SBM. Any VBScript programmer can use SBM AppScript to implement custom features in your SBM system.

Script Editor

In the script editor, you create and edit scripts written in the SBM AppScript language.

Note: To edit an existing script, you must have the script checked out.

To validate the script, click the **Validate Script** button in the upper left corner of the SBM Composer window. You can view the results in the Validation Output pane.

For more information about creating and editing scripts, see the *SBM AppScript Reference* or the SBM AppScript reference section of the online help.

Validation Output Pane

The Validation Output pane displays the status and results when you validate a script written in the SBM AppScript language.

The Validation Output pane occupies the same space as the Message List, below the script editor. When both the Message List and Validation Output pane are open, you can arrange them by dragging, just as you can arrange editor tabs in the editor pane.

Chapter 21: Working With Triggers

Triggers can be set up so that they execute automatically or as specified by users. When used with transition types that automatically link items (**Copy**, **Post**, **Subtask**, and **External Post**), triggers can fire without any intervention. You can also create triggers that users manually activate when they create a link between two items in the SBM User Workspace.

Triggers can be used to:

- Transition items that are selected as values in the *primary relational field* [page 691] or in which a primary relational field is the selected value.
- Transition items that have principal or *subtask* [page 698] relationships.

CAUTION: Triggers will fail when enabled for transitions that have unspecified values for required fields. For best results, provide default values for fields that are set as required for transitions in which triggers are enabled.

This section contains the following information:

Setting Up a New Trigger

To set up a new trigger:

1. Create the trigger.
   a. Open the *Global Process App* [page 687].
   b. In App Explorer, select the **Triggers** heading.
   c. Right-click an existing trigger or an empty area in the left pane of the triggers editor.
   d. Select **Add New Trigger**.
   e. Type a name for the trigger in the **Name** field on the right side of the triggers editor.
2. Enable the trigger for a specific state.
   a. In App Explorer, select the workflow containing the state for which you want to enable a trigger.
   b. In the workflow editor, select the state.
   c. In the Property Editor for the state, click the **Actions** tab, and then click **New**.
   d. In the **Action Wizard** that opens, select **Transition**, and then click **Next**.

e. Click **Next** again.

f. Select **Trigger Received**.

g. Click **specified**, select the trigger you just created, and then click **Next**.

h. Select the transition you want to invoke, and then click **Finish**.

3. Set the trigger to fire for a specific transition.

a. In the workflow editor, select the transition you specified above.

b. On the **Actions** tab of the transition Property Editor, click **New**.

c. In the **Action Wizard** that opens, select **Transition**, and then click **Next**.

d. Select **Triggerable linked items**, and then click **Next**.

e. Select the conditions under which you want the trigger to be invoked.

- **Field Value:** Click **specified**, and select a field to be checked. Click the other links to define the condition (is, is not) and the value.
- **Sibling Tasks** or **Subtask's Status:** Click the links to define the condition (all are, half are, most are, none are) and the value.
- **Unconditionally:** *(No additional settings are required.)*

f. Click **Next**.

g. Select the trigger, and then click **Finish**.

The trigger is available to be manually activated by users in the SBM User Workspace when they transition linked items as specified by the trigger.

Considerations for Automatic Transitions

You can create triggers that automatically transition items without intervention from the user.

Triggers can be automatically generated when used with **Copy**, **Post**, and **Subtask** transition types because these transition types can automatically create *primary item* [page 691] links.

Tip: In some cases, it may be easier to use a **Subtask** transition than to manually create triggers. **Subtask** transitions create a new primary item in the specified project. The new item is linked to the original item and can be set up so that, when the new item reaches an inactive state, the original item is transitioned from a waiting state to another state in the workflow. The transition occurs only when the new item reaches an inactive state, though; if you want a transition to fire at an active state, you should use a trigger.

To set up an automatic trigger:

1. In the workflow editor, add a **Copy**, **Post**, or **Subtask** transition to the workflow in which the trigger should be enabled.
2. On the **Post Options** tab of the transition Property Editor, select an item link type.

3. On the **Options** tab of the Property Editor, select **Hide button on form**. This specifies that the transition be used only with triggers, automatically and with no intervention from users, and hides the transition button from users in the SBM User Workspace.

Chapter 22: Working with Web Services

You can add *Web services* [page 701] to your process apps, making their defined operations available as actions assigned to states and transitions in your application workflows. (See Chapter 18: About Actions [page 371] for details.) To use this feature, you should be familiar with how Web services work and with the particular operations, inputs, and outputs of the Web services you are calling.

Using the Web service operation property editor, you can map each operation's inputs and outputs to application fields. In applications, Web services add capabilities to workflows that are primarily human-centered. That is, the workflows, forms, and tables are intended to structure and capture information that comes from people. Examples include issue and defect tracking.

Note: By comparison, Web services in orchestrations are organized into a workflow that is primarily machine-centered. They're arranged using control flow structures such as repeating loops, decision steps, and fault handlers. See Part 4: Working with Orchestrations [page 429] for details.

Here are a few points to remember when using Web services in an *application* [page 679]:

- Support for development efforts writing Web services is provided by Professional Services . Questions regarding use of Web services operations in orchestration processes as documented are handled by Serena Customer Support.
- Authentication for Web services is specified for the service's *endpoint* [page 684] in either SBM Composer (in the Deploy Process App Dialog Box [page 630]) or Application Administrator.
- Web service calls from an application are executed synchronously, which means that SBM waits for each Web service call to return (or for the specified time-out period to expire).

  Note: Web service calls from an *orchestration workflow* [page 690] are executed asynchronously, so your workflow can go on to something else after the Web service is called.
- Input data is passed to the Web service in UTF-8.
- Output data from the Web service is assumed to be in UTF-8.
- WSDLs may fail to import if they are not formatted correctly or they contain functionality that is not supported by SBM.
- All errors and Web service invocation faults are recorded to the Event Viewer on the SBM Server.

Note: You can also use the Web Services API to create integrations that create, read, update, and delete SBM items. See the *Web Services Developer's Guide* for an overview of the services and how to use them.

This section contains the following information:

Using the Web Services List

The list of *Web services* [page 701] is displayed in the editor pane when you click the **Web Services** heading in App Explorer.

Select a listed Web service to view or edit it in the Web service Property Editor. Double-click a listed Web service to open it in the Web service editor as well.

Click the **Find** icon on the **Home** tab of the Ribbon to search for a specific Web service in the list.

Web Service Editor

The Web service editor is displayed when you select a Web service in App Explorer (or in the list of *Web services* [page 701] that is displayed when you select the **Web Services** heading in App Explorer). Use it to view (though not actually modify) information about the selected Web service.

Use the Web service Property Editor to view and edit the information about the selected Web service.

Select one of the listed operations to view their properties in the Web service operation Property Editor.

General Tab of the Web Service Property Editor

This tab is displayed in the Property Editor when you click the banner (bearing the Web service name) in the Web service editor, or when you select a Web service from the list of *Web services* [page 701] that is displayed when you select the **Web Services** heading in App Explorer.

Element	Description
Name	Shows the name of the Web service.
Description	An optional description of the Web service.
WSDL	Shows the location from which the WSDL file was imported.

Element	Description
Reimport/ Refresh	Reimports the WSDL file and updates it if there is a later version available.  **Note:** If you add an *event* [page 685] field or change the name of an existing event field in an application's *primary table* [page 692], you could need to reimport the *event definition* [page 685] WSDL file before you *publish* [page 692] or export the process app.
Documentation	Displays the documentation for the Web service, as specified in the WSDL file.

General Tab of the Web Service Operation Property Editor

This tab is displayed in the Property Editor when you select (in the Web service editor) a specific operation of the Web service.

Element	Description
Name	Shows the name of the Web service operation.
Description	Shows the description of the Web service operation.
Documentation	Lists the documentation for the Web service, as specified in the WSDL file.

 **Tip:** Click the banner (bearing the Web service's name) at the top of the Web service editor to display General Tab of the Web Service Property Editor [page 400], where you can view and edit properties of the Web service itself.

Inputs Tab of the Web Service Operation Property Editor

This tab is displayed in the Property Editor when you select (in the Web service editor) a specific operation of the Web service. It display the inputs to the Web service, as specified in the WSDL file.

Although you cannot edit this information, you can see what inputs are required by the Web service operation. For example, a stock quote Web service might require an input string identifying the stock symbol.

When the Web service is included as part of an *orchestration workflow* [page 690], Data Mapping Tab of the Orchestration Workflow Property Editor [page 459] displays very similar information. Editable values, such as a password string, can be edited there.

 **Note:** The drop-down list at the top of the Property Editor lets you view other operations in the Web service.

Element	Description
Elements	Shows the data elements defined as inputs to the selected Web service.
Type	Shows each input's data type, such as integer, Boolean, and complex types.
Vertical divider	Clicking the vertical divider to the right of the **Type** column displays additional information for the selected data element (such as its type and namespace).

Tip: Click the banner (bearing the Web service's name) at the top of the Web service editor to display General Tab of the Web Service Property Editor [page 400], where you can view and edit properties of the Web service itself.

Outputs Tab of the Web Service Operation Property Editor

This tab is displayed in the Property Editor when you select (in the Web service editor) a specific operation of the Web service. It displays the outputs from the Web service, as specified in the WSDL file.

Although you cannot edit any of the information, you can see what outputs are returned by the Web service operation. For example, a stock quote Web service could return an output string representing the quoted stock.

When the Web service is included as part of an *orchestration workflow* [page 690], Data Mapping Tab of the Orchestration Workflow Property Editor [page 459] displays very similar information.

Note: The drop-down list at the top of the Property Editor lets you view other operations in the Web service.

Element	Description
Elements	Shows the data elements defined as outputs from the selected Web service.
Type	Shows each output's data type, such as integer, Boolean, and complex types.
Vertical divider	Clicking the vertical divider to the right of the **Type** column displays additional information for the selected data element (such as its type and namespace).

Tip: Click the banner (bearing the Web service's name) at the top of the Web service editor to display General Tab of the Web Service Property Editor [page 400], where you can view and edit properties of the Web service itself.

Faults Tab of the Web Service Operation Property Editor

This tab is displayed in the Property Editor when you select (in the Web service editor) a Web service operation for which a fault has been defined.

Element	Description
Fault name	Lists the name of the fault.
Fault message	Lists the message defined in the WSDL file for this fault.

Tip: Click the banner (bearing the Web service's name) at the top of the Web service editor to display General Tab of the Web Service Property Editor [page 400], where you can view and edit properties of the Web service itself.

Chapter 23: Providing Guidance to Users

Serena provides online help for using the SBM User Workspace. You might want to provide additional guidance or help that is specific to your process apps. The following topics describe ways you can use SBM Composer to provide guidance or help to users in the SBM User Workspace.

Creating Links to Web Pages or Popup Windows

On custom forms, you can create links to Web pages or popup windows that contain information about an open item. You can type HTML code or plain text to create the content of the popup windows, and can specify various popup window features, such as their size and whether they should have menu bars.

You can insert references to table fields and other form controls in the URLs for Web pages and in content of popup windows.

For information about inserting references, see Using the String Builder Tool [page 249].

The following controls can provide the links:

- Button
- HyperLink
- Image

Creating Links to Web Pages

This section describes how to make a Web page open when a user clicks a control.

To create a link to a Web page:

1. Add a **Button**, **HyperLink**, or **Image** control to a custom form near the field for which you want to provide guidance. These controls are in the **Form Palette** under **Other Controls**.
2. Click the **Behavior** tab on the control Property Editor.
3. Select **Open URL**.
4. In the **URL** box, enter the URL of the Web page, or build the URL by typing { to insert references to table fields and other form controls.

5. In the **Target** list box, select **New Window** or **Same Window**, depending on whether you want the Web page to open in the same browser window as the SBM User Workspace or in a separate window.

Web Page Examples

- In an employee time-off application, you could put a link on the Submit *transition form* [page 700]. When a user clicks the link, the intranet page that explains the time-off policy opens.
- In a human resources application, you could have link that opens a URL to google.com. When you configure the link, you use the *string builder tool* [page 698] and enter `http://www.google.com/search?q={Candidate Name}`. The search returns links to things such as articles the candidate published.

Creating Popup Windows

This section describes how to make a popup window open when a user clicks a control.

To create a popup window:

1. Add a **Button**, **HyperLink**, or **Image** control to a custom form near the field for which you want to provide guidance. These controls are in the **Form Palette** under **Other Controls**.
2. Click the **Behavior** tab on the control Property Editor.
3. Select **Open custom popup**.
4. If you want the text in the **Body HTML** box to wrap as you enter text into it, select the **Wrap** check box. If you clear this check box, the text will be on one line, with a scroll bar at the bottom of the window.
5. In the **Body HTML** box, enter the text that you want users to see when they click the control. You can type HTML code if you want the text to be formatted, or simply type text if you do not want the text to be formatted. Type `{` to insert references to table fields and other form controls.

   Important: JavaScript within the HTML code is not supported.

   Note: There is no limit to the number of characters in the window.

6. The **Options** box contains default values for some window features. You can change the values or add additional features and values as described in the following table.

   Note: To restore the default values, click **Reset.**

Option	Description
height	The height of the window, in pixels.
left	The left position of the window, in pixels.

Option	Description
location	Whether the field where you enter the URL is displayed (`yes` or `1`, `no` or `0`).
menubar	Whether the menu bar is displayed (`yes` or `1`, `no` or `0`).
resizable	Whether handles used to resize the window are displayed at the corners of the window (`yes` or `1`, `no` or `0`).
scrollbars	Whether horizontal and vertical scrollbars are displayed if the content is larger than the window (`yes` or `1`, `no` or `0`).
status	Whether a status bar is displayed at the bottom of the window (`yes` or `1`, `no` or `0`).
toolbar	Whether the Web browser toolbar is displayed (`yes` or `1`, `no` or `0`).
top	The top position of the window, in pixels.
width	The width of the window, in pixels.

Popup Window Examples

- In a customer support application, you could put a link next to an *Impact* field. When a user clicks the link, a popup window opens. The popup window describes each level of impact that a user can select from the field. For example, the window could describe the **Unable to Work**, **Using Workaround**, and **Inconvenienced** values.
- You could have a button at the top of a transition form. When a user clicks the button, a popup window opens. The popup window describes each transition that can be executed from the form.
- In an IT asset procurement application, you could add a **Confirm** button on the **Submit** transition form. After employees complete the form, they can click the button to see a popup window containing the values of fields on the form. If the information is inaccurate, they can change the field values before submitting the form.

Note: When you configure the button, you could insert references to database fields. For example, you could insert references to the *Employee Name*, *Computer Type*, and *Employee Location* fields, as shown in the following illustration:

Body HTML:

```
<table>
    <tr><td>Name: </td><td>{Employee Name}</td></tr>
    <tr><td>Computer:</td><td>{Computer Type}</td></tr>
    <tr><td>Location:</td><td>{Employee Location}</td></tr>
</table>
```

Providing Field Descriptions

This method is ideal for providing short descriptions for fields. You can include a description of up to 255 characters for a field in its Property Editor. This marks the field with a dotted underline in the SBM User Workspace. The user can hover over the field to read the description. For example, in an issue defect management *application* [page 679], you could include a description of what "regression" means when you configure the *Regression* field. The user can read the description before selecting **Yes** or **No**.

To provide a field description:

1. In App Explorer, select the table containing the field.
2. In the table editor, select the field for which you want to provide guidance.
3. Click the **General** tab in the field Property Editor.
4. Type the guidance information in the **Description** box.

Using Read-Only Text Fields

You can use read-only *Text* fields on custom forms to provide guidance information. For example, at the top of a form in an employee time off application, you could include a read-only *Text* field that informs employees that they can request time off in no less than 4-hour increments.

To use read-only *Text* fields:

1. In App Explorer, select the table that will contain the text field.
2. Drag a *Text* field onto the table.
3. In the table editor, select the field.
4. Click the **Options** tab in the field Property Editor.
5. In the **Style** section, type the number of characters that you expect the guidance information to contain.
6. Click the **Attributes** tab in the field Property Editor.
7. Type the guidance information in the **Default Value** box, and select the **Read Only** check box.
8. In App Explorer, click the custom form.
9. Drag the text field from the **Form Palette** to the appropriate area on the custom form.

Annotating Application Workflows

You can add annotations, or notes, to application workflows and sub-workflows. Annotations let you provide comments about a state, transition, workflow, or workflow section. They are useful in complex workflows and can be helpful to someone else who needs to understand the logic and flow of the workflow.

In the SBM User Workspace, users can click the **Display this item's associated workflow** icon to see the workflow and its annotations. This icon is visible only if the user has the **View Workflow Graphically** privilege.

To add an annotation:

- Right-click a state, transition, or empty area in the workflow editor, and select **Add Annotation**.

  The state, transition, or workflow name appears in the annotation.

Chapter 24: Troubleshooting

In SBM Composer, the Message List and the Log Viewer are the primary tools for troubleshooting applications and orchestrations.

In this section, you learn how to use the Message List to validate process apps before you *publish* [page 692] and deploy them. You also learn how to debug process apps using the Log Viewer after they are deployed and run.

For additional information on troubleshooting orchestrations, see Chapter 30: Troubleshooting Orchestration Workflows [page 573] and Using the Scope, Throw, and Compensate Steps to Handle Faults From Web Services [page 498].

This section covers the following topics:

- Using the Message List [page 411]
- Using the Log Viewer [page 415]
- Using the Validation Output Pane [page 427]
- Debugging for Development and Support [page 427]

Using the Message List

Although the Message List provides information about the check-in, check-out, and *publish* [page 692] processes, you mainly use the Message List to troubleshoot validation and *deployment* [page 683] problems. You can also use the Message List to troubleshoot

Validation Problems

When you click the **Validate** button on the Quick Access Toolbar, SBM Composer attempts to validate the open process app to determine if it is ready to be published to SBM Composer. (If you click the **Publish**, **Deploy**, or **Quick Deploy** buttons before you validate, SBM Composer tries to validate the process app first.)

For example, if you did not connect a state with a transition in an *application workflow* [page 680], when you validate, a warning message is displayed in the Message List telling you that the state is unreachable. Or, if you fail to provide a required value for a step input in an *orchestration workflow* [page 690], an error message is displayed.

Deployment Problems

When you click the **Deploy** or **Quick Deploy** button in the Quick Access Toolbar, SBM Composer attempts to deploy the process app. The message that is displayed in the Message List advises you if the deployment operation succeeded or if it was aborted.

Note: When you click the **Deploy** or **Quick Deploy** button, the process app is automatically validated, checked in, and published if you have not performed these steps before you deploy.

For additional information about troubleshooting orchestrations using the Message List, refer to Troubleshooting Orchestrations Using the Message List [page 573].

This section covers the following topics:

Opening the Message List

To open the Message List:

1. On the **Home** tab of the Ribbon, in the **Common Views** group, select the **Message List** check box.

   The Message List opens.

   Note: The Message List automatically opens when you validate a process app.

2. If the **Message List** check box is already selected and the Message List is not visible, click the **Message List** tab below the Property Editor.

Message List

The following table describes the elements that appear in the Message List.

Element	Description
Description	Error message text
Name	Name of the element to which the message refers
Element type	Element type (used for validation messages only)
Design element	Name of the design element that contains the element (validation messages only)
Application/ Orchestration	Name of the *application* [page 679] or *orchestration* [page 690] that contains the design element (validation messages only).
Show	Messages for the selected logging level: Error, Warning, and Info. (See Filtering Messages by Logging Level [page 414].)
Time	Date and time the message was received

Validating a Process App

Before a process app can be published or deployed to Application Administrator or exported to a file, it must pass validation.

Tip: While you are creating an *application workflow* [page 680] or *orchestration workflow* [page 690], validate often to catch potential problems.

To validate the open process app:

1. In the Quick Access Toolbar, click the **Validate** button. Alternatively, click **Validate** from the Composer menu, or press Alt and then V, or press Ctrl+Shift+V.

   A message box opens showing the number of any errors, warnings, or messages and advising you if the process app passed or failed validation.

2. Click **OK**.

   Details about the errors, warnings, and messages are displayed in the Message List, along with informational messages. You can use the filtering options in the to choose which types of message you want to view. (See Filtering Messages by Logging Level [page 414].)

Note: By default, the name of any *design element* [page 683] that contains errors appears in red in App Explorer. You can disable this feature or change the color on the **General** tab of the Options Dialog Box [page 43].

Following are some typical situations that generate error and warning validation messages:

- You used a *process app* [page 692], *application* [page 679], or table name that already exists in a deployment environment.
- One or more references in a process app are unresolved.
- SBM Composer detects that a process app or a feature in a process app is incompatible with on-demand *deployment* [page 683]. You receive these error messages if you are working in on-demand mode, or if you are working in on-premise mode and the **Include validation for On-Demand compatibility** option is selected on the **General** tab of the Options Dialog Box [page 43] dialog box, and if one or more of the following conditions is true:
  - The process app is the Global Process App.
  - Scripts or triggers are included in the process app.
  - A **Publish** or **External Post** transition is included in an application workflow.
  - A **Post** transition with an **Item link type** that includes "triggers" is included in an application workflow.
  - A transition that uses a **Trigger** action type is included in an application workflow.
  - A state that has a transition action with a **Trigger Received** condition is included in an application workflow.

- The database name for a table name exceeds the maximum number of characters that are allowed.
- A table in the process app includes a *Folder* field, *Company* field, *Contact* field, *Resolution Summary* field, *Resolution Description* field, *Multi-User* field with a **Selection mode** of **Groups & users**, or a *User*, *Multi-User*, or *Multi-Group* field that is a dependency or has dependencies.
- Various project and table privileges are selected for a *role* [page 695] (for example, **View Item if Contact**).

Filtering Messages by Logging Level

Message List messages are categorized by significance, or severity. The following table describes the three levels available in the Message List. The levels are listed in descending order of severity.

Logging Level	Icon	Description
Error		An error message is displayed when something is wrong in the process app that you need to fix.
Warning		A warning message is displayed to notify you of a potential problem or of a condition that could cause an undesirable or unexpected situation.
Info		An information message is logged to provide information about the check-in and *deployment* [page 683] processes.

To specify which messages are displayed in the Message List by logging level:

- Select or clear the appropriate check boxes in the **Show** section.

Debugging With Error and Warning Validation Messages

This section shows you how to work with error and warning validation messages to debug process apps.

The following is a sample warning message. (All validation messages appear in this format.)

Description	Name	Type	Design Element	Application/ Orchestration
State 'Closed' is unreachable	Closed	Regular State	CreditApprovalWorkflow	CreditApprovalApp

In this message, `State 'Closed' is unreachable` indicates that there is a problem with the regular state named `Closed` that is part of the *application workflow* [page 680] named `CreditApprovalWorkflow`, which is contained within the *application* [page 679] named `CreditApprovalApp`.

To go to the source of the message:

- If the **Type** column contains an entry other than `Workflow`, right-click anywhere in the message row, and then select **Go to Location** on the menu.

  SBM Composer opens the appropriate design element in the workflow editor and selects the element. The Property Editor for some of the design elements also opens.

In the example, SBM Composer opens `CreditApprovalWorkflow` (application workflow) in the workflow editor and selects the `Closed` state. In this case, the `Closed` state is not connected by a transition in `CreditApprovalWorkflow`. If the `Closed` state needs to be connected to another state, then you should fix the problem. If the `Closed` state does not need to be connected, then you can ignore the warning.

Determining the Cause of Failed Deployments

If a process app fails to deploy, an info (information) message is displayed in the Message List. For example, if the deployment operation for a process app named **`TestProcessApp`** fails, you will see the following message: `Deployment of 'TestProcessApp' has aborted`.

To find the reason for the failure:

1. Right-click the message, and then select **Go to Location**.

   The **Deploy Log** dialog box opens.

2. Look for any `ERROR` entries in the log.

Following are some typical reasons that process apps fail to deploy:

- An *endpoint* [page 684] is missing or is not configured correctly in the runtime environment.
- You do not have the proper permissions to deploy the process app.
- You used a name for an *application workflow* [page 680] or a *primary table* [page 692] that already exists in the repository.
- A *BPEL* [page 680] compilation error occurred.

Clearing the Message List

To clear the Message List:

- Right-click anywhere in the Message List, and select **Clear**.

  Validation messages clear when you *publish* [page 692] and deploy a process app, and deployment messages clear each time you validate a process app.

Using the Log Viewer

This section provides information on how to use the various features of the Log Viewer and some tips on locating and interpreting the messages that are displayed there.

The Log Viewer displays a list of messages that tell you what went wrong if a process app encountered an error or if it did not work as expected during runtime. The Log Viewer also contains general diagnostic messages, such as notifications of when a Web service is invoked or when the SBM Orchestration Engine is sending a message.

The Log Viewer only generates messages for workflow elements, including the workflows themselves (design elements), not for other elements such as tables or references.

Using the various features in the Log Viewer, you can specify which messages you want to see based on their type (user and technical) and logging level and when and where they occurred. You can also sort the messages and export them to a file.

User messages generally consist of SOAP messages and significant runtime events. Technical messages contain information about routine runtime activity and process flow.

For additional information about using the Log Viewer to troubleshoot orchestrations, refer to Troubleshooting Orchestrations Using the Log Viewer [page 573].

Opening the Log Viewer

To open the Log Viewer:

1. On the **Home** tab of the Ribbon, in the **Common Views** group, select the **Log Viewer** check box. Alternatively, press Ctrl+Shift+L.

   The Log Viewer opens.

2. If the **Log Viewer** check box is already selected and the Log Viewer is not visible, click the **Log Viewer** tab.

The following table describes the elements that appear in the Log Viewer.

Element	Description
Overview tab	Displays a list of all the applications and orchestrations in the process app and the workflows contained in them. (See Using Debug Logging [page 417].)
Details tab	Displays Log Viewer messages and filters the messages by logging level, by a selected *design element* [page 683], and by run. Also sorts the messages in a variety of ways, changes the way messages are presented, and locates one or more messages using the Find feature. (See Viewing Log Viewer Messages [page 419].)
Set filter	Opens the **Message Filter** dialog box, which filters Log Viewer messages by *environment* [page 684], timeframe, and type. (See Filtering Messages [page 420].) The messages are displayed on the Details tab of the Log Viewer.
Refresh	Updates the Log Viewer with messages logged since the last time you ran the process app based on the filters that you set in the **Message Filter** dialog box and on the **Details** tab. This button is not available until you deploy the process app.
Export	Displays the **Export log messages** dialog box, which lets you save the Log Viewer messages to an XML file. (See Exporting Messages [page 426].)

Using Debug Logging

To use the Log Viewer for debugging process apps, you must turn the Debug Logging feature on. You can only do this after you deployed the process app.

Note: If you do not turn on Debug Logging, only error messages are displayed in the Log Viewer. If you turn off Debug Logging, the only *new* messages that are displayed are error messages.

To turn Debug Logging on:

1. In the Log Viewer, select the **Overview** tab.

   A list of all the applications and orchestrations and the workflows contained in them is displayed.

2. Turn on Debug Logging for the applications and orchestrations in the process app as follows:

   a. Right-click the name of an *application* [page 679] or *orchestration* [page 690].

   b. Select **Debug Logging**.

[Debug Logging ON] appears to the right of the name of each application and orchestration. When Debug Logging is on, the option on the menu is checked.

Important: You should only turn on Debug Logging while you are developing and testing a process app. Running a process app with Debug Logging turned on causes the process app to run more slowly, so be sure to turn this feature off in the production environment when it is ready to be used by users.

To turn Debug Logging off:

1. In the Log Viewer, select the **Overview** tab.
2. Right-click the name of an application or orchestration for which Debug Logging is turned on.
3. Select **Debug Logging**.

   When Debug Logging is off, the option on the menu is not checked.

Overview Tab

The following table describes the elements that appear in the **Overview** tab and the three buttons that appear on the right side of the Log Viewer.

Element	Description
Hierarchical list	Displays a list of all the applications and orchestrations in the process app and the workflows contained in them.
Set filter	Displays the **Message Filter** dialog box, which filters Log Viewer messages by *environment* [page 684], timeframe, and type. (See Message Filter Dialog Box [page 420].) The messages are displayed on the **Details** tab of the Log Viewer.
Refresh	Updates the Log Viewer with messages logged since the last time you ran the process app based on the filters that you set in the **Message Filter** dialog box and on the **Details** tab. This button is not available until you deploy the process app.
Export	Displays the **Export log messages** dialog box, which lets you save the Log Viewer messages to an XML file. (See Exporting Messages [page 426].)

Generating Log Viewer Messages

To generate messages for the Log Viewer:

1. Create and deploy a process app, and then run it in the SBM User Workspace.
2. In SBM Composer, in the Log Viewer, click **Refresh** to display any new messages.

   If there are messages for any workflows in the process app, numbers other than `0` (zero) appear in brackets to the right of the name. The number of error messages appears in the first position, the number of warning messages in the second, the

number of info messages in the third, and the number of debug messages appears in the fourth position.

For example, `AnOrchestrationWorkflow [1/1/0/15]` indicates that this *orchestration workflow* [page 690] has 1 error message, 1 warning message, 0 info messages, and 15 debug messages. (This example assumes that you selected **User and Technical Messages** in the **Message Filter** dialog box. (See Filtering Messages [page 420].)

The messages are displayed on the **Details** tab.

Viewing Log Viewer Messages

In the Log Viewer, you can only view messages for the workflow that is selected in the **Details** tab.

To view messages for the selected workflow:

1. In the Log Viewer, click the **Overview** tab.
2. Select a workflow in the list, where one of the numbers in brackets is greater than `0` (zero).
3. Click the **Details** tab.

Details Tab

The following table describes the elements that appear in the **Details** tab of the Log Viewer and the three buttons that appear on the right side of the Log Viewer. Using this tab, you can filter Log Viewer messages by run, by logging level, and by selected *design element* [page 683]. Messages are displayed in the **Details** tab for the workflow that you selected on the **Overview** tab.

Element	Description
Date	Displays the date and time that the message was logged.
Run (column)	Displays the Item ID for the *application workflow* [page 680] (project), and the run number for an *orchestration workflow* [page 690].
Associated element	Displays the design element that is associated with the message.
Text	Displays the text of the message.
Run (menu)	Displays the run as the Item ID or run number along with the starting time and duration of each run. (See Filtering By Run [page 422].)
Error, Warning, Info, Debug	Displays messages by logging level. (See Filtering By Logging Level [page 422].)

Element	Description
Restrict by selection	Displays only messages for the design element that is selected in the workflow editor. (See Filtering By Selected Design Element [page 423].)
Set filter	Opens the **Message Filter** dialog box, which filters Log Viewer messages by *environment* [page 684], timeframe, and type. (See Message Filter Dialog Box [page 420].) The messages are displayed on the **Details** tab of the Log Viewer.
Refresh	Updates the Log Viewer with messages logged since the last time you ran the process app based on the filters that you set in the **Message Filter** dialog box and on the **Details** tab. This button is not available until you deploy the process app.
Export	Displays the **Export log messages** dialog box, which lets you save the Log Viewer messages to an XML file. (See Exporting Messages [page 426].)

Filtering Messages

Log Viewer filtering tools let you display only the messages that you want to see on the **Details** tab of the Log Viewer. You can filter messages based on the following criteria:

- Environment
- Type (user and technical messages)
- Timeframe
- Run
- Selected design element
- Logging levels (significance, or severity)

Use the **Message Selector** dialog box to filter Log Viewer messages by environment, type, and timeframe. Use the **Details** tab of the Log Viewer to filter Log Viewer messages by run, logging levels, and selected *design element* [page 683].

Note: Any filters that you set in the **Message Filter** dialog box remain in effect until you change them, even after you close and then reopen a process app in SBM Composer.

Message Filter Dialog Box

The following table describes the elements that appear in the **Message Filter** dialog box.

Element	Description
Environment	Displays messages for the selected environment. (See Filtering By Environment [page 421].)

Element	Description
Since last publish	Displays messages since the process app was last published. (See Filtering By Timeframe [page 422].)
Range	Displays messages for a specified date and time range. (See Filtering By Timeframe [page 422].)
User messages only	Displays user messages only. (See Filtering By Type [page 421].)
User and technical messages	Displays both user and technical messages. (See Filtering By Type [page 421].)
Show messages from failed runs only	Select this check box if you only want to see messages about orchestration workflows whose execution failed. The reason for failure is typically an unhandled fault.

Filtering By Environment

If you ran a process app in more than one runtime environment, you can choose the environment from which you want to view messages in the **Message Filter** dialog box.

To choose the runtime environment:

1. In the Log Viewer, click **Set Filter**.

   The **Message Filter** dialog box opens.

2. Select an environment from the **Environment** list.
3. Click **OK**.

Filtering By Type

The two types of Log Viewer messages are *user* and *technical*. User messages are most useful to business users for debugging process apps. Technical messages are more appropriate for advanced users such as developers, integrators, and support personnel. Usually, you should be able to find the cause of a problem by reading user messages. However, in some cases, you might have to look at technical messages as well.

To display user and technical messages:

1. In the Log Viewer, click **Set Filter**. The **Message Filter** dialog box opens.
2. Click either **User messages only** or **User and technical messages**.

   Tip: It is recommended that business users select **User messages only**.

3. Click **OK**.

Filtering By Timeframe

You can choose whether you want to see all the messages since you last published a process app, or you can choose to display message within a specified date and time range.

To display messages since you last published a process app:

1. In the Log Viewer, click **Set Filter**.

   The **Message Filter** dialog box opens.

2. Click **Since last publish**.

3. Click **OK**.

To display messages within a specific date and time range:

1. In the Log Viewer, click **Set Filter**.

   The **Message Filter** dialog box opens.

2. Set the **From** and **To** date and time as follows:

   a. In the date box, click the down arrow, and then select a date in the calendar.

   b. In the time box, click the up or down arrow to set the time earlier or later, respectively.

3. Click **OK**.

Filtering By Run

For application workflows, a *run* is logged each time an item is submitted into a project. The run is the Item ID assigned to the submitted item. For orchestration workflows, a *run* is logged each time an *orchestration workflow* [page 690] is invoked by a process app. You can choose to display messages for a single run or for all runs for the workflow that is selected on the **Overview** tab. The most recent run is the one with the highest number.

To specify the messages that are displayed by run:

1. In the Log Viewer, select the **Overview** tab.

2. Select a workflow in the list.

3. Select the **Details** tab.

4. On the **Run** menu, select a single run or **All Runs**.

Filtering By Logging Level

Log Viewer messages are categorized by significance, or severity. The following table describes the four levels available in the Log Viewer. The levels are listed in descending order of severity. As the severity decreases, the number of messages increases.

Note: The Common Log settings in Application Administrator determine which levels of messages are generated for an *application* [page 679] or *orchestration* [page 690]. If the setting for a particular workflow is less than info, you will not be able to see all logging levels in the Log Viewer for that workflow. (See *SBM Application Administrator Guide*.)

Logging Level	Icon	Description
Error		An error message is logged when the execution of an *application workflow* [page 680] or an *orchestration workflow* [page 690] fails or when a process app fails or is stopped. The message explains the reason for the error and might suggest a possible solution, and an icon is displayed in the workflow editor next to the *design element* [page 683] where the error occurred.
Warning		A warning message is logged to notify you that an operation completed, but there were potential problems or unexpected results.
Info		An info message is logged to provide information about significant runtime events.
Debug		A debug message is logged to provide detailed information about the flow through the system and about routine operations.

To specify which messages are displayed based on logging level:

1. In the Log Viewer, select the **Details** tab.
2. Select or clear the **Error**, **Warning**, **Info**, and **Debug** check boxes.

Filtering By Selected Design Element

You can choose to see messages for a single element only. For example, you could select a transition in an *application workflow* [page 680] or a **Decision** step in an *orchestration workflow* [page 690]. You can select an element using App Explorer or you use the Log Viewer.

To view messages for a single element using App Explorer:

1. In App Explorer, select an application workflow or an orchestration workflow.
2. In the workflow editor that opens, select an element.
3. In the Log Viewer, select the **Details** tab.
4. Select the **Restrict by selection** check box.

You can also select an element using the Log Viewer.

To view messages for a single element using the Log Viewer:

1. In the Log Viewer, locate a message associated with a specific element. (The **Associated elements** column should contain an entry other than `Workflow`.)
2. Right-click anywhere in the message row, and then select **Go to Location**.

   If the application workflow is not already visible in the workflow editor, SBM Composer opens the appropriate workflow in the editor and selects the associated element.

3. In the Log Viewer, select the **Restrict by selection** check box.

Troubleshooting the Log Viewer

If you do not see any messages in the Log Viewer, or if you see only error messages when you expect other levels as well, check for the following:

- Did you turn on Debug Logging for the applications and orchestrations that contain the workflows you want to debug?
- Have you run the process app since you turned on Debug Logging?
- Did you click the **Refresh** button after you ran the process app?
- Is the **Message timeframe** setting in the **Message Filter** dialog box **Since last publish**, or were the messages generated within the specified range?
- If the process app contains an asynchronous *orchestration workflow* [page 690], did you wait for it to run?
- If the process app contains an orchestration workflow that is invoked by external events, did you send the correct authentication credentials, such as user ID and password?

Sorting Messages

You can sort Log Viewer messages in a variety of ways. Following are five ways to change how messages are displayed.

To sort messages by date and time:

- Click the **Date** column heading to sort the messages in ascending or descending order by date and time, as indicated by the direction of the arrow.

To sort messages by Item ID or run number:

- Click the **Run** column to group the messages by run in ascending or descending numerical order, as indicated by the direction of the arrow.

To sort messages by associated element:

- Click the **Associated element** column to group the messages by the first letter of the name of the associated element in ascending or descending alphabetical order, as indicated by the direction of the arrow.

To sort message text in alphabetical order:

- Click the **Text** column to group the message text by the first letter of the message in ascending or ascending alphabetical order, as indicated by the direction of the arrow.

To display the message text in a separate row below the Date, Run, and Associated element columns:

- Right-click any of the messages, and then select **Show Details**.

  The second row can contain up to five lines of text.

Viewing Messages in a Dialog Box

Sometimes Log Viewer messages are too long to fit in the **Text** column. You can display the entire contents of the message in the **Message Detail** dialog box.

To display the contents of a message in a dialog box:

1. Right-click the message, and then select **Show Message**.

   The **Message Detail** dialog box opens.

2. Click the **Next** and **Previous** buttons to move up or down the list of messages according to the specified sort order. (See Sorting Messages [page 424].)

Message Detail Dialog Box

The following table describes the elements that appear in the **Message Detail** dialog box.

Element	Description
Previous	Displays the contents of the previous message on the list.
Next	Displays the contents of the next message on the list.
Word wrap	Wraps the lines of text to fit the width of the dialog box. If the message includes line breaks, clearing this check box enables you to easily scan multiple lines.
Close	Closes the dialog box.

Finding Messages

The Log Viewer provides a feature that lets you locate messages quickly.

To use the Find feature:

1. Right-click any message on the list, and then select **Find**.

   The Find Bar appears above the message list.

2. In the **Look for** box, type any text or numbers that you want to search for, or click the arrow on the **Look for** box to use previous search text.

3. Optionally, select any or all of the follow options from the **Options** menu:

   a. **Search Details.** Select this option if you want to search the **Text** column only.

   b. **Match Case.** Select this option if you want the search results to exactly match the capitalization of the text in the **Look for** box.

   c. **Match Whole Words.** Select this option if you want the search to exactly match the word or words in the **Look for** box, not parts of words.

4. Click **Find now**.

To clear the Look for box and the search results and to display all the messages in the current view:

- Click **Clear**.

Exporting Messages

You can export and save the current Log Viewer messages to an XML document for later reference or for use by support personnel. All messages are saved, that is, any filters that you specified in the Log Viewer are not applied.

To export the messages for all of the application workflows and orchestration workflows:

1. In the Log Viewer, click the **Overview** tab.
2. Click **Export**.

   The **Export log messages** dialog box opens with a default file name of `DebugLogSnapshot`. You can change the name.
3. Using the **Saved in** menu, navigate to the location where you want to save the file, and then click **Save**.

   The file is saved in XML format along with a cascading style sheet (`.css`) and an XSL style sheet (`.xsl`) file. You can view the file in a Web browser or in any XML editor.

To export the messages for a particular *application workflow* [page 680] or *orchestration workflow* [page 690]:

1. In the Log Viewer, on the **Overview** tab, select a workflow.
2. Click the **Details** tab.
3. Select any one of the messages in the list.
4. Click **Export**.

   The **Export log** messages dialog box opens with a default file name of `DebugLogMessages`. You can change the name.
5. Using the **Saved in** menu, navigate to the location where you want to save the file, and then click **Save**.

   The file is saved in XML format along with a cascading style sheet (`.css`) and an XSL style sheet (`.xsl`) file. You can view the file in a Web browser or in any XML editor.

Exceeding the Message Limit

The Log Viewer can display a maximum of 20,000 messages. If you exceed this limit, you will receive an error message.

To limit the number of messages to the allowable number after you receive an error message:

1. In the error message box, click **OK**.
2. Click **Set filter**.

3. Do one or both of the following:
   - If **User and technical messages** is selected, select **User messages only**.
   - Reduce the range in the **Timeframe** section.
4. Click **OK**.

Using Application Administrator to Filter and View Log Viewer Messages

If you have access to the Common Log in Application Administrator, you can see all of the messages that are logged for the applications and orchestrations in a process app. Using the Common Log settings, you can specify the types of messages, by logging level, that are generated for a particular *application* [page 679] or *orchestration* [page 690]. If Verbosity is set to all, the log shows fatal and trace messages in addition to error, warning, info, and debug. See *SBM Application Administrator Guide* for additional information.

Using the Validation Output Pane

Scripts are shown under the **AppScripts** heading in App Explorer. After you add a script or change a script in the script editor, right-click in the script editor and then select **Validate**.

- If the script is valid, the Validation Output pane displays "This script is valid."
- If the script has errors, the Validation pane displays "INVALID SCRIPT" and a description of the error or errors that were found.

Debugging for Development and Support

Advanced options for debugging, such as setting the logging level to trace or fatal, are available in Application Administrator. Refer to the *SBM Application Administrator Guide* for more information.

Part 4: Working with Orchestrations

You use orchestrations to link *Web services* [page 701] into workflows that are executed in response to events. After you create an *orchestration workflow* [page 690] in SBM Composer, the workflow is saved as a *Business Process Execution Language (BPEL)* [page 681] file and is executed by the BPEL *target server* [page 699] defined in Application Administrator.

Audience

This section is intended for designers who have the following knowledge and skills:

- Familiarity with Web services, including the SOAP specification, Extensible Markup Language (XML), and Web Services Description Language (WSDL)
- Knowledge of basic programming concepts such as branching and looping
- The ability to arrange the elements of orchestration workflows in logical sequences that work within or interact with an *application* [page 679]

Note: Support for development efforts writing Web services is provided by Professional Services. Questions regarding the use of Web services operations in orchestration processes as documented are handled by Serena Customer Support.

Overview

This section contains the following information on creating and managing orchestrations in SBM Composer:

Chapter 25: Orchestration Concepts

This section describes the concepts related to orchestrations created in SBM Composer.

About Orchestration Workflows

The primary purpose of an *orchestration workflow* [page 690] is to enable the use of *Web services* [page 701] for coordinating the interaction between an *application workflow* [page 680] and one or more external systems. This lets the application workflow present data that can be exchanged and modified in these external systems. Orchestration workflows can also perform modifications on the data that flows within the application workflow.

Note: Orchestration workflows are used to automate processes, while application workflows are generally manual processes for users. Typically, users never see a process controlled by an orchestration workflow, but they must interact with an application workflow.

An orchestration workflow is an arrangement of control flow structures, Web services, and data elements.

- Control flow structures enable an orchestration workflow to make calculations, decide between two possible sets of instructions, handle exceptions, and so on. In SBM Composer, these structures are known as *steps*, and they are listed on the **Step Palette** of the orchestration workflow editor.
- Web services can be called by means of the **Service** step, which is also listed on the **Step Palette**. Web services let orchestration workflows query and update data in application workflow items and in external products.
- Data elements include data sent with the *event* [page 685], data expected by or returned by a Web service, and working data created for temporary use during the

execution of an orchestration workflow. Data elements are mapped in the step Property Editor.

Comparing Synchronous With Asynchronous Orchestration Workflows

Orchestration workflows can run *synchronously* or *asynchronously*. An *orchestration workflow* [page 690] that runs synchronously immediately returns the data to an *application* [page 679]. However, an orchestration workflow that runs asynchronously can keep going, independently of the application that contains it.

The following table lists the differences between synchronous and asynchronous orchestration workflows.

Synchronous Orchestration Workflow	Asynchronous Orchestration Workflow
Used when an immediate reply is required. For example, a synchronous orchestration workflow might be used when certain data is needed before a user can transition to the next step in the *application workflow* [page 680], or when a Web service call is expected to return a quick reply such as a stock quote or a weather forecast.	Used when an immediate reply is not required. For example, an asynchronous orchestration workflow might be used when the orchestration contains a long-running program, when the data that is returned by the Web service is required in a later step in the application workflow, or when the orchestration workflow performs some unrelated task such as sending an e-mail.
Requires the application workflow to wait for a reply before it can continue.	Does not require the application workflow to wait for a reply. Runs independently of the application workflow.
Can be used for transition actions and for state actions.	Can only be used for transition actions.
The SBM Application Engine invokes the SBM Orchestration Engine directly. The orchestration workflow can only be called by the SBM Application Engine.	The SBM Application Engine calls the *Event Manager* [page 685], which then invokes the SBM Orchestration Engine based on event mappings in the *event definition* [page 685].

In the exercises in Chapter 29: Orchestration Tutorials [page 563], you will create a process app that contains both synchronous (event with reply) and asynchronous (event without reply) orchestration workflows. When you run the process app, you will be able to see the differences in behavior between them.

About Working Data

You can see the working data hierarchy on the **Data Mapping** tab of the orchestration workflow Property Editor. Because the working data is available throughout the *orchestration workflow* [page 690], it is displayed when no specific step is selected.

Working data is composed of individual data elements that you create and delete. The possible types are:

- Private Simple: A type such as Integer or String.
- Library Type: A system type such as User, Field, or Privilege.
- Private Complex: A structure made of some combination of the other two types. This type has child data elements.

Working data is used in orchestration workflows. For example, you might want to temporarily store the value returned by a Web service or a loop counter variable. As with the inputs to *Web services* [page 701], you can define source elements and default values for working data.

Important: Working data elements cannot be named `input` or `output`.This includes any variations in capitalization such as `Input` or `Output`.

About Data Mapping

As you create orchestrations in SBM Composer, you find that you need to pass data to *Web services* [page 701], use the return values from Web services, or store data values in temporary locations.

In SBM Composer, you use data mapping to arrange the inputs and outputs of Web services. **Data Mapping** is one of the tabs that is displayed in the orchestration workflow Property Editor when you select an *orchestration workflow* [page 690] in App Explorer or when you select a **Service** step in an orchestration workflow. (As with all Property Editor tabs, the content of the **Data Mapping** tab depends on what is selected.) When you define inputs, you can either use the mappings suggested by SBM Composer or specify other mappings.

You can do any of the following tasks from the **Data Mapping** tab:

- Choose an input suggested by SBM Composer (suggested mappings).
- Open the **Select a Source** tool, in which you select an input from a hierarchical list of choices.

  Note: The outputs from a Web service are available as source elements only for subsequent items in the orchestration workflow.

- Display advanced options with which you map incompatible types or multiple items in the **Select a Source** tool.
- Use data that accompanies the *event* [page 685] that triggered the orchestration workflow.
- Clear mappings for the selected item.
- Set the display of data elements to show a red highlight on the icons of required items.
- Switch between mapping mode and properties mode. In properties mode, you can view and edit the lowest-level details for a working data element or step input, including its namespace and type.

Note: Elements on the **Data Mapping** tab that have a lock icon on them are external types that are defined by a Web service; they are not defined in the process app. These external types are shown in the Type Library Editor [page 462]. You cannot change the type of locked elements or change their child elements.

About Complex Types and Namespaces

Complex types differ from simple types in that complex types can have child data elements and attributes, and simple types cannot. Complex types can be named or anonymous. All types except anonymous types are identified by their name and target namespace. Anonymous complex types do not have names.

You can view the name and namespace for a data element by switching from mapping mode to properties mode on the **Data Mapping** tab in the Property Editor for an *orchestration workflow* [page 690] or **Service** step. To switch modes, click the vertical bar near the right side of the tab.

You can use the Select Library Type Dialog Box [page 461] to associate a named type with a data element. This dialog box contains the named types defined by the *Web services* [page 701] that were imported into the orchestration workflow. In the orchestration workflow Property Editor, the **NamedType**, **NamedType Namespace**, and **Namespace** are read-only. You cannot manually add a child element to a data element with a named type, even if it is a complex type.

Note the following points about anonymous complex types:

- The default target namespace is `http://<workflow name>`.
- Unless it is changed explicitly, a child data element continues to have the default namespace, even if the namespace of its parent data element is changed explicitly.

- If you want a child data element to store data from a Web service response, its namespace must be the same as the namespace in the Web service schema. In other words, it must be the same as the `targetNamespace` of the schema element defined in your WSDL file. The schema element must contain the complex type from which you are trying to map.

Example: Named Type

A complex data element associated with a named type inherits its namespace and children data elements from the named type. For example, suppose you do the following:

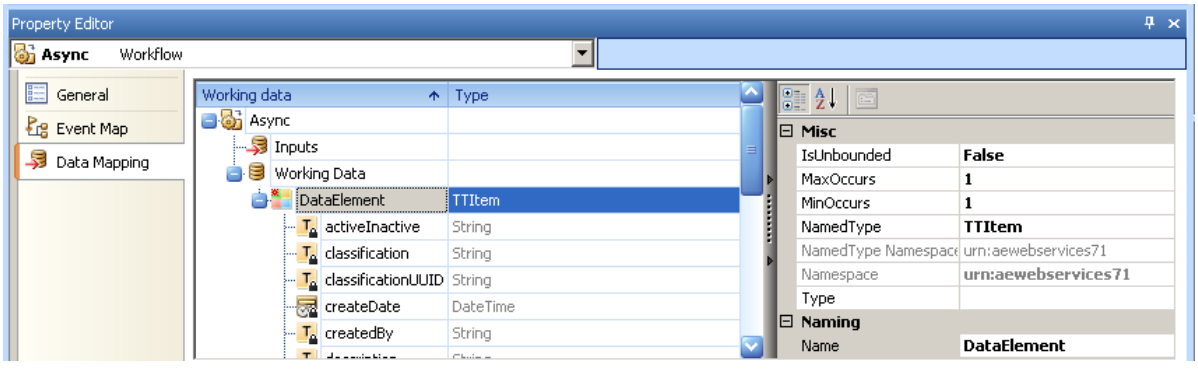
1. Add the **sbmappservices72** Web service to the process app. To do so, right-click the **Web Services** heading in App Explorer, and select **Add New Service**. In the **Web Service Configuration** dialog box that opens, navigate to the `sbmappservices72.wsdl` file, and then click **OK**. This file is in the *`installDir`*`\Application Engine\webservices\bin` directory.

2. Add a data element to the **WorkingData** node on the **Data Mapping** tab in the Property Editor for the orchestration workflow.

3. Right-click the new data element, point to **Type[String]**, and then select **Select from Type Library**.

4. In the Select Library Type Dialog Box [page 461] that opens, find and select **TTItem**) and then click **OK**.

   As shown in the following illustration, the **Namespace** value is inherited from the **NamedType Namespace** value.



Example: Anonymous Type

For a complex data element with an anonymous type, you must create the structure manually and specify the correct namespace. You get the information you need from the WSDL file for the Web service.

The following illustration of shows part of a WSDL file. This file includes the target namespace (`urn:SerenaSampleTickerService`) and some of the elements that can be added to the structure (`GetBuyRating`, `GetBuyRatingResponse`, `GetTickerSymbol`).

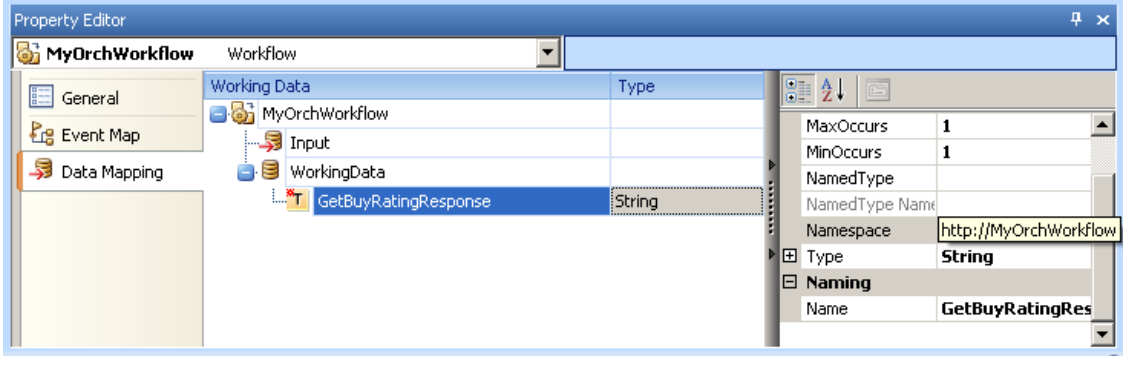
```
<?xml version="1.0" encoding="UTF-8" ?>
- <wsdl:definitions targetNamespace="urn:SerenaSampleTickerService" xmlns:tns="urn:SerenaSampleTickerService"
    xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/" xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
    xmlns:http="http://schemas.xmlsoap.org/wsdl/http/" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:s="http://www.w3.org/2001/XMLSchema">
  - <wsdl:types>
    - <s:schema attributeFormDefault="unqualified" elementFormDefault="qualified" targetNamespace="urn:SerenaSampleTickerService">
      - <s:element name="GetBuyRating">
        - <s:complexType>
          - <s:sequence>
              <s:element maxOccurs="1" minOccurs="0" name="symbol" type="s:string" />
            </s:sequence>
            <s:attribute name="type" type="s:string" />
          </s:complexType>
        </s:element>
      - <s:element name="GetBuyRatingResponse">
        - <s:complexType>
          - <s:sequence>
              <s:element maxOccurs="1" minOccurs="0" name="GetBuyRatingResult" type="s:string" />
            </s:sequence>
          </s:complexType>
        </s:element>
      - <s:element name="GetTickerSymbol">
```

Suppose you want to store the `GetBuyRatingResponse` value and use it later in the orchestration workflow. You would perform the following steps:

1. Add the **SerenaSampleTickerService** Web service to the process app. To do so, right-click the **Web Services** heading in App Explorer, and then select **Add New Service**. In the **Web Service Configuration** dialog box that opens, in the **WSDL** box, type `http://<localhost>:8085/ticker/services/SerenaSampleTickerService?wsdl` and then click **OK**.

2. Add a **Service** step to the orchestration workflow. On the **General** tab in the Property Editor for the step, select **SerenaSampleTickerService** from the **Service** list, and select **GetBuyRating** from the **Operation** list.

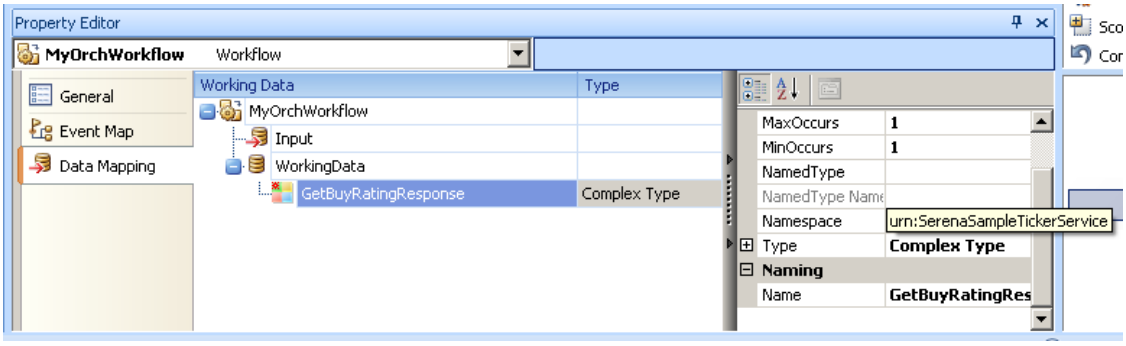
3. Add a new data element to the **WorkingData** node and name it **`GetBuyRatingResponse`**.



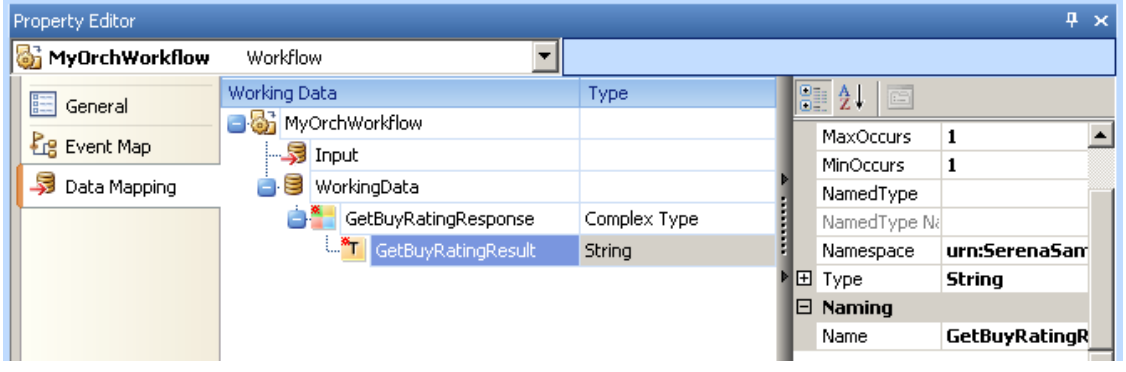
   The default type is **String**, and the default namespace is **http://MyOrchWorkflow** (the name of the orchestration workflow).

4. Change the type to **Complex**. To do this, right-click **GetBuyRatingResponse**, point to **Type[String]**, and then select **Private Complex**.

5. Change the namespace to the `targetNamespace` shown in the WSDL file (`urn:SerenaSampleTickerService`). To do this, simply copy and paste the namespace into the **Namespace** box.



6. Add a child data element and name it **`GetBuyRatingResult`**.



The default type is **String**, and the namespace was inherited from the parent data element. The namespace matches the `targetNamespace` in the WSDL file.

Important: If the namespace does not match the `targetNamespace` in the WSDL file, you must change it manually.

About Events

An *event* [page 685] signals a meaningful change from a SBM *application* [page 679] or an external product. For example:

- An issue defect management application in SBM raises an event every time a user submits a new defect.
- A software build product raises an event every time a build completes.
- Salesforce.com raises an event every time a potential customer is initally contacted.

Any SBM application or external product that is capable of calling a Web service can raise events in SBM by calling a corresponding Web service. After an event is raised, the *Event Manager* [page 685] receives it, and then calls the SBM Orchestration Engine to execute the asynchronous *orchestration workflow* [page 690] linked to the event. The orchestration workflow could update a SBM item, create a new SBM item, and so on.

The orchestration workflow provides the integration point between a SBM application and an external product, or two SBM applications in different process apps. In the first case,

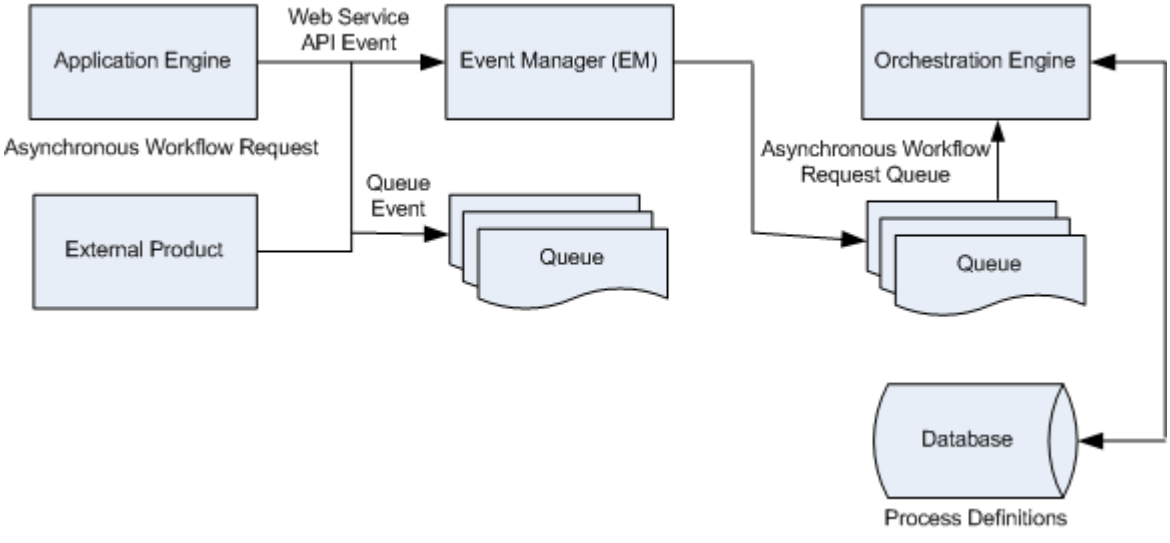
the orchestration workflow can be initiated from either the application or the external product. In the second case, the orchestration workflow can be initiated from either application. Something happens in one that raises an event, and the orchestration workflow runs in response to the event and updates the other.

An event can be raised in the following ways:

- From the SBM Application Engine on a transition in an *application workflow* [page 680]. You use the **Action Wizard** to create this type of event. For more information, see Action Wizard [page 374].
- From an external product using any of the following:
  - Event Manager Web service API
  - E-mail message
  - JMS queue

  The first option is described in this topic and in various topics in Part 4: Working with Orchestrations [page 429]. The second and third options are described in Part 5: Advanced Orchestration Topics [page 581].

The following diagram illustrates the way an event is processed.



Remember: Events are relevant for asynchronous orchestration workflows only. For information about the differences between asynchronous and synchronous orchestration workflows, see Comparing Synchronous With Asynchronous Orchestration Workflows [page 432].

In addition to initiating an orchestration workflow, an event sends data to that orchestration workflow. For an event raised from a SBM application, you can select which fields from the primary *application table* [page 679] are to be sent with the event, and can define additional data to be sent with the event. You do this on the **Event** tab, which you access by clicking **Event** under the **Orchestration Links** heading in App Explorer.

Because orchestration workflows that are invoked from an event raised from an external product do not have inputs corresponding to fields in SBM applications, you cannot define them by selecting them from a list, as you would for events raised from an application.

Instead, you must import an existing *event definition* [page 685] or create a new custom event definition using SBM Composer. Existing process event definitions could have been exported from other process apps or created independently for particular external products. See About Application Links and Event Definitions [page 439] for more information.

If you want the event definition to be used by an asynchronous orchestration workflow in another process app, lock the event definition before you export it, using the Event Editor [page 448]. The event definition includes an event type (contructed from the application, state, and transition names) and the extension data fields that will be sent with the event. If you lock the event definition, it becomes portable, because these names will not change in the event definition, even if they change in the original process app.

Important: See Raising External Events [page 556] for a use case and for a step-by-step description of how you raise events from an external product.

About Application Links and Event Definitions

An *event definition* [page 685] is a specific format that lets SBM applications and external products declare the events they can raise. An event definition also lets receiving applications understand the events so they can respond to them. Event definitions are listed under the **Application Links** heading in App Explorer.

An event definition defines the values for the events an external product can raise. These values are named `EventType`, `ObjectType`, `Product`, `ProductVersion`, and `ProductInstance`. In addition, an event definition defines the structure of any additional data that is sent with the event.

SBM uses the event values defined by the event definition to construct an event map that is deployed to the *Event Manager* [page 685]. SBM uses the event data defined by the event definition to define orchestration workflows that are specific to the events. At runtime, the Event Manager receives the event, and if the event is in a deployed event map, it invokes the associated asynchronous *orchestration workflow* [page 690], passing on the event definition data.

Note: Orchestration links, not event definitions, are used to define events for synchronous orchestration workflows. For more information, see About Orchestration Links [page 440].

You use event definitions in the following scenarios:

- Defining events from application workflows. You use the **Action Wizard** to update an existing event definition or create a new event definition. The **Action Wizard** guides you through the process of defining an orchestration workflow *action* [page 679] associated with a transition. The action invokes an asynchronous orchestration workflow after the transition is executed. For more information, see Action Wizard [page 374].

- Defining events so other process apps know how to respond to the events. The event information is exported as an event definition (`.mtd`) file so that it can be associated with an asynchronous orchestration workflow in another process app.

  You create event definitions by clicking **Export event definition** on the **General** tab of the event definition Property Editor.

- Raising external events from an external product. Event definitions that perform this function define custom, system-specific events, and are known as *custom event*

definitions. The event information is exported as a Web Service Definition Language (`.wsdl`) file that the external product uses to call *Web services* [page 701] that raise events.

There are two ways you can create custom event definitions:

- Creating a event definition in SBM Composer. For instructions, see Creating a New Custom Event Definition [page 475].
- Importing the `.mtd` (or `.wsdl`) file for an existing custom event definition. This file could have been created manually, or could have been created new in SBM Composer and then exported to be used for other process apps. For instructions, see Importing an Event Definition File for a New Custom Event Definition [page 476].

When you click **Export external event WSDL** on the **General** tab of the event definition Property Editor, SBM Composer creates an external event `.wsdl` file with the appropriate inputs that you can use for this purpose. After you export this `.wsdl` file, consult the documentation for the external system to determine how to call an operation on the Web service to raise the event.

Important: See Raising External Events [page 556] for a use case and for a step-by-step description of how you use an event definition to raise events from an external product.

About Orchestration Links

An orchestration link defines the data that is sent to an *orchestration workflow* [page 690] when a transition is executed or a state is reached in an *application workflow* [page 680]. The values from the *primary table* [page 692] fields that you select and any additional data items you define are passed to the orchestration workflow that you select when you create the orchestration link. For synchronous orchestration workflows, you also select primary table fields whose values should be returned from the orchestration workflow.

Remember: Because changes to an orchestration link propogate to the inputs and outputs of an orchestration workflow, the orchestration link is the primary area in which the inputs and outputs are defined.

Orchestration links are displayed under the **Orchestration Links** heading in App Explorer.

- **Event** is added automatically. This is actually an event definition for asynchronous orchestration workflows that are called from an application workflow when a transition is executed. This event definition can be exported and then imported into other orchestrations so that they can handle the events that the *application* [page 679] raises. These orchestrations do not have to be in the same process app.

  Note: You can also use a custom event definition to define the data that is sent to an asynchronous orchestration workflow. For more information, see About Application Links and Event Definitions [page 439].

- Other orchestration links are for synchronous orchestration workflows that are called from an application workflow when a transition is executed or a state is reached. These orchestration links are typically added when you create a new synchronous workflow through the Action Wizard [page 374].

The orchestration link is displayed in the **Event definition** field on **General** tab of the orchestration workflow Property Editor.

Important: Synchronous orchestration workflows are not associated with external events, and therefore do not have event definitions. You cannot define a synchronous orchestration workflow using an event definition.

About the Step Palette

One of the things that distinguishes the use of *Web services* [page 701] in orchestrations, rather than in applications, is the availability of control flow structures. These control flow structures make it possible to design an *orchestration workflow* [page 690] that can branch based on a value, make calculations based on returned results, iterate through a set of data, and perform many other tasks.

The control flow structures are available on the area known as the **Step Palette**. Each item in the **Step Palette** is known as a *step*.

The following table describes the steps on the **Step Palette**.

Step	Purpose
Calculate	Performs a calculation or sets values on data.
Decision	Inserts branches into the workflow.
ForEach	Cycles through all members of a complex data element.
While	Repeats during the time a certain condition is true.
Service	Inserts a Web service into the workflow.
Scope	Creates a scope, that is, a separate block of steps that can be treated as a unit.
Compensate	Defines steps that can compensate for a problem.
Throw	Defines steps to throw an exception.

About Scope, Compensate, and Throw

The **Scope**, **Compensate**, and **Throw** steps in the **Step Palette** deserve special mention. These steps are related and are used to create a structure for handling faults that occur during Web-service execution.

- The **Scope** step makes it possible to have a FaultHandler and a CompensationHandler.
- The **Compensate** step is designed to work with a CompensationHandler or FaultHandler to roll back actions from an entire scope. You can use the **Compensate** step in one of two ways:
  - If you name a scope explicitly in the step Property Editor, then only the named scope's CompensationHandler is invoked.

  - If you do not name a scope in the step Property Editor, then every CompensationHandler for scopes enclosed immediately within the scope of the **Compensate** step is invoked, in reverse order of execution.

- The **Throw** step is used inside a nested scope to pass the exception or fault to the outer scope.

Note: The values of working data are not affected by scopes. Creating a scope does not create an area where working data is only visible within that scope. Working data is global to an orchestration and is visible across all scopes.

Note: For detailed information about these steps, see Using the Scope, Throw, and Compensate Steps to Handle Faults From Web Services [page 498].

About the Expression Editor

Many of the steps on the **Step Palette** include areas that let you define an arithmetic or logical expression. SBM Composer contains a feature to help you write these expressions. This feature is called the *expression editor*.

Expressions might require a long string such as "EventNotice.Extension.ActiveInactive." To reduce typing errors, the expression editor helps you complete these long strings.

The expression editor behaves similarly to the feature in word processors that suggests potential word choices when you begin typing. The expression editor tries to fill in the expression based on what it knows about the hierarchies of working data, *Web services* [page 701], and process app events.

You can see the expression editor by creating a **Calculate** step in an *orchestration workflow* [page 690] and then typing a letter into either the **Target** or **Expression** section on the **Options** tab of the step Property Editor. The expression editor tries to match that letter, showing you all available choices.

When you get to the end of a structure element, such as EventNotice, press the period key. The expression editor shows you the choices available in the next level of the hierarchy.

The expression editor also recognizes the XPath functions that are available.

For more information, see the following topics:

Using Data Mapping [page 463]

Using the Step Palette [page 477]

Supported XPath Functions [page 442]

Supported XPath Functions

Some of the steps in the **Step Palette** let you define expressions for a calculation or value assignment. In these expressions, you can use any of the functions or operators that are available in the step Property Editor.

The functions in the **Functions** list in the step Property Editor are XPath functions. Only XPath 1.0 functions are currently supported. XPath 2.0 functions are not supported.

Note: Some function names in SBM Composer differ from the names in the XPath standard. For example, `NORMALIZESPACE()` is the SBM Composer name for the `normalize-space()` XPath function.

The following table describes the supported XPath 1.0 functions.

Function	Description
BOOLEAN()	Returns a Boolean value for a number, string, or array. Numbers that are not equal to 0 and strings that are not null or empty always return `true`. For example, `BOOLEAN(0)` returns `false`, and `BOOLEAN("false")` returns `true`.
CEILING()	Returns the smallest integer that is greater than or equal to the number argument. For example, `CEILING(2.35)` returns `3`.
CONCAT()	Returns the concatenation of two or more strings. For example, `CONCAT("day","light")` returns `"daylight"`.
CONTAINS()	Returns `true` if the first string contains the second string. Otherwise, returns `false`. For example, `CONTAINS("XPath","Path")` returns `true`.
COUNT()	Returns the number of elements in an array. For example, suppose you have a `Files[]` array under an `Issues` data element. Each array element in the array stores a file name. `COUNT(Issues.Files)` returns `5`, because there are five array elements in the array.
FLOOR()	Returns the largest integer that is less than or equal to the number argument. For example, `FLOOR(2.35)` returns `2`.
LAST()	Returns the last element in an array. For example, suppose you have a `Testers[]` array under a `Users` data element. Each array element in the array stores the name of a software tester. `Users.Testers[LAST()]` returns `Jim Wilson`, because his name was stored in the last array element.
NORMALIZESPACE()	Returns an argument string after removing leading and trailing spaces and replacing each sequence of spaces with a single space. For example, `NORMALIZESPACE(" 555-1212 ")` returns `"555-1212"`.
NOT()	Reduces an argument to a Boolean expression, and then returns the opposite value. For example, `NOT(false())` returns `true`.

Function	Description
NUMBER()	Returns the numeric value of an argument. The argument can be a Boolean, string, or array. For example, `NUMBER("500")` returns `500`.
POSITION()	Returns the index position of an array that is being processed. For example, `testcase[POSITION()<=2]` selects the first two test cases.
ROUND()	Returns a number that is the nearest integer to the specified value. If there are two such numbers, the greater one is returned. For example, `ROUND(5.24)` returns `5`, and `ROUND(6.5)` returns `7`.
STARTSWITH()	Returns `true` if the first string starts with the second string. Otherwise, returns `false`. For example, `STARTSWITH("XPath","XP")` returns `true`.
STRING()	Returns the string value for a Boolean or number. For example, `STRING(1117)` returns `"1117"`.
STRINGLENGTH()	Returns the number of characters in a string. For example, `STRINGLENGTH("HOLIDAY")` returns `7`.
SUBSTRING()	Returns a substring from the starting position to the provided length. The index of the first character is 1. If the length argument is not provided, returns the substring from the starting position to the ending position. (If two arguments are provided, they represent the starting index and the length. If only one argument is provided, it represents the starting index.) For example, `SUBSTRING("Salesperson",1,3)` returns `"Sal"`, and `SUBSTRING("Salesperson",4)` returns `"esperson"`.
SUBSTRINGAFTER()	Returns the remaining characters in the first string after the second string occurs in it. If the first string does not contain the second string, an empty string is returned. For example, `SUBSTRINGAFTER("555-1212","-")` returns `"1212"`.

Function	Description
SUBSTRINGBEFORE()	Returns the characters in the first string before the second string occurs in it. If the first string does not contain the second string, an empty string is returned. For example, `SUBSTRINGBEFORE("555-1212","-")` returns `"555"`.
SUM()	Returns the sum of the numeric values of each element in the array. For example, suppose you have a `SalesTax[]` array under a `Taxes` data element. Each array element in the array stores the sales tax amount from a transaction. `SUM(Taxes.SalesTax)` returns `100.00`, because there are four array elements in the array, with values of `20`, `40`, `25`, and `15` respectively.
TRANSLATE()	Returns the first string with occurrences of characters in the second string replaced by the character at the corresponding position in the third string. For example, `TRANSLATE("bat","abc","ABC")` returns `"BAt"`.

About SOAP Messages

Orchestrations communicate with *Web services* [page 701] by sending and receiving *SOAP messages*. SOAP messages are XML (eXtensible Markup Language) documents that are formatted according to the rules of the SOAP specification. (See the World Wide Web Consortium Web site at http://www.w3.org for more information about the SOAP specification.)

SOAP Envelope

A SOAP message consists of a SOAP envelope. The SOAP envelope contains an optional SOAP Header and a required SOAP Body.

SOAP Header

The optional SOAP Header includes *application* [page 679]-specific information about how the SOAP message is to be processed. Each Header contains one or more *header blocks*, which can include message routing and delivery instructions, payment information, authentication credentials, or any other information that relates to processing the data in the SOAP Body.

The SOAP Header can also contain a *headerfault message* element that usually relates to Header-processing errors.

SOAP Body

The required SOAP Body contains the actual message to be processed by the ultimate *endpoint* [page 684]. The Body may contain an XML element such as `employeeNumber`, or an element that maps to the arguments or parameters in a programming method or function.

SOAP Fault

The SOAP Body can also contain an optional *SOAP fault*, which is used to carry error and status information about a SOAP message. If an error occurs during the processing of a request, a response SOAP message is returned to the sender that contains the SOAP fault in the Body of the message.

Chapter 26: Orchestration User Interface

This section describes the following orchestration-related dialog boxes, editors, and Property Editors.

Orchestration Link Editor

This editor is displayed when you select an existing orchestration link (not the **Event without Reply** item) under the **Orchestration Links** heading in App Explorer. For information about orchestration links, see About Orchestration Links [page 440].

Important: The **Event** item under the **Orchestration Links** heading is used for asynchronous orchestration workflows, and the other items are used for synchronous orchestration workflows. For information about the **Event without Reply** item, see About Events [page 437] and Event Editor [page 448].

Element	Description
Name	The name of the orchestration link. This name is displayed under the **Orchestration Links** heading in App Explorer.
Description	An optional description of the orchestration link.
Orchestration	The orchestration that contains the *orchestration workflow* [page 690] that is invoked when the *action* [page 679] corresponding to this orchestration link is executed. For convenience, you can click the name of the orchestration to view its editor. Click in the Quick Access Toolbar to return to the current view of the orchestration link editor.
Workflow	The orchestration workflow that is invoked when the action corresponding to this orchestration link is executed. For convenience, you can click the name of the orchestration workflow to view it. Click in the Quick Access Toolbar to return to the current view of the orchestration link editor.

Element	Description
Fields used by event	The fields from the *primary table* [page 692] that are passed as inputs to the linked orchestration workflow. If you need data that is not listed, define it as **Additional data used by event**.
Fields returned by event	The fields in the primary table that are set by the results of the synchronous orchestration workflow.
Additional data used by event	Additional data items that are passed as inputs to the linked orchestration workflow.
New	Adds an item to the list of additional data items.
Delete	Removes the selected data item from the list.
Move up, Move down	Moves the selected data item higher or lower in the list.

Event Editor

The event editor is displayed when you select **Event without Reply** under the **Orchestration Links** heading in App Explorer. This editor describes field values and additional data that is sent with events raised during *application workflow* [page 680] transitions. These events are raised when a transition *action* [page 679] is created that invokes an asynchronous *orchestration workflow* [page 690] (one created in the Action Wizard [page 374] with the **continue executing** option). The event corresponds to an *event definition* [page 685] in the called orchestration. For information about events and event definitions, see About Events [page 437] and About Application Links and Event Definitions [page 439].

Note: The event is shown as the **Event definition source** on the General Tab of the Event Definition Property Editor [page 455] for the event definition that is automatically created when you use the Action Wizard [page 374] to create an asynchronous orchestration workflow.

Note: The fields and additional data that you specify in this editor appear as **Custom data** elements in the Event Definition Editor [page 453].

Element	Description
Description	An optional description of the event.

Element	Description
Export to file	Lets you save the event to an `.mtd` file that can be imported by another process app that responds to this event. The file is used to create an event definition in the other process app. You can only export an event if an asynchronous orchestration action is created for some transition. For information about creating actions, see Action Wizard [page 374]. After you export an event, a new event definition appears under the **Application Links** heading in App Explorer.
Lock definition	Lets an asynchronous orchestration workflow in another process app use the event definition. For details, see About Events [page 437].
Fields to send with event	Lets you specify which fields from the *primary table* [page 692] are sent with the event. If you want to use a field that is not on the list, add the field to the primary table.
Additional data to send with event	Lets you add data elements to send with the event. These data elements are then accessible to the receiving orchestration workflow.
New	Adds an entry to the list of additional data elements.
Delete	Removes the selected data element from the list.
Move up, Move down	Moves the selected data element up or down in the list.

Event with Reply Dialog Box

This dialog box opens when you select the following options to create an orchestration link and a new synchronous *orchestration workflow* [page 690] from the Action Wizard [page 374]:

- **Orchestration Workflow** as the *action* [page 679] type
- **and wait for reply** in the rule description
- **(Add new workflow...)** from the **Select workflow service** list

The information you specify in this dialog box is reflected in the orchestration link editor. For more information, see About Orchestration Links [page 440] and Orchestration Link Editor [page 447].

Note: You can also open this dialog box by right-clicking the **Orchestration Links** heading in App Explorer and selecting **Add New Event With Reply**. In this scenario, you create the orchestration link first and then use it later in the **Action Wizard**. However, this method is not recommended.

The following table describes the configuration settings for a new orchestration link.

Element	Description
Name	The name of the new orchestration link. This name appears under the **Orchestration Links** heading in App Explorer, where you can select it to make changes.
Description	An optional description for the orchestration link.
Orchestration	The orchestration that contains the orchestration workflow. You can select **(New orchestration...)** to associate the orchestration link with an orchestration workflow in a new orchestration that is to be added to the open process app.
Workflow	The name of the new orchestration workflow.
Fields used by event	The fields from the *primary table* [page 692] whose values are passed as inputs to the orchestration workflow. If you need data that is not listed, define it as **Additional data used by event**, (described below).
Fields returned by event	The fields in the primary table whose values are returned as outputs from the orchestration workflow.  **Important:** The *Item ID* field cannot be returned by the workflow.
Additional data used by event	Additional data items that are passed as inputs to the orchestration workflow.
New	Adds an item to the list of additional data items.
Delete	Removes the selected data item from the list.
Move up, Move down	Moves the selected data item higher or lower in the list.

New Orchestration Dialog Box

Use the **New Orchestration** dialog box to add an orchestration to a process app.This dialog box opens when you select the following options to define an orchestration *action* [page 679] in the Action Wizard [page 374]:

- **Orchestration Workflow** as the action type
- **continue executing** or **and wait for reply** in the rule description
- **(New orchestration...)** from the **Select an orchestration and an orchestration workflow** list

This dialog box also opens when you do one of the following:

- Click the down arrow under the **Component** icon on the Ribbon, and select **Orchestration**.
- Right-click the name of an existing *application* [page 679] in App Explorer, point to **Add New**, and then select **Orchestration**.
- Right-click the process app name in App Explorer, point to **Add New**, and then select **Orchestration**.
- Right-click the name of an existing orchestration in App Explorer, point to **Add New**, and then select **Orchestration**.

Element	Description
Name	The name of the orchestration, which is limited to 128 characters, including spaces. Orchestrations in the SBM Application Repository must be uniquely named, even if they are used in different process apps.
Category	An optional category for the orchestration.

Event Definitions List

When you select the **Application Links** heading under the name of an orchestration in App Explorer, the list of event defintions associated with the orchestration is displayed on the **Event Definition** tab in the editor pane.

Tip: Double-click an event definition in the list to view it in the Property Editor and on the event definition tab.

The following table describes the elements in the event definition editor.

Element	Description
Name	The name of the event definition.
Product name	The name of the external product or application raising the event associated with the event definition.
Tool version	The version number of the event definition. This information is used to distinguish between event definitions used by the same product.
Product instance	The instance of the event definition. This information is used to distinguish between event definitions used by the same product.
Type	For an event defintion, the type is always Event Definition.
Updated by	The user name of the person who last reimported the event definition.
Updated on	The date and time that the event definition was last reimported.

Event Definition Configuration Dialog Box

This dialog box opens when you right-click **Application Links** under an orchestration name in App Explorer and then select **Add New Event Definition**.

Note: For more information about event definitions, see About Application Links and Event Definitions [page 439].

Creating a New Custom Event Definition

The following table describes the configuration settings for a new custom *event definition* [page 685]. These settings are present when **Create new custom event definition** is selected in the dialog box (the default selection). The *Event Manager* [page 685] uses these settings (and the event notice, object type, and event type that you define later) to identify the *orchestration workflow* [page 690] that will be called in response to an event raised from an external product or another process app.

Element	Description
Event definition name	A unique name for the new event definition. This name does not affect the mapping of an event to a workflow.
Event definition version	The event definition version. This information is used to distinguish between event definitions used by the same product.
Product name	The name of the external product or application raising the event associated with the event definition.
Product instance	The instance of the event definition. This information is used to distinguish between event definitions used by the same product.

Importing an Event Definition File to Create a New Custom Event Definition

The following table describes the configuration settings for a new event definition that uses an existing event definition (`.mtd`) file or `.wsdl` file that was exported from another event definition. These configuration settings are present when you select **Create from event definition file** in the dialog box.

Element	Description
MTD	The name of the event definition file that defines the event definition. This file was exported from another event definition, and contains the available messages, object types, and event types. Browse for a file with an `.mtd` (or `.wsdl`) extension that defines the event you want to process. Alternatively, you could enter a URL to a Web page containing information that defines the event definition. When you are satisfied with your selection, click **OK**.  **Note:** Be sure to specify a file that contains valid ALF event definitions. Otherwise, when you click **OK**, SBM Composer reminds you to select a valid file.
Documentation	Information about the event definition, as defined in the event definition file.
Operations	The operations defined in the process event definition file.

Event Definition Editor

The *event definition* [page 685] Property Editor is displayed when you select an event definition in App Explorer. The tab displays the details of an event definition that SBM automatically creates when an *application workflow* [page 680] is linked to an asynchronous *orchestration workflow* [page 690] using the **Action Wizard**, or an event definition that you create or import using the **Event Definition Configuration** dialog box.

Each *event* [page 685] is defined by an object type and an event type. When an event is fired during the execution of a transition, information is passed to the asynchronous orchestration workflow. The event definition editor shows what the information could be.

Element	Description
Object types	The object types defined in the event definition.
Event types	The event types defined in the event definition.
Custom data	The data sent with the event, shown in the **Extension** node. You can define data elements when you create a new event definition. If you import the event definition, the **Extension** node includes the field data and additional data you specified when you created the event definition (see Event Editor [page 448]).

Map Event Definition to Workflow Dialog Box

This dialog box opens when you add a new event map from the **Event Map** tab of the *event definition* [page 685] Property Editor. It enables you to associate an event definition

event with an asynchronous *orchestration workflow* [page 690]. In this dialog box, you select from the events that the event definition can raise, and select an orchestration workflow that is compatible with the selected event. You can also use this dialog box to create a new orchestration workflow.

When the event definition defines specific events, the dialog box contains only compatible events. Event compatibility is determined by the content of the **Extension** element in the **Custom data** section of the event definition editor. If two different elements have the same information in the **Extensions** element, then the same orchestration workflow can handle both events.

Important: An event definition must contain at least one object type and one event type.

The following table describes the configuration settings for a new event map.

Element	Description
Event Definition	The name of the event definition. This field is read-only.
Version	The version number of the event definition. This information is used to distinguish between event definitions used by the same product.
Product name	The name of the external product or application raising the event associated with the event definition.
Product instance	The instance of the event definition. This information is used to distinguish between event definitions used by the same product.
Object type	The object type that together with the event type, defines this mapping to the *Event Manager* [page 685]. The object type identifies the type of object or record that triggers the event type. Select the object type to associate with the orchestration workflow.
Event type	The event type that together with the object type, defines this mapping to the Event Manager. The event type identifies what occurs to trigger the event. Select the event type to associate with the orchestration workflow.
Workflow	Select an existing orchestration workflow or select **[New Workflow]** to create a new one. This should be the orchestration workflow that you want to run when an event with the selected values is received by the *Event Manager* [page 685].

Event Definition Property Editor

The *event definition* [page 685] Property Editor is displayed when you select an event definition in App Explorer. Use it to control how event definitions are mapped to asynchronous orchestration workflows in your process app. You can also use it to perform operations such as exporting the `.mtd` or `.wsdl` file for the event definition so it can be

imported by another process app or an external product, making an imported event definition editable, and refreshing the definition of the event definition.

General Tab of the Event Definition Property Editor

The **General** tab of the event definition Property Editor is displayed when you select an event definition in App Explorer. It is also displayed when you double-click an event definition in the event definition editor.

Element	Description
Name	The name of the event definition.
Product name	The name of the external product or application raising the event associated with the event definition.
Event definition source	The origin of the event definition: • Imported from the *event* [page 685] for the *application* [page 679]. (The event is automatically created when you create an asynchronous *orchestration workflow* [page 690] using the **Action Wizard**.) • An `.mtd` file or a `.wsdl` file that was exported from another event definition, or a `.wsdl` file that was used to manually create an event definition. (Select **Create from event definition file** on the **Event Definition Configuration** dialog box to create this event definition). • Defined manually by the user. (Select **Create new custom event definition** on the **Event Definition Configuration** dialog box to create this event definition.)
Description	An optional description.
Event definition version	The version number from `.mtd` or `.wsdl` file.
Product instance	The instance of the event definition. This information is used to distinguish between event definitions used by the same product.
Export event definition	Saves the `.mtd` or `.wsdl` file for the event definition to the file system of your computer, so it can be transported and used by another process app. For more information, see About Application Links and Event Definitions [page 439].
Reimport	Reimports the `.mtd` or `.wsdl` file. This button is only available for existing event definitions.

Element	Description
Regenerate	Regenerates the event definition. This button is only available for event definitions that were automatically generated from an event in an application. Use this button to regenerate the event definition if the changes you made to the event do not appear in the event definition editor.
Export external event WSDL	Saves the `.wsdl` file for the event definition to the file system of your computer, so it can be used to raise an event from any external product that can call *Web services* [page 701].  **Note:** This button is only available for new event definitions. For more information, see About Application Links and Event Definitions [page 439].
Convert to custom event definition	Converts an event definition that was created in another process app or created manually into a custom event definition that you can edit.  **Note:** This button is available for any event definition that was created in and exported from SBM Composer. It is also available for a manually-created event definition, if its structure and naming conventions are consistent with those of custom event definitions created by SBM Composer.

Event Map Tab of the Event Definition Property Editor

The **Event Map** tab of the event definition Property Editor is displayed when you select an event definition in App Explorer. It is also displayed when you double-click an event definition in the event definition editor.

Element	Description
Product instance	The instance of the event definition. This information is used to distinguish between event definitions used by the same product.
Object type	The object type that together with the event type, defines this mapping to the *Event Manager* [page 685]. The object type identifies the type of object or record that triggers the event type.
Event type	The event type that together with the object type, defines this mapping to the Event Manager. The event type identifies what occurs to trigger the event.
Workflow	The *orchestration workflow* [page 690] that is mapped to the event definition.
Add	Lets you map the event definition to an orchestration workflow.

Element	Description
Remove	Lets you remove the mapping between an orchestration workflow and the event definition.
View/ Edit	Opens the **Map Event Definition to Workflow** dialog box, which lets you view or edit the mapping between the event definition and the orchestration workflow.  **Note:** If the mapped orchestration workflow is checked in, the button is **View**, and the dialog box is read-only. If it is checked out, the button is **Edit**, and you can modify some fields in the dialog box.

Orchestration Workflow Editor

The orchestration workflow editor is displayed when you select an *orchestration workflow* [page 690] in App Explorer. This is where you create and edit an orchestration workflow. You use the orchestration workflow editor (with the and the orchestration workflow Property Editor) to visually lay out and sequence the components of your orchestration workflow.

You create an orchestration workflow by dragging and dropping steps from the **Step Palette** and arranging them in sequence. You also configure details such as which Web service is to be called or what calculations are to be performed; and map data among inputs, outputs, and working data.

Step Palette

The **Step Palette** is located to the right of the orchestration workflow editor. It contains the icons that represent the various functions that an *orchestration workflow* [page 690] can perform.

The **Step Palette** has three main sections:

- **New Items**, from which you can drag-and-drop orchestration workflow steps.
- **Configured Items**, where items are shown if they have been configured. For example, after you specify the WSDL file for a Web service, it appears in this section.
- Zoom preview section, where you can manipulate the portion of a large orchestration workflow that is visible in the orchestration workflow editor.

Orchestration Workflow Property Editor

This section describes the tabs of the orchestration workflow Property Editor. This Property Editor is displayed when you select an *orchestration workflow* [page 690] in App Explorer.

- General Tab of the Orchestration Workflow Property Editor [page 458]
- Event Map Tab of the Orchestration Workflow Property Editor [page 458]
- Data Mapping Tab of the Orchestration Workflow Property Editor [page 459]

General Tab of the Orchestration Workflow Property Editor

The **General** tab of the orchestration workflow Property Editor is displayed when you select an orchestration workflow in App Explorer.

Element	Description
Name	The name of an orchestration workflow. The name must start with a letter (A-Z, a-z), and can contain any combination of letters, digits (0-9), and underscores.
Description	An optional description of the process app.
Event definition	The name of the *event definition* [page 685] that is linked to the orchestration workflow.
WSDL message	The message from the WSDL file for the event definition. Often this is the EventNotice containing the data that accompanies the *event* [page 685].

Event Map Tab of the Orchestration Workflow Property Editor

The **Event Map** tab of the orchestration workflow Property Editor is displayed when you select an aynchronous orchestration workflow in App Explorer. The linking between an event and the orchestration workflow is called an *event map*, and is displayed on this tab.

Element	Description
Event definition	The name of the *event definition* [page 685] that is mapped to the orchestration workflow.
Object type	The object type that together with the event type, defines this mapping to the *Event Manager* [page 685]. The object type identifies the type of object or record that triggers the event type.
Event type	The event type that together with the object type, defines this mapping to the Event Manager. The event type identifies what occurs to trigger the event.
Add	Opens the **Map Workflow to Event Definition** dialog box, in which you map the orchestration workflow to an event definition. See Map Workflow to Event Definition Dialog Box [page 460] for details.
Remove	Removes the selected event map entry.

Element	Description
View/ Edit	Opens the **Event Definition Event Mapping** dialog box, which lets you view the mapping of event information to an orchestration workflow. See Event Definition Event Mapping Dialog Box [page 459] for details.  **Note:** If the orchestration workflow is checked in, the button is **View**, and the dialog box is read-only. If it is checked out, the button is **Edit**, and you can modify some fields in the dialog box.

Data Mapping Tab of the Orchestration Workflow Property Editor

The **Data Mapping** tab of the orchestration workflow Property Editor is displayed when you select an orchestration workflow in App Explorer. See About Data Mapping [page 433] for related information.

Element	Description
Working data	Displays (in a hierarchical manner) the working data for the orchestration workflow. Right-click a working data element for a menu of applicable commands. For example, the **Properties Mode** and **Mapping Mode** commands toggle between the two nodes, and the **Type [String]** command gives you the option of selecting or changing the type of the selected data element.  **Note:** Properties mode shows additional information for the selected element (such as its type and namespace).
Source elements	Shows the source of each working data element, if any. Right-click a data element and select **Suggested Mappings** to see the mappings that SBM Composer suggests. Select a cell in this column and click the down arrow to open the **Select a Source** tool, which offers options beyond the suggested mappings. Right-click a source element to open a menu of applicable commands.
Default value	Lets you view or edit the default values, if any, for data elements.  **Note:** Any value specified in the **Source elements** column will override the value in the corresponding **Default value** column.
Vertical divider	Clicking the vertical divider to the right of the Property Editor switches the Property Editor between mapping mode and properties mode.

Event Definition Event Mapping Dialog Box

The **Event Definition Event Mapping** dialog box lets you view or edit the mapping between an *event definition* [page 685] and an asynchronous *orchestration workflow*

[page 690]. The map defines the specific values for the event that causes the workflow to be invoked.

Element	Description
Event definition	Identifies the event definition.
Tool version	The version number of the event definition. This information is used to distinguish between event definitions used by the same product.
Product name	The name of the external product or application raising the event associated with the event definition.
Product instance	The instance of the event definition. This information is used to distinguish between event definitions used by the same product. If the mapped orchestration workflow is checked out, you can modify the value of this element.
Object type	The object type that together with the event type, defines this mapping to the *Event Manager* [page 685]. The object type identifies the type of object or record that triggers the event type. If the mapped orchestration workflow is checked out, you can modify the value of this element.
Event type	The event type that together with the object type, defines this mapping to the Event Manager. The event type identifies what occurs to trigger the event. If the mapped orchestration workflow is checked out, you can modify the value of this element.
Workflow	The orchestration workflow that you want to run when an event with the selected values is received by the *Event Manager* [page 685].

Map Workflow to Event Definition Dialog Box

This dialog box opens when you add a new event map from the **Event Map** tab on an asynchronous orchestration workflow Property Editor. It lets you associate an *orchestration workflow* [page 690] with an event definition. In this dialog box, you select from the events that the event definition can raise, and select an orchestration workflow that is compatible with the selected event. You can also use this dialog box to create a new orchestration workflow.

You can associate an orchestration workflow with more than one event but the events must be compatible. Event compatibility is determined by the content of the **Extension** element in the **Custom data** section of the event definition editor. If two different elements have the same information in the **Extensions** element, then the same orchestration workflow can handle both events.

After an orchestration workflow is associated with an event definition event, the dialog box shows only compatible event definition events.

Note: Event definition events can come from SBM applications or from external products.

The following table describes the configuration settings for a new event map.

Element	Description
Event definition	The name of the event definition. This field is read-only.
Tool version	The version number of the event definition. This information is used to distinguish between event definitions used by the same product.
Product name	The name of the external product or application raising the event associated with the event definition.
Product instance	The instance of the event definition. This information is used to distinguish between event definitions used by the same product.
Object type	The object type that together with the event type, defines this mapping to the *Event Manager* [page 685]. The object type identifies the type of object or record that triggers the event type. Select the object type to associate with the orchestration workflow.
Event type	The event type that together with the object type, defines this mapping to the Event Manager. The event type identifies what occurs to trigger the event. Select the event type to associate with the orchestration workflow.
Workflow	Select an existing orchestration workflow or select **[New Workflow]** to create a new one. This should be the orchestration workflow that you want to run when an event with the selected values is received by the *Event Manager* [page 685].

Select Library Type Dialog Box

You use the **Select Library Type** dialog box when you create a new data element and need to assign a type to it, or when you need to change the type of an existing data element.

This dialog box makes it easy to select a type when there are many types in the Type Library. The types that are listed in the dialog box are all named types defined by the *Web services* [page 701] that were imported into the orchestration, and that are listed in the Type Library Editor [page 462].

This dialog box opens when you perform the following steps:

1. Click the **Data Mapping** tab on the orchestration workflow Property Editor.

2. Right-click in the **Working Data** area, and select **Properties Mode**, if not already selected.
3. If you want to add a new data element and select the library type, perform the following steps:
   a. Right-click the **Working Data** node, and select **Add New DataElement**.
   b. Right-click the new data element, select **Type [String]**, and then select **Select from Type Library**.
4. If you want to change the library type of an existing data element with the "String" type, right-click the data element, select **Type [String]**, and then select **Select from Type Library**.

Element	Description
Look for	Type a partial or complete data element name you want to find.
Find	Starts the search.
Clear	Clears the information in the **Look for** box.
Options	Provides other search options.
Show type details	Opens the **Type details** section, which shows detailed information about the selected element.

Type Library Editor

When you select the **Type Library** heading under the name of an orchestration in App Explorer, a list of named types associated with the process app is displayed in the editor pane. This read-only list contains all named types defined by the *Web services* [page 701] that were imported into the orchestration.

The named types are grouped by namespace at the top part of the editor. When you select a named type, its data elements are displayed at the bottom part of the editor.

You can search for a named type by typing the full name or a few letters of the name in the **Look for** box, and then clicking **Find**. Search options are available in the **Options** menu.

Chapter 27: Orchestration Procedures

This section includes the following orchestration procedures:

Using Data Mapping

You use the **Data Mapping** tab of the orchestration workflow Property Editor to create working data and define its default values, and to specify the source of data needed by a Web service. This section includes data mapping procedures.

Creating a Practice Process App for Data Mapping

In this section, you will create a new process app (DataMappingProcessApp) that contains an *application workflow* [page 680]. Then you will create an orchestration workflow (DataMappingOrchWF), which you will use to practice mapping data.

Important: This process app is for demonstration purposes only and is not valid. Do not try to *publish* [page 692] or deploy it.

To create the practice process app:

1. Start SBM Composer.
2. Click the Composer button, and then select **New**.
3. In the **Create New Process App** dialog box that opens, in the **Available Templates** pane, click **Application Process App**, and then click **Create**.
4. In the **Configure Process App** dialog box that opens, in the **Process app name** box, type `DataMappingProcessApp`.
5. In the **Category** box, type `Examples`.
6. In the **Application name** box, type `DataMappingApp`, and then click **OK**.
7. In App Explorer, under the **Application Workflows** heading, right-click **DataMappingApp** and then select **Rename**.
8. Change the name to `DataMappingAppWF` and then press the Tab key.
9. In App Explorer, right-click **DataMappingProcessApp**, point to **Add New**, and then select **Orchestration**.
10. In the **New Orchestration** dialog box, type `DataMappingOrch` in the **Name** box and then click **OK**.
11. In App Explorer, under **Application Workflows**, select **DataMappingAppWF**.
12. In the application workflow editor, select the **New** state. On the **General** tab of the application workflow Property Editor, change the value of the **Name** field to `State`, and then press the Tab key.
13. In the application workflow editor, select the **Submit** transition.
14. On the **General** tab of the transition Property Editor, change the value of the **Name** field to `Transition`, and then press the Tab key.
15. Right-click the **Transition** transition, and select **Show Actions**.
16. On the **Actions** tab of the transition Property Editor, click **New.**

    The **Action Wizard** asks, "Which type of action do you want to execute?"
17. Without changing anything, click **Next.**

    The **Action Wizard** asks, "What do you want to affect?"
18. Without changing anything, click **Next**.

    The **Action Wizard** asks, "Which condition do you want to check?"

19. Without changing anything, click **Next**.

    The **Action Wizard** asks, "Which orchestration workflow do you want to invoke?"

20. In the list under **Step 1**, select **(Add new workflow...)**.

    SubmitTransitionWorkflow is added to list and selected.

21. Without changing anything in **Step 2**, click **Finish**.

22. In App Explorer, under **DataMappingOrch**, under **Orchestration Workflows**, select **SubmitTransitionWorkflow**.

23. On the **General** tab of the orchestration workflow Property Editor, change the value of the **Name** field to `DataMappingOrchWF`, and then press the Tab key.

24. Save the process app by performing the following steps:

    a. On the Quick Access Toolbar, click the **Save locally** icon. A message reminds you that the design elements were saved to the *Local Cache* [page 688] only.

    b. Click **OK**.

25. If you closed SBM Composer after saving the process app, perform the following steps:

    a. Start SBM Composer.

    b. Click the Composer button, and then select **Open**.

    c. In the **Open Process App** dialog box, select **DataMappingProcessApp**, and then click **Open.**

    d. In App Explorer, click the **All Items** filter, if it is not already selected.

Creating Private Simple or Library Type Working Data

On the **Data Mapping** tab of the orchestration workflow Property Editor, you can create *private simple* or *library type* working data elements to use in an orchestration. The steps for creating them are the same.

Private simple working data is classified by data types. In computer programming, a data type restricts a data element to a particular type of information. For example, a *string* can contain only characters and spaces. "ProcessApp101" and "ab cde!f" are both strings. An *integer* can contain only whole numbers, that is, numbers that do not contain fractional parts. The numbers 1, 88, and 1099 are integers; however, 1.5, 88.72, and 1099.579 are not (they are called *floating-point numbers*). Private simple working data is shared by all of the steps in the *orchestration workflow* [page 690].

Library type working data is provided by the WSDL files that are imported into an orchestration workflow. For example, the "auth" library type is a private complex type that contains the userId, password, hostname, and loginAsUserId parameters supplied by the SBM Web service. For more information about the "auth" data element (argument), refer to the *Serena® Business Manager Web Services Developer's Guide*.

Procedure Summary

1. Select an orchestration workflow to display it in the orchestration workflow editor.

2. Click the **Data Mapping** tab of the orchestration workflow Property Editor.

3. To add a new data element, right-click the **WorkingData** step input, and then select **Add New DataElement**.
4. Change the name of the new data element, if you want.
5. Right-click the new data element, point to **Type[String]**, click **Select from Type Library**, and then select a data type. (The icon to the left of the name changes accordingly.)

In the following exercise, you will create a private simple working data element (PrivateSimple) and a library type data element (LibraryType).

To create a private simple working data element and a library type working data element:

1. In App Explorer, select **DataMappingOrchWF** under the **Orchestration Workflows** heading.
2. In the orchestration workflow Property Editor, select the **Data Mapping** tab.

   The **WorkingData** step input should be visible under the name of the orchestration workflow (**DataMappingOrchWF**) in the **Working data** column. (If it is not, click a blank area in the orchestration workflow editor.)
3. To create a private simple working data element:
   a. Right-click the **WorkingData** step input, and then select **Add New DataElement**.

      A new data element (**DataElement**) is added under the **WorkingData** step input.
   b. While **DataElement** is still selected, type `PrivateSimple`, and then press the Tab key.
   c. Right-click **PrivateSimple**, point to **Type[String]**, and then select a data type such as **Boolean** or **Integer**. (**Type[String]** indicates that the data type of **PrivateSimple** was String.)

      If you select **Boolean**, the icon to the right of **PrivateSimple** changes from to  . If you select **Integer**, the icon changes from  to  .
4. To create a library type working data element:
   a. Right-click the **WorkingData** step input, and then select **Add New DataElement**.

      A new data element (**DataElement**) is added under **PrivateSimple**.
   b. While **DataElement** is still selected, type `LibraryType`, and then press the Tab key.
   c. Right-click **LibraryType**, point to **Type[String]**, click **Select from Library Type**, and then select a data type such as **Auth** or **CredentialsType**. If you selected **Auth**, four child elements are added under the **Auth** data element.

      The icon to the left of the **LibraryType** data element changes from  to  .

Creating Private Complex Working Data

On the **Data Mapping** tab of the orchestration workflow Property Editor, you can create private complex data to use in orchestrations.

Procedure Summary

1. Select an *orchestration workflow* [page 690] to display it in the orchestration workflow editor.
2. Click the **Data Mapping** tab of the orchestration workflow Property Editor.
3. To add a new data element, right-click the **WorkingData** step input, and then select **Add New DataElement**.
4. Rename the new data element, if you want.
5. Right-click the new data element, point to **Type[String]**, and then select **Private Complex**.
6. To add a child data element, right-click the new data element again, and then select **Add Child DataElement**.
7. Rename the new child data element, if you want.
8. Repeat the previous two steps to add more child data elements.

In the following exercise, you will create a private complex working data element named "PrivateComplex" that contains three child data elements (Child1, Child2, and Child3).

To create a private complex working data element:

1. Under **Orchestration Workflows**, select **DataMappingOrchWF.**
2. In the orchestration workflow Property Editor, click the **Data Mapping** tab.

   The **WorkingData** step input should be visible under the name of the orchestration workflow (**DataMapping**) in the **Working data** column. (If it is not, click a blank area in the orchestration workflow editor.)
3. Right-click the **WorkingData** step input, and then select **Add New DataElement**.

   DataElement appears under the **WorkingData** step input.
4. While **DataElement** is still selected, type `PrivateComplex`, and then press the Tab key.
5. Right-click **PrivateComplex**, point to **Type[String]**, and then select **Private Complex**. (**Type[String]** indicates that the original data type of **PrivateComplex** was string.)

   The  icon changes to  .
6. Right-click **PrivateComplex**, and then select **Add Child DataElement**.

   A new data element (**DataElement**) is added under **PrivateComplex.**
7. While **DataElement** is still selected, type `Child1`, and then press the Tab key. You can also change the data type of the child data element.

8. Repeat steps 6 [page 467] and 7 [page 467] two more times, naming the new child data elements `Child2` and `Child3`.

Creating Arrays of Working Data

On the **Data Mapping** tab of the orchestration workflow Property Editor, you can create an array of working data to use in orchestrations. To do this, you first create an array container data element, then the array, and then the array elements.

Procedure Summary

1. Select an *orchestration workflow* [page 690] to display it in the orchestration workflow editor.
2. Click the **Data Mapping** tab of the orchestration workflow Property Editor.
3. Create the array container data element by right-clicking the **Working Data** step input and then selecting **Add New DataElement**.
4. Rename the new array container data element, if you want.
5. Right-click the array container data element, point to **Type[String]**, and then select **Private Complex**.
6. Create the array by right-clicking the array container data element again and then selecting **Add Child DataElement**.
7. Right-click the array, and then select **Properties Mode**, if it is not already selected.
8. In the properties area under **Misc**, select the **IsUnbounded** row.
9. Click the down arrow next to **False**, and then select **True**.
10. Close the properties area by right-clicking the array and then selecting **Mapping Mode**.
11. Rename the array and change its data type, if you want.
12. Create an array data element by right-clicking the array and then selecting **Add Array Element**. (The new array data element inherits the name and data type of the array. You cannot change this.)
13. Repeat the previous step to add more array data elements.

In the following exercise, you will create an array container data element (ArrayContainer) that contains one array (Array) with three array elements (Array[1], Array[2], and Array[3]).

To create an array of working data elements:

1. In App Explorer, select **DataMappingOrchWF** under the **Orchestration Workflows** heading.
2. In the orchestration workflow Property Editor, click the **Data Mapping** tab.

   The **WorkingData** step input should be visible under the name of the orchestration workflow (**DataMappingOrchWF**) in the **Working data** column. (If it is not, click a blank area in the orchestration workflow editor.)

3. Right-click the **WorkingData** step input, and then select **Add New DataElement**.

   A new data element (**DataElement**), which is the array container, is added under the **WorkingData** step input.

4. While **DataElement** is still selected, type `ArrayContainer`, and then press the Tab key.

5. Right-click **ArrayContainer**, point to **Type[String]**, and then select **Private Complex**. (**Type[String]** indicates that the original data type of **ArrayContainer** was **String**.)

   The  icon changes to  .

6. Right-click **ArrayContainer**, and then select **Add Child DataElement**.

   A new **DataElement** is added under **ArrayContainer**.

7. While **DataElement** is still selected, type `Array`, and then press the Tab key.

8. Right-click **Array**, and then select **Properties Mode**.

   The properties area opens on the right side of the orchestration workflow Property Editor.

9. In the properties area under **Misc**, select the **IsUnbounded** row.

10. Click the down arrow next to **False**, and then select **True**.

    A left and right square bracket are added to the array name (**Array[]**) to indicate that the data element is an array.

11. To close the properties area, right-click **Array[]**, and then select **Mapping Mode**.

    The number of array elements is displayed in parentheses to the right of **Array[]**. The number begins at 0 (zero) and increases by one each time you add an array element.

12. Right-click **Array[]**, and then select **Add Array Element**.

    A new array element (**Array[1]**) is added under **Array[]**.

13. Repeat step 12 [page 469] two more times.

    Array elements **Array[2]** and **Array[3]** are added to **Array[]** under **Array[1]**, and **(3 elements)** is added to the right of **Array[]**.

Setting Default Values

On the **Data Mapping** tab of the orchestration workflow Property Editor, you can set the default value for a data element.

Procedure Summary

1. Select an *orchestration workflow* [page 690] to display it in the orchestration workflow editor.

2. Click the **Data Mapping** tab of the orchestration workflow Property Editor.

3. Click in the **Default value** column of a data element. Depending on the data type, enter a value, click the down arrow and select an option on a menu, click the down arrow and select a date from a calendar, and so on.

In this procedure, you will set the default value for PrivateSimple, which is a String data type, to `Text`.

To set the default value for a data element:

1. In App Explorer, select **DataMappingOrchWF** under the **Orchestration Workflows** heading.
2. In the orchestration workflow Property Editor, click the **Data Mapping** tab.

   The **WorkingData** step input should be visible under the name of the orchestration workflow (**DataMappingOrchWF**) in the **Working data** column. (If it is not, click a blank area in the orchestration workflow editor.)
3. If the data type for **PrivateSimple** is something other than **String**, right-click the **PrivateSimple** data element, point to **Type[*type*]**, point to **Private Simple**, and then select the **String** data type.
4. Locate the **PrivateSimple** data element, click in the cell corresponding to the **Default value** column, enter `Text`, and then press the Tab key.

Setting Source Values Using Suggested Mappings

On the **Data Mapping** tab of the orchestration workflow Property Editor, you can set the source value for a data element in its **Source elements** column to use in an *orchestration workflow* [page 690]. You can perform this procedure by selecting suggested mappings or by selecting the values manually. Suggested mappings are for compatible data types only. You can also map compatible and incompatible data types using the **Select a source** tool, as explained in Setting Source Element Mappings Manually [page 471].

Procedure Summary

1. Select an orchestration workflow to display it in the orchestration workflow editor.
2. Click the **Data Mapping** tab.
3. Right-click the **Source elements** column of the data element for which you want to set a source value, point to **Suggested Mappings**, and then select a source value. (You might have to expand one or more data elements to locate the data element you want to map.)

In the following exercise, you will map the source data for the **LibraryType** data element to a suggested data source.

To use suggested mappings for setting the source value for a data element:

1. In App Explorer, select **DataMappingOrchWF** under the **Orchestration Workflows** heading.
2. In the orchestration workflow Property Editor, click the **Data Mapping** tab.

   The **Working Data** step input should be visible under the name of the orchestration workflow (**DataMappingOrchWF**) in the **Working data** column. (If it is not, click a blank area in the orchestration workflow editor.)

3. Right-click the **Working Data** step input, and then select **Add New DataElement**.
4. While the new **DataElement** is still selected, type `SuggestedMapping`, and then press the Tab key.
5. Locate the **SuggestedMapping** data element, right-click the corresponding cell in the **Source elements** column, point to **Suggested Mappings**, point to **Compatible Items**, and then select **Array[1](Working Data - \ArrayContainer)**.

   DataMappingOrchWorkflow\ArrayContainer\Array[1] is added to the **Source elements** column.

   Note: The **Suggested Mappings** menu is limited to 40 items: a maximum of 20 exact matches, 10 similar matches, and 10 other matches.

Setting Source Element Mappings Manually

On the **Data Mapping** tab of the orchestration workflow Property Editor, you can set the source value for a data element in its **Source elements** column to use in an *orchestration workflow* [page 690]. You can perform this procedure by selecting suggested mappings or by selecting the values manually. You select compatible and incompatible data types manually using the **Select a Source** tool. Selecting suggested mappings is explained in Setting Source Values Using Suggested Mappings [page 470].

Procedure Summary

1. Select an orchestration workflow to display it in the orchestration workflow editor.
2. Click the **Data Mapping** tab of the orchestration workflow Property Editor.
3. Find a data element for which you want to set a source value, select its **Source Elements** column, and then click the down arrow.
4. In the **Select a source** tool, select the source value or values that you want to map, and then click **OK**.

In the following exercise, you will manually map the data for **Child1** under the **PrivateComplex** data element to a source value of a compatible data type.

To manually set the source value for a data element:

1. In App Explorer, select **DataMappingOrchWF** under the **Orchestration Workflows** heading.
2. In the orchestration workflow Property Editor, click the **Data Mapping** tab.

   The **Working Data** step input should be visible under the name of the orchestration workflow (**DataMappingOrchWF**) in the **Working data** column. (If it is not, click a blank area in the orchestration workflow editor.)
3. Right-click the **Working Data** step input, and then select **Add New DataElement**.
4. While the new **DataElement** is still selected, type `ManualMapping`, and then press the Tab key.
5. Locate the **ManualMapping** data element, select the corresponding cell in the **Source elements** column, and then click the down arrow.

6. In the **Select a source** tool, under **Working Data**, expand **ArrayContainer** and **Array[] (3 records)**, select **Array[2]**, and then click **OK.**

   DataMappingOrchWorkflow\ArrayContainer\Array[2] is added to the **Source elements** column.

   This data source is of a compatible data type, as indicated by the message "Compatible type selected" that briefly appears in the box to the left of the **Clear** button in the **Select a Source** tool. The **Select a Source** tool also lets you map incompatible data types or to select multiple data types to map to a single data element, or both. To do this, click the **Show advanced options** link, and then select the appropriate check box or check boxes.

Viewing and Editing Data Element Properties

On the **Data Mapping** tab of the orchestration workflow Property Editor, you can view and edit property information for a data element. This is useful if you need to view the data types of data elements, or if you need to view or edit other properties, such as the namespace, for a particular data element.

Procedure Summary

1. Select an *orchestration workflow* [page 690] to display it in the orchestration workflow editor.
2. Click the **Data Mapping** tab of the orchestration workflow Property Editor.
3. Right-click the **Working data** or **Source elements** column for working data, or the **Step inputs** or **Source elements** column for source data, and then select **Properties Mode**.
4. View or edit the properties for the selected working data, or view the data elements for a **Service** step in the properties area that opens on the right side of the orchestration workflow Property Editor.
5. To exit properties mode, right-click the **Working data**, **Step inputs**, or **Type** column, and then select **Mapping Mode**.

   Tip: You can also switch between mapping mode and properties mode by clicking the vertical divider on the right side of the orchestration workflow Property Editor.

In the following exercise, you will view the properties of the **PrivateSimple** working data element.

To view property information for a data element:

1. In App Explorer, select **DataMappingOrchWF** under the **Orchestration Workflows** heading.
2. In the orchestration workflow Property Editor, click the **Data Mapping** tab.

   The **Working Data** step input should be visible under the name of the orchestration workflow (**DataMappingOrchWF**) in the **Working data** column. (If it is not, click a blank area in the orchestration workflow editor.)
3. Locate the **SuggestedMapping** data element, right-click the corresponding cell in the **Source elements** column, and then select **Properties Mode**.

You can view the properties of working data elements and change any editable values in the property area that opens on the right side of the orchestration workflow Property Editor.

Note: You can only view the properties of a **Service** step.

4. To exit properties mode, right-click the **SuggestedMapping** data element, and then select **Mapping Mode**.

Tip: You can also switch between mapping mode and properties mode by clicking the vertical divider on the right side of the orchestration workflow Property Editor.

Showing the Required Flag

In the hierarchy of step inputs and data elements on the **Data Mapping** tab of the orchestration workflow Property Editor, each element has an associated icon. You can identify required step inputs and data elements by a red symbol that appears in the top left corner of the icon. This symbol is called the required flag.

Flagged input and output data elements for a Web service require specific data. Sometimes a step input or a data element of the complex type displays the required flag, but its child data elements are not flagged. To determine which data elements and the type of data that are required for a Web service, refer to its WSDL file or to the documentation for the Web service.

Flagged working data must be assigned a value before it can be used in an *orchestration workflow* [page 690].

The following table gives examples of icons that identify required data elements.

Required Data Type	Icon
String	
Integer	
Date	

Procedure Summary

1. Select an orchestration workflow to display it in the orchestration workflow editor.
2. Click the **Data Mapping** tab in the orchestration workflow Property Editor.
3. To show the required flag for all required step inputs and data elements, right-click anywhere in the data mapping area, and then select **Show Required Flag**.
4. To hide all required flags, right-click anywhere in the data mapping area, and then clear **Show Required Flag**.

To show the required flag for all required step inputs and data elements:

1. In App Explorer, select **DataMappingOrchWF** under the **Orchestration Workflows** heading.
2. In the orchestration workflow Property Editor, click the **Data Mapping** tab.

   The **Working Data** step input should be visible under the name of the orchestration workflow (**DataMappingOrchWF**) in the **Working data** column. (If it is not, click a blank area in the orchestration workflow editor.)
3. Right-click anywhere in the data mapping area, and then select **Show Required Flag**.

   The required flag appears on all required step inputs and data elements.
4. To hide all required flags, right-click anywhere in the data mapping area, and then clear **Show Required Flag**.

Clearing Data Mapping

On the **Data Mapping** tab of the orchestration workflow Property Editor, you can clear the mapping for a data element. (You cannot use this procedure to clear the default value.)

Procedure Summary

1. Select an *orchestration workflow* [page 690] to display it in the orchestration workflow editor.
2. Click the **Data Mapping** tab of the orchestration workflow Property Editor.
3. Find a data element for which you want to clear the data mapping, right-click the corresponding cell in the **Source elements** column, and then select **Clear Mapping**.
4. You can also clear the mapping by selecting the appropriate cell in the **Source elements** column, clicking the down arrow, selecting **[no mapping]** in the **Select a source** tool, and then clicking **OK**.

In this exercise, you will clear the mapping for the **SuggestedMapping** data element.

To clear the data mapping for a data element:

1. In App Explorer, select **DataMappingOrchWF** under the **Orchestration Workflows** heading.
2. In the orchestration workflow Property Editor, click the **Data Mapping** tab.

   The **Working Data** step input should be visible under the name of the orchestration workflow (**DataMappingOrchWF**) in the **Working data** column. (If it is not, click a blank area of the orchestration workflow editor.)
3. Locate the **SuggestedMapping** data element, right-click the corresponding cell in the **Source elements** column, and then select **Clear Mapping**.

Tip: You can also clear the mapping by selecting the appropriate cell in the **Source elements** column, right-clicking the down arrow, selecting **[no mapping]** in the **Select a source** tool, and then clicking **OK**.

Creating a New Custom Event Definition

This topic describes how to create a new custom *event definition* [page 685] in SBM Composer.

Note: For information about event definitions, see About Application Links and Event Definitions [page 439].

To create a new custom event definition:

1. Right-click **Application Links** under the orchestration name in App Explorer, and then select **Add New Event Definition**. The **Event Definition Configuration** dialog box opens.
2. Click **Create new custom event definition**, if this option is not already selected.

3. Complete the dialog box as described in Event Definition Configuration Dialog Box [page 452].

Importing an Event Definition File for a New Custom Event Definition

This topic describes how to import an existing event definition (`.mtd`) file or `.wsdl` file to use in a new custom event definition.

Note: For information about event definitions, see About Application Links and Event Definitions [page 439].

To import a file for a new custom event definition:

1. Right-click **Application Links** under the orchestration name in App Explorer, and then select **Add New Event Definition**. The **Event Definition Configuration** dialog box opens.
2. Click **Create from event definition file**.
3. Complete the dialog box as described in Event Definition Configuration Dialog Box [page 452].

Mapping an Orchestration Workflow to an Event Definition

To connect events with asynchronous orchestration workflows, you set up a mapping on the **Event Map** tab of the event definition Property Editor or orchestration workflow Property Editor. You can either generate an *orchestration workflow* [page 690] from an event, or you can create an orchestration workflow and choose what event it is going to handle. In either case, the event `Extension` data becomes visible as part of the input data for the orchestration workflow and is available for processing by it.

You can add additional events to an orchestration workflow, but only if they are compatible with the existing event definition. To be compatible, the events must have the same `Extension` data. Generally, this means that the events are defined by the same event definition.

To map a new orchestration workflow to an event definition:

1. In App Explorer, under the **Application Links** heading, click the name of the event definition.
2. On the **Event Map** tab of the event definition Property Editor, click **Add**.
3. In the **Map Event Definition to Workflow** dialog box that opens, perform the following steps:
   a. Select the event, defined by the **Object type** and **Event type** values, that you want the new orchestration workflow to handle.
   b. Select **[New Workflow]** from the **Workflow** list.
   c. Click **OK**.

An orchestration workflow named **Workflow** appears under the **Orchestration Workflows** heading, and the event definition is automatically mapped to it.

Note: To see the mapping, view the information on the **Event Map** and **Data Mapping** tabs of the orchestration workflow Property Editor.

To map an existing orchestration workflow to an event definition (first method):

1. In App Explorer, under the **Orchestration Workflows** heading, click the name of a new orchestration workflow.
2. On the **Event Map** tab of the orchestration workflow Property Editor, click **Add**.

   Note: This tab is not available for synchronous orchestration workflows.

3. In the **Map Workflow to Event Definition** dialog box that opens, perform the following steps:
   a. Select an event definition from the **Event definition** list. This list is read-only if there is only one event definition in the orchestration.
   b. If the product supports more than one set of custom data, select the message to determine what will be received as inputs to the orchestration workflow.
   c. Select the event, defined by the **Object type** and **Event type** values, that you want the new orchestration workflow to handle.
   d. Click **OK**.

To map an existing orchestration workflow to an event definition (second method):

1. In App Explorer, under the name of an orchestration, under the **Application Links** heading, click the name of the event definition.
2. On the **Event Map** tab of the event definition Property Editor, click **Add**.
3. In the **Map Event Definition to Workflow** dialog box that opens, perform the following steps:
   a. Select the event, defined by the **Object type** and **Event type** values, that you want the new orchestration workflow to handle.
   b. Select the existing orchestration workflow from the **Workflow** list.
   c. Click **OK**.

Using the Step Palette

Steps are the building blocks of an *orchestration workflow* [page 690]. You select steps from the **Step Palette** to the right of the orchestration workflow editor, and then use the drag-and-drop operation to move them onto the line between the **Start** and **End** steps in the orchestration workflow. This creates the control flow structure that determines what the orchestration workflow does.

The following topics describe the steps available on the **Step Palette**:

Creating a Practice Process App for Using the Step Palette

In this section, you create a new process app with one *application workflow* [page 680]. Then you create the following orchestration workflows in the corresponding sections for each step: CalculateOrchWF, DecisionOrchWF, ForEachOrchWF, WhileOrchWF, and ServiceOrchWF. You use these orchestration workflows to practice using the following steps in the **Step Palette**: **Calculate**, **Decision**, **ForEach**, **While**, and **Service**.

To create the practice process app:

1. Start SBM Composer.
2. Click the Composer button, and then click **New**.
3. In the **Create New Process App** dialog box that opens, in the **Available Templates** pane, click **Application Process App**, and then click **Create**.
4. In the **Configure Process App** dialog box that opens, in the **Process app name** and **Application name** boxes, type `StepPaletteProcessApp`, and then click **OK.**

   The new process app appears in App Explorer.
5. In App Explorer, click the **All Items** filter.
6. Click **StepPaletteProcessApp**.
7. On the **StepPaletteProcessApp** tab, change the **Logical Name** to `StepPaletteApp`.
8. Under **Application Workflows**, click **StepPaletteProcessApp**.
9. On the **General** tab of the **Property Editor**, change the **Name** to `StepPaletteAppWF`, and then press the Tab key.
10. In the application workflow editor, double-click the name of the **New** state, type `State1`, and then press the Tab key.
11. Double-click the name of the **Submit** transition, type `Calculate`, and then press the Tab key.
12. In the **Common Items** section of the **Workflow Palette**, drag an **State** onto the application workflow editor and drop it to the right of the **State1** state.
13. Change the **Name** to `State2`, and then press the Tab key.

14. In the **Common Items** section of the **Workflow Palette**, drag a **Transition** onto the **State1** state, release the mouse button, and then click **State2**.
15. Change the **Name** of the **Transition** to `Decision`, and then press the Tab key.
16. Add three more states and three more transitions. Drop each new state somewhere after the previous state, and name the new states `State3`, `State4`, and `State5`. Name the transitions as follows:
    - Between **State2** and **State3**: `ForEach`
    - Between **State3** and **State4**: `While`
    - Between **State4** and **State5**: `Service`

    Tip: You can drop a state anywhere on the application workflow editor as long as you connect it, using a transition, to the previous state in the workflow.
17. In App Explorer, click **StepPaletteProcessApp**.
18. On the **Home** tab of the Ribbon, in the **New** group, click **Component**, and then select **Orchestration**.
19. In the **New Orchestration** dialog box, type `StepPaletteOrch` in the **Name** box, and then click **OK**.
20. Save the process app:
    a. On the Quick Access Toolbar, click the **Save locally** icon.

       A message reminds you that the design elements have been saved to the *Local Cache* [page 688] only.

    b. Click **OK**.
21. If you closed SBM Composer after saving the process app:
    a. Start SBM Composer.
    b. Click the Composer button, and then click **Open**.
    c. In the **Open Process App** dialog box, select **StepPaletteProcessApp**, and then click **Open.**
    d. In App Explorer, select the **All Items** filter.

Using the Calculate Step

You use the **Calculate** step to perform calculations on the data elements in an orchestration workflow. When you select a **Calculate** step, the **General** and **Options** tabs appear in the step Property Editor. The **General** tab contains the type, name, and description of the **Calculate** step. The **Options** tab contains the following two sections:

- The **Target** section defines the entity that receives the data.
- The **Expression** section defines the source of the data.

Procedure Summary

1. Select an *orchestration workflow* [page 690] to display it in the orchestration workflow editor.
2. In the **New Items** section of the **Step Palette**, drag and drop a **Calculate** step onto the line between the **Start** and **End** steps.
3. On the **General** tab of the Property Editor, you can change the name of the **Calculate** step and enter a description.
4. On the **Options** tab, in the **Target** section, enter a working data element or an expression that represents the data element that receives the value from the **Expression** section. You can use any of the functions, logical operators, or arithmetic operators available on the **Functions**, **Logical**, and **Operator** menus, respectively.
5. In the **Expression** section, enter an expression that represents the source of the data to be received by the target. You can use any of the functions, logical operators, or arithmetic operators available on the **Functions**, **Logical**, and **Operator** menus, respectively.

   For more information about creating expressions, see About the Expression Editor [page 442].

Creating an Empty Orchestration Workflow For the Calculate Step

In this exercise, you create an empty synchronous workflow. In the next section, you add and configure a **Calculate** step.

To create an empty orchestration workflow for the Calculate step:

1. In App Explorer, under **StepPaletteApp**, under **Application Workflows**, select **StepPaletteAppWF.**
2. In the application workflow editor, right-click the **Calculate** transition, and select **Show Actions** on the menu.
3. On the **Actions** tab of the Property Editor, click **New.**

   The **Action Wizard** asks, "Which type of action do you want to execute?"
4. In the **Step 2** box, click the **and continue executing** link, select **wait for reply**, and then click **Next**.

   The **Action Wizard** asks, "When do you want to call the orchestration workflow?"
5. In the **Step 1** box, select **After**, do not change anything in the **Step 2** box, and then click **Next**.

   The **Action Wizard** asks, "Which orchestration workflow do you want to call?"
6. On the menu under **Step 1**, select **(Add new workflow...)**.
7. In the **Event With Reply** dialog box, do the following:

   a. Change the **Name** to `CalculateOrchWFWR`.

   b. On the **Orchestration** menu, select **StepPaletteOrch.**

c. In the **Workflow** box, change the name to `CalculateOrchWF`.

d. In the **Fields used by event** column, select the **Title** check box.

e. In the **Fields returned by event** column, select the **Title** check box.

f. Click **OK.**

8. In the **Action Wizard**, click **Finish**.

Practicing With the Calculate Step

In this exercise, you add a **Calculate** step to the CalculateOrchWF *orchestration workflow* [page 690] and define values for it. When CalculateOrchWF is invoked, it places the text `are here.` in the **Title** box of the **State1** state form after you click the **OK** button on the *transition form* [page 700] between the **Submit** state and **State1**.

To use the Calculate step in an orchestration workflow:

1. In App Explorer, under **StepPaletteOrch**, under **Orchestration Workflows**, select **CalculateOrchWF**.
2. In the **New Items** section of the **Step Palette**, drag and drop a **Calculate** step onto the line between the **Start** and **End** steps.
3. On the **Options** tab of the Property Editor, in the **Target** section, enter the following expression: `EventNoticeWithReply.Extension.Title`
4. In the **Expression** section, enter the following function:
   `CONCAT(EventNoticeWithReply.Extension.Title, " are here.")`

   For more information about creating expressions, see About the Expression Editor [page 442].
5. Select the **End** step.
6. On the **Data Mapping** tab, under **Extension**, locate the **Title** data element, select the cell corresponding to the **Source elements** column, and then click the down arrow.
7. In the **Select a source** tool, expand **Inputs**, **EventNoticeWithReply**, and **Extension**; select **Title**; and then click **OK**.

Using the Decision Step

You use the **Decision** step to decide between two or more possible outcomes. This step lets you branch the flow of control based on *rules*. Rules are expressions, as described in About the Expression Editor [page 442].

Using the **Insert New Branch** command, you can add branches to the **Decision** step. When you select a branch, the **General** and **Options** tabs appear in the Property Editor. The **General** tab provides a name and an optional description for the branch. The **Options** tab defines the rule associated with the branch. (The default branch, named **Otherwise**, has no rule. It is invoked if all the other branches are false.)

Procedure Summary

1. Select an *orchestration workflow* [page 690] to display it in the orchestration workflow editor.

2. In the **New Items** section of the **Step Palette**, drag a **Decision** step onto the line between the **Start** and **End** steps.
3. On the **General** tab of the Property Editor, you can change the name of the **Decision** step and you can also enter a description.
4. To add a branch, right-click the **Decision** step, and then select **Insert New Branch** on the menu.

   A **Branch** is added above the **Otherwise** branch.
5. On the **General** tab of the Property Editor, you can change the name of the **Branch** and you can also enter a description.
6. On the **Options** tab of the Property Editor, in the **Rule** section, enter an expression that represents the rule for that branch. You can use any of the functions, logical operators, or arithmetic operators available on the **Functions**, **Logical**, and **Operator** menus, respectively.

   (For more information about expressions, see About the Expression Editor [page 442].)
7. Select the new (non-Otherwise) **Branch**.
8. Add a step or steps from the **Step Palette** to the **Branch** to define the actions that should be taken or the calculations that should be performed while the rule defined for the branch is true.
9. Repeat the previous step to add and configure other decision branches.

   Tip: You can switch from branch to branch by selecting a branch on the menu of the Property Editor. Also, when you right-click the name of a branch in the orchestration workflow editor, you can select menu items that let you duplicate and rename the branch, and to reorder branches.
10. Add a step or steps from the **Step Palette** to the **Otherwise** branch. You do not define a rule for this branch.

    Tip: To hide the branches as you work on another part of the orchestration workflow, click the minus sign to the left of the **Decision** step.

Creating an Empty Orchestration Workflow For the Decision Step

In this exercise, you create an empty synchronous orchestration workflow. In the next section, you add and configure a **Decision** step.

To create an empty orchestration workflow for the Decision step:

1. In App Explorer, under **StepPaletteApp**, under **Application Workflows**, select **StepPaletteAppWF.**
2. In the application workflow editor, right-click the **Decision** transition, and select **Show Actions**.
3. On the **Actions** tab of the Property Editor, click **New.**

   The **Action Wizard** asks, "Which type of action do you want to execute?"

4. In the **Step 2** box, click the **and continue executing** link, select **and wait for reply**, and then click **Next**.

   The **Action Wizard** asks, "When do you want to call the orchestration workflow?"

5. In the **Step 1** box, select **After**, do not change anything in the **Step 2** box, and then click **Next**.

   The **Action Wizard** asks, "Which orchestration workflow do you want to call?"

6. On the menu under **Step 1**, select **(Add new workflow...)**.

7. In the **Event With Reply** dialog box that opens:

   a. Change the **Name** to `DecisionOrchWFWR`.

   b. On the **Orchestration** menu, select **StepPaletteOrch.**

   c. In the **Workflow** box, change the name to `DecisionOrchWF`.

   d. In the **Fields used by event** column, select the **Title** check box.

   e. In the **Fields returned by event** column, select the **Title** check box.

   f. Click **OK.**

8. In the **Action Wizard**, click **Finish.**

Practicing with the Decision Step

In this exercise, you add a **Decision** step to the DecisionOrchWF *orchestration workflow* [page 690] and define values for it. When DecisionOrchWF is invoked, it inserts `One`, `Two`, or `Otherwise` (depending on your input) in the **Title** box of the **State2** state form after you click the **OK** button on the *transition form* [page 700] between **State1** and **State2**. If you enter the number `1` and then click **OK**, `One` is inserted into the **Title** box. If you enter the number `2`, `Two` is inserted. If you enter any other text, `Otherwise` is inserted.

To use the Decision step in an orchestration workflow:

1. In App Explorer, under **StepPaletteOrch**, under **Orchestration Workflows**, select **DecisionOrchWF**.

2. In the **New Items** section of the **Step Palette**, drag and drop a **Decision** step onto the line between the **Start** and **End** steps.

3. Right-click the **Decision** step, and then select **Insert New Branch**.

   A new branch is added above the **Otherwise** branch.

4. On the **General** tab of the Property Editor, change the **Name** to `One`, and then press the Tab key.

5. On the **Options** tab, in the **Rule** section, enter the following function:
   `NORMALIZESPACE(EventNoticeWithReply.Extension.Title)="1"`.

6. In the **New Items** section of the **Step Palette**, drag and drop a **Calculate** step onto the **One** branch.

7. On the **Options** tab of the **Calculate** step Property Editor, in the **Target** section, enter the following expression: `EventNoticeWithReply.Extension.Title`
8. In the **Expression** section, enter the following text, including the quotation marks: `"One"`
9. Right-click the **Decision** step, and then select **Insert New Branch**.

   A new branch is added between the **One** branch and the **Otherwise** branch.
10. On the **General** tab of the Property Editor, change the **Name** to `Two`, and then press the Tab key.
11. On the **Options** tab, in the **Rule** section, enter the following function: `NORMALIZESPACE(EventNoticeWithReply.Extension.Title)="2".`
12. In the **New Items** section of the **Step Palette**, drag and drop a **Calculate** step onto the **Two** branch.
13. On the **Options** tab of the Property Editor, in the **Expression** section, enter the following text, including the quotation marks: `"Two"`
14. In the **Target** section, enter the following expression: `EventNoticeWithReply.Extension.Title`
15. In the **New Items** section of the **Step Palette**, drag and drop a **Calculate** step onto the **Otherwise** branch.
16. On the **Options** tab of the Property Editor, in the **Expression** section, enter the following text, including the quotation marks: `"Otherwise"`
17. In the **Target** section, enter the following expression: `EventNoticeWithReply.Extension.Title`
18. Select the **End** step.
19. On the **Data Mapping** tab, expand **Extension**, locate the **Title** data element, select the cell corresponding to the **Source elements** column, and then click the down arrow.
20. In the **Select a source** tool, expand **DecisionOrchWF**, **Inputs**, **EventNoticeWithReply**, and **Extension**; select **Title**; and then click **OK.**

Using the ForEach Step

The **ForEach** step repeats the operations that you specify for each element in a list or multi-item data element. These operations continue to execute until the last element has been processed. Then the *orchestration workflow* [page 690] continues to the next step. The ForEach index counts the number of loops, and the ForEach item is the value of the data element at the end of a particular loop.

When you select a **ForEach** step, the **General** and **Options** tabs appear in the Property Editor. The **General** tab provides a name and optional description for the step. The **Options** tab defines the source for the multi-item data element.

Procedure Summary

1. Select an orchestration workflow to display it in the orchestration workflow editor.

2. On the **Data Mapping** tab of the Property Editor, create an array of working data. (See Creating Arrays of Working Data [page 468].)

3. In the **New Items** section of the **Step Palette**, drag and drop a **ForEach** step onto the line between the **Start** and **End** steps.

   The lines from the **ForEach** step icon travel in two directions. One points to subsequent steps and the other loops back to the **ForEach** step icon. You place the steps or calculations that are to be repeated on the loop that points to the **ForEach** step icon.

4. On the **General** tab of the Property Editor, you can change the name of the **ForEach** step and you can also enter a description.

5. On the **Options** tab, in the **Source** section, enter an expression that describes the source of the data to be processed. You can use any of the functions, logical operators, or arithmetic operators available on the **Functions**, **Logical**, and **Operator** menus, respectively.

   For more information about expressions, see About the Expression Editor [page 442].

6. Add other steps from the **Step Palette** to the repeating section (the top loop) to define the actions that should be taken or calculations that should be performed while there are still members of the data element to be processed.

7. Add other steps from the **Step Palette** to the non-repeating section (the bottom loop) to define the actions that should be taken or calculations that should be performed after all of the members of the data element have been processed.

Tip: To duplicate or delete a **ForEach** step, right-click the step, and then select the appropriate option. To hide the loops while you work on another part of the orchestration workflow, click the minus sign to the left of the **ForEach** step.

Creating an Empty Orchestration Workflow for the ForEach Step

In this exercise, you create an empty synchronous orchestration workflow. In the next section, you add a **ForEach** step and then configure it.

To create an empty orchestration workflow for the ForEach step:

1. In App Explorer, under the **StepPaletteApp** heading, under **Application Workflows**, select **StepPaletteAppWF.**

2. In the application workflow editor, right-click the **ForEach** transition, and select **Show Actions**.

3. On the **Actions** tab of the Property Editor, click **New.**

   The **Action Wizard** asks, "Which type of action do you want to execute?"

4. In the **Step 2** box, click the **and continue executing** link, select **and wait for reply** on the menu, and then click **Next**.

   The **Action Wizard** asks, "When do you want to call the orchestration workflow?"

5. In **Step 1**, select **After**, do not change anything in the **Step 2** box, and then click **Next**.

The **Action Wizard** asks, "Which orchestration workflow do you want to call?"

6. On the menu under **Step 1**, select **(Add new workflow....)**.
7. In the **Event With Reply** dialog box that opens:
   a. Change the **Name** to `ForEachOrchWFWR`.
   b. On the **Orchestration** menu, select **StepPaletteOrch.**
   c. In the **Workflow** box, change the name to `ForEachOrchWF`.
   d. In the **Fields used by event** column, select the **Title** check box.
   e. In the **Fields returned by event** column, select the **Title** check box.
   f. Click **OK.**
8. In the **Action Wizard**, click **Finish.**

Practicing with the ForEach Step

In this exercise, you add a **ForEach** step to the ForEachOrchWF *orchestration workflow* [page 690] and define values for it. When ForEachOrchWF is invoked, it appends an equals sign and the current loop index value to the default value of each data element in the list of data elements under `Array[]`. After `Array[5]` is processed, the resulting values are inserted in the **Title** box of the **State3** state form after you click the **OK** button on the *transition form* [page 700] between **State2** and **State3**.

To use the ForEach step in an orchestration workflow:

1. In App Explorer, under **StepPaletteOrch**, under **Orchestration Workflows**, select **ForEachOrchWF**.
2. On the **Data Mapping** tab of the Property Editor, create an array of data elements as follows:
   a. Right-click the **WorkingData** step input, and then select **Add New DataElement**.
   b. While **DataElement** is still selected, type `ArrayContainer`, and then press the Tab key.
   c. Right-click **ArrayContainer**, point to **Type[String]**, and then select **Private Complex**.
   d. Right-click **ArrayContainer**, and then select **Add Child DataElement**.
   e. While **DataElement** is still selected, type `Array`, and then press the Tab key.
   f. Right-click **Array**, and then select **Properties Mode**.
   g. In the properties area under **Misc**, select the **IsUnbounded** row.
   h. Click the down arrow next to **False**, and then select **True**.
   i. To close the properties area, right-click **Array[]**, and then select **Mapping Mode**.

j. Right-click **Array[]**, and then select **Add Array Element**. **Array[1]** is added to **Array[]**.

k. Repeat the previous step four more times. Array elements **Array[2]**, **Array[3]**, **Array[4]**, and **Array[5]** are added to **Array[]**, under **Array[1]**.

l. In the **Default value** column for **Array[1]**, type the word `One`; for **Array[2]**, type `Two`; for **Array[3]**, type `Three`; for **Array[4]**, type `Four`; and for **Array[5]**, type `Five`.

3. Create a ForEach index:

   a. Right-click the **WorkingData** step input, and then select **Add New DataElement**.

   b. While **DataElement** is still selected, type `Index`, and then press the Tab key.

   c. Right-click **Index**, point to **Type[String]**, and then select **Integer** .

   d. In the **Default value** column, type `-1`.

4. Create a ForEach item value:

   a. Right-click the **WorkingData** step input, and then select **Add New DataElement**.

   b. While **DataElement** is still selected, type `Value`, and then press the Tab key.

   Leave the data type as **String**.

   c. In the **Default value** column, type `Test`.

5. In the **New Items** section of the **Step Palette**, drag and drop a **ForEach** step onto the line between the **Start** and **End** steps.

6. On the **Options** tab of the Property Editor, in the **Source** section, enter the following expression: `ArrayContainer.Array`

7. In the **New Items** section of the **Step Palette**, drag and drop a **Calculate** step onto the line that loops back to the **ForEach** step.

8. On the **General** tab of the Property Editor, change the **Name** to `SetIndex`, and then press the Tab key.

9. On the **Options** tab, in the **Target** section, type the following: `Index`

10. In the **Expression** section, enter the following expression: `ForEach.index`

11. In the **New Items** section of the **Step Palette**, drag and drop a **Calculate** step onto the line that loops back to the **ForEach** step, to the right of the **SetIndex** step.

12. On the **General** tab of the Property Editor, change the **Name** to `SetValue`, and then press the Tab key.

13. On the **Options** tab, in the **Target** section, type `Value`

14. In the **Expression** section, enter `ForEach.item`

15. In the **New Items** section of the **Step Palette**, drag and drop a **Calculate** step onto the line that loops back to the **ForEach** step, to the right of the **SetValue** step.
16. On the **General** tab of the Property Editor, change the **Name** to `MakeTitle`, and then press the Tab key.
17. On the **Options** tab, in the **Target** section, enter the following:
    `EventNoticeWithReply.Extension.Title`
18. In the **Expression** section, enter the following:
    `CONCAT(EventNoticeWithReply.Extension.Title," ", STRING(Index)," =", STRING(Value), " ")`
19. Select the **End** step.
20. On the **Data Mapping** tab of the Property Editor, under **Extension**, locate the **Title** data element, select the corresponding cell in the **Source elements** column, and then click the down arrow.
21. In the **Select a source** tool, expand **Inputs**, **EventNoticeWithReply**, and **Extension**; select **Title**, and then click **OK.**

Using the While Step

The **While** step repeatedly executes the operations you specify while the conditions defined in the rule are true, or until the conditions are false. The rule is an expression, as described in About the Expression Editor [page 442].

When you click a **While** step, the **General** and **Options** tabs appear in the Property Editor. The **General** tab provides a name and an optional description for the step. The **Options** tab defines the rule for the step.

Procedure Summary

1. Select an *orchestration workflow* [page 690] to display it in the orchestration workflow editor.
2. In the **New Items** section of the **Step Palette**, drag and drop a **While** step onto the line between the **Start** and **End** steps.
3. On the **General** tab of the Property Editor, you can change the name of the **While** step and you can also enter a description.
4. On the **Options** tab, in the **Rule** section, enter an expression that represents the rule that must hold true for the **While** step actions to repeat. You can use any of the functions, logical operators, or arithmetic operators available on the **Functions**, **Logical**, and **Operator** menus, respectively.

   For more information about expressions, see About the Expression Editor [page 442].
5. Add other steps from the **Step Palette** to the repeating section (the top loop) to define the actions that should be taken or calculations that should be performed while the **While** step is true.

6. Add other steps from the **Step Palette** to the nonrepeating section (the bottom loop) to define the actions that should be taken or calculations that should be performed if the **While** step is false.

Tip: To duplicate or delete the **While** step, right-click the step, and then select the appropriate options on the menu. To hide the loops as you work on another part of the orchestration workflow, click the minus sign to the left of the **While** step.

Creating an Empty Orchestration Workflow For the While Step

In this exercise, you create an empty synchronous *orchestration workflow* [page 690] with reply. In the next section, you add a **While** step and then configure it.

To create an empty orchestration workflow for the While step:

1. In App Explorer, under **StepPaletteApp**, under **Application Workflows**, select **StepPaletteAppWF.**
2. In the application workflow editor, right-click the **While** transition, and select **Show Actions**.
3. On the **Actions** tab of the Property Editor, click **New.**

   The **Action Wizard** asks, "Which type of action do you want to execute?"
4. In the **Step 2** box, click the **and continue executing** link, select **and wait for reply** on the menu, and then click **Next**.

   The **Action Wizard** asks, "When do you want to call the orchestration workflow?"
5. In the **Step 1** box, select **After**, do not change anything in the **Step 2** box, and then click **Next**.

   The **Action Wizard** asks, "Which orchestration workflow do you want to call?"
6. On the menu under **Step 1**, select **(Add new workflow...)**.
7. In the **Event With Reply** dialog box that opens:
   a. Change the **Name** to `WhileOrchWFWR`.
   b. On the **Orchestration** menu, select **StepPaletteOrch.**
   c. In the **Workflow** box, change the name to `WhileOrchWF`.
   d. In the **Fields used by event** column, select the **Title** check box.
   e. In the **Fields returned by event** column, select the **Title** check box.
   f. Click **OK.**
8. In the **Action Wizard**, click **Finish**.

Practicing With the While Step

In this exercise, you add a **While** step to the WhileOrchWF *orchestration workflow* [page 690] and define values for it. When WhileOrchWF is invoked, it counts the number of the strings in the **Title** box that consist of the letters `one`. If you type `oneoneoneoneone` in the **Title** box, the following text is inserted in the **Title** box of the **State3** state form when

you click the **OK** button on the *transition form* [page 700] between **State2** and **State3**: `I found 5 ones.`

To use the While step in an orchestration workflow:

1. In App Explorer, under **StepPaletteOrch**, under **Orchestration Workflows**, select **WhileOrchWF**.
2. On the **Data Mapping** tab of the Property Editor, create a loop counter as follows:
   a. Right-click the **Working Data** step input, and then select **Add New DataElement**.
   b. While **DataElement** is still selected, type `LoopCounter`, and then press the Tab key.

      Right-click **LoopCounter**, point to **Type[String]**, and then select **Integer**.
   c. In the **Default value** column, type the number `0`.
3. Create a temporary storage area for the working title as follows:
   a. Right-click the **Working Data** step input, and then select **Add New DataElement**.
   b. While **DataElement** is still selected, type `WorkingTitle`, and then press the Tab key. Leave **WorkingTitle** as a **String** data type.
   c. For the **WorkingTitle** data element, select the cell corresponding to the **Source elements** column, and then click the down arrow.
   d. In the **Select a source** tool, expand **Inputs**, **EventNoticeWithReply**, and **Extension**; select **Title**; and then click **OK**.
4. In the **New Items** section of the **Step Palette**, drag a **While** step onto the orchestration workflow editor, and drop it between the **Start** and **End** steps.
5. On the **Options** tab of the Property Editor, in the **Rule** section, enter the following function: `CONTAINS(WorkingTitle,"one")`
6. In the **New Items** section of the **Step Palette**, drag and drop a **Calculate** step onto the top loop (the repeating section) of the **While** step.
7. On the **General** tab of the Property Editor, change the **Name** to `Counter`, and then press the Tab key.
8. On the **Options** tab, in the **Target** section, enter the following: `LoopCounter`
9. In the **Expression** section, enter the following: `LoopCounter+1`
10. In the **New Items** section of the **Step Palette**, drag and drop a **Calculate** step onto the top loop of the **While** step, to the right of the **Counter** step.
11. On the **General** tab of the Property Editor, change the **Name** to `Title`, and then press the Tab key.
12. On the **Options** tab, in the **Expression** section, enter the following: `SUBSTRINGAFTER(WorkingTitle,"one")`

13. In the **Target** section, enter the following: `WorkingTitle`
14. In the **New Items** section of the **Step Palette**, drag and drop a **Calculate** step onto the bottom loop (the non-repeating section) of the **While** step.
15. On the **General** tab of the Property Editor, change the **Name** to `EndTitle`, and then press the Tab key.
16. On the **Options** tab, in the **Expression** section, enter the following function: `CONCAT("I found",STRING(LoopCounter)," ones.")`
17. In the **Target** section, enter the following: `WorkingTitle`
18. Select the **End** step.
19. On the **Data Mapping** tab, expand **Extension**, locate the **Title** data element, select the corresponding cell in the **Source Elements** column, and then click the down arrow.
20. In the **Select a source** tool, expand **WhileOrchWF**, **Inputs**, **EventNoticeWithReply**, and **Extension**; select **Title**; and then click **OK**.

Using the Service Step

You use the **Service** step to call a Web service from an *orchestration workflow* [page 690]. On the Property Editor for the **Service** step, you specify the WSDL file that describes the Web service and which of its defined operations to use.

When you select a **Service** step, the **General** and **Data Mapping** tabs appear in the Property Editor. The **General** tab provides a name and an optional description for the step and specifies which Web service and operation are to be used. The **Data Mapping** tab defines the default values or source elements for the inputs to the Web service.

Procedure Summary

1. Select an orchestration workflow to display it in the orchestration workflow editor.
2. In the **Configured Items** section of the **Step Palette**, drag and drop a **Service** step onto the line between the **Start** and **End** steps.
3. On the **General** tab of the Property Editor, you can change the name of the **Service** step and you can also enter a description.
4. On the **Service** menu, select the WSDL file for the Web service that you want to use.
5. On the **Data Mapping** tab, you can identify the inputs, or data elements, that are required by the Web service for the selected operation by the *required* flag that appears on the icon associated with the data element.

   If you do not see the required flag, right-click in any part of the data mapping area and select **Show Required Flag**.
6. To define default values for any of the Web service inputs, enter them in the **Default values** column.

7. To map the source values for a specific input using suggested mappings, see Setting Source Values Using Suggested Mappings [page 470].

   The outputs of the Web service appear as potential inputs for subsequent steps in the orchestration workflow.

   Tip: To duplicate a **Service** step, right-click the step, and then select **Duplicate**.

Creating an Empty Orchestration Workflow For the Service Step

In this exercise, you create an empty synchronous *orchestration workflow* [page 690] with reply. In the next section, you add a **Service** step and then configure it.

To create an empty orchestration workflow for the Service step:

1. In App Explorer, under **StepPaletteApp**, under **Application Workflows**, select **StepPaletteAppWF.**
2. In the application workflow editor, right-click the **Service** transition, and select **Show Actions**.
3. On the **Actions** tab of the Property Editor, click **New.**

   The **Action Wizard** asks, "Which type of action do you want to execute?"
4. In the **Step 2** box, click the **and continue executing** link, select **and wait for reply**, and then click **Next**.

   The **Action Wizard** asks, "When do you want to call the orchestration workflow?"
5. In the **Step 1** box, select **Before**, and then click **Next**.

   The **Action Wizard** asks, "Which orchestration workflow do you want to call?"
6. On the menu under **Step 1**, select **(Add new workflow...)**.
7. In the **Event With Reply** dialog box that opens:
   a. Change the **Name** to `ServiceOrchWFWR`.
   b. On the **Orchestration** menu, select **StepPaletteOrch.**
   c. In the **Workflow** box, change name to `ServiceOrchWF`.
   d. In the **Fields used by event** column, select the **Title** check box.
   e. In the **Fields returned by event** column, select the **Title** check box.
   f. Click **OK.**
8. In the **Action Wizard**, click **Finish**.
9. In App Explorer, under **StepPaletteApp**, under **Orchestration Links**, select **ServiceOrchWFWR**.
10. On the **ServiceOrchWFWR** tab:
    a. In the **Fields used by event** column, select the **Item Id** check box. The **Title** check box should already be selected.

b. In the **Fields returned by event** column, the **Title** check box should be selected.

Practicing with the Service Step

In this exercise, you add a **Service** step to the ServiceOrchWF *orchestration workflow* [page 690] and define values for it. When ServiceOrchWF is invoked, it inserts your login ID in the **Title** box of the transition page between **State4** and **State5** when you click the **Service** button on the **State4** state form.

To use the Service step in an orchestration workflow:

1. In App Explorer, under **StepPaletteOrch**, under **Orchestration Workflows**, select **ServiceOrchWF**.
2. In the **New Items** section of the **Step Palette**, drag a **Service** step onto the orchestration workflow editor, and drop it between the **Start** and **End** steps.
3. On the **General** tab of the Property Editor, on the **Service** menu, select the **sbmappservices72** Web service.
4. On the **Operation** menu, select the **GetItem** Web service operation.
5. On the **Data Mapping** tab, expand the **Auth** data element, and in the **Default value** column, enter your SBM user ID and password.
6. Still on the **Data Mapping** tab, locate the **TableId_ItemId** data element, select the corresponding cell in the **Source elements** column, and then click the down arrow.
7. In the **Select a source** tool, expand **Inputs**, **EventNoticeWithReply**, and **Extension**; select **ItemId_TableRecId**; and then click **OK**.
8. In the orchestration workflow editor, select the **End** step.
9. On the **Data Mapping** tab of the Property Editor, under **Extension**, locate the **Title** data element, select the corresponding cell in the **Source elements** column, and then click the down arrow.
10. In the **Select a source** tool, expand **sbmappservices72_GetItem**, **Outputs**, **GetItemResponse**, **return**, **item**, and **createdBy**; select **loginId**, and then click **OK.**

Data Mapping Tab of the Service Step Property Editor

The **Data Mapping** tab of the **Service** step Property Editor is available when you select a **Service** step in an orchestration workflow in the orchestration workflow editor.

Element	Description
Step inputs	This column lists the inputs to the Web service associated with the selected **Service** step. Right-click a step input to display a menu of applicable commands.

Element	Description
Source elements	This column shows the source of each of the step inputs. Select a cell in this column and click the down arrow that appears to open the **Select a source** tool, which offers options beyond the suggested mappings. Right-click a source element to display a menu of applicable commands.
Default value	This column allows you to view or edit the default values, if any, for the step inputs. **Note:** Any value specified in the **Source elements** column overrides the value in the corresponding **Default value** column.
Vertical divider	Clicking the vertical divider on the right of the Property Editor switches the Property Editor between mapping mode and properties mode, which displays additional information for the selected data element (such as its type and namespace). To see the inputs from the state before the transition to which the orchestration workflow has been added, click on a blank area of the orchestration workflow editor, and then click the vertical divider.

Mapping SOAP Header Data

SBM Composer lets you map SOAP Header information, if the WSDL file you are using defines SOAP Header data. This data is listed on the **Data Mapping** tab of a **Service** step in the **Step Inputs** column, and its name ends in _Envelope.

Note: SBM Composer does not support `headerfault message` elements.

Procedure Summary

1. In the orchestration workflow editor, select the **Service** step for which you want to map data that will be sent in the SOAP Header. The Web service's WSDL file must define SOAP Header data.
2. On the **General** tab of the Property Editor, make sure that the **Service** step is associated with a Web service and an operation.
3. On the **Data Mapping** tab of the Property Editor, expand the **_Envelope** step input.

   All defined SOAP Header data is listed as data elements under this step input. Step inputs that appear below and at the same level as the **_Envelope** step input and all of their data elements are defined in the SOAP Body.
4. For each of the required data elements, do one of the following:
   - In the **Source elements** column:
     1. Click the down arrow.

2. In the **Select a source** tool, select the data element to be used from another location in the workflow, such as the output of a **Service** step or a working data element.

3. Click **OK**.

- In the **Default values** column, type the path of the data element to be used, such as `SomeWorkflow\FirstCategory\SecondItem`.

Using Basic Access Authentication

Some *Web services* [page 701] require authentication, such as a user name and a password. In one common method, called *basic access authentication*, this information is included in the HTTP header of the Web service call. SBM Composer supports basic access authentication for individual Web service operations within an *orchestration workflow* [page 690].

For example, you can create two working data elements to hold a user name and a password, enter text as default values, and set up the **Authentication** input's **UserName** and **Password** components for multiple **Service** steps to use those working data elements. That way, if the user name or password changes, you only need to update the two working data elements to keep all of the **Service** steps set up to call the Web service.

Important: The **UserName** and **Password** values are ignored if an administrator configures the basic access authentication type for the *endpoint* [page 684] pointing to the Web service manually in Application Administrator. In this case, the credentials specified in Application Administrator are used instead of those specified in this procedure. Process apps are more portable if the Application Administrator credentials are used, because you can bind to the appropriate endpoint for your environment when you deploy.

Procedure Summary

1. Select an orchestration workflow to display it in the orchestration workflow editor.
2. Select a **Service** step for which you want to set up basic access authentication.
3. On the **General** tab of the Property Editor, make sure the **Service** step is associated with a Web service and an operation, and that the Web service supports basic access authentication.
4. In the orchestration workflow editor, right-click the **Service** step, and then select **Use Basic Access Authentication**.

   This adds the **Authentication** step input (with the **UserName** and **Password** data elements), to the **Data Mapping** tab of the Property Editor. These data elements can be used for the **Service** step and for any other **Service** steps in the orchestration workflow that are associated with the same Web service operation.
5. For each of the **Service** steps to which the **Authentication** step input was added, do one of the following for both the **UserName** and **Password** data elements:
   - In the **Source elements** column:
     1. Click the down arrow.

2. In the **Select a source** tool that opens, select the source value to be used from some location in the workflow, such as the output of another **Service** step or a working data element.
3. Click **OK**.

- In the **Default values** box, type the value to be used.

Using Dynamic Endpoints

SBM Composer provides the option of setting the endpoints for individual Web service operations dynamically, when the *orchestration workflow* [page 690] is run, to a URL returned by a different **Service** step.

For example, a login **Service** step returns the URL of a server to be used for subsequent **Service** steps. For each subsequent **Service** step, you can use the procedure below to map the **DynamicEndpointURL** input to the server URL included in the output of the login **Service** step.

Procedure Summary

1. Select an orchestration workflow to display it in the orchestration workflow editor.
2. Select a **Service** step for which you want to set a dynamic endpoint.
3. On the **General** tab of the Property Editor, make sure the **Service** step is associated with a Web service and an operation.
4. Right-click the **Service** step in the orchestration workflow, and then select **Use Dynamic Endpoint**.

   This adds the **DynamicEndpointURL** step input to the **Data Mapping** tab of the Property Editor. This data element can be used for the **Service** step and for any other **Service** steps in the orchestration workflow that are associated with the same Web service operation.
5. For each **Service** step to which the **DynamicEndpointURL** step input has been added:
   a. Select the **Source elements** column in the cell corresponding to the new **DynamicEndpointURL** step input, and then click the down arrow.
   b. In the **Select a Source** tool, select the URL to be used.
   c. Click **OK**.

Note: You can return a Web service operation to using a static endpoint rather than a dynamic endpoint. To do this, in the orchestration workflow editor, right-click a **Service** step associated with the Web service operation, and then select **Use Dynamic Endpoint**.

Running the StepPalette Process App

You are now ready to deploy and run the StepPalette process app.

To deploy and run the StepPalette process app:

1. *Publish* [page 692] and deploy the StepPalette process app.

See Step 6: Publish the Process App [page 570] and Step 7: Deploy the Process App [page 570] for instructions.

2. Log on to the SBM User Workspace.
3. Click the **StepPaletteApp** tab. If this tab is not visible, click the **More** tab, and then select **StepPaletteApp** from the list.
4. On the **StepPaletteApp Task Page**, under **Submit**, click the **Submit a new StepPaletteProcessApp** link.
5. Under the **Submit Tree**, click the **StepPaletteAppWF Project** link.

   The **Submit** transition form opens.
6. On the *transition form* [page 700], type `The process apps` in the **Title** box, and then click the **OK** button.

   The Calculate orchestration workflow is invoked, and it prepends the text in the **Title** box of the **State1** state form, which now contains `The process apps are here.`
7. On the **State1** state form, click the **Decision** button.

   The transition form between the **State1** and **State2** state forms opens.
8. On the transition form, delete the text in the **Title** box, type `1`, `2`, or any other text, and then click the **OK** button.

   The Decision orchestration workflow is invoked, and the results are displayed in the **Title** box of the **State2** state form. If you entered `1` in the **Title** box, then `One` is displayed. If you entered `2`, the **Title** box contains `Two`. And if you entered any other text, the **Title** box contains `Otherwise`.
9. On the **State2** state form, click the **ForEach** button.

   The transition form between the **State2** and **State3** state forms opens.
10. On the transition form, delete the text in the **Title** box, type `Results:` and a space, and then click the **OK** button.

    The ForEach orchestration workflow is invoked, and the results are displayed in the **Title** box of the **State4** form. The **Title** box contains `Results: One = 1 Two = 2 Three = 3 Four = 4 Five = 5`.
11. On the **State3** state form, click the **While** button.

    The transition form between the **State3** and **State4** state forms opens.
12. Delete the text in the **Title** box, enter the word `one` zero to eight times, and then click **OK**.

    The While orchestration workflow is invoked, and the results are displayed in the **Title** box of the **State3** state form. If *n* is the number of times you entered `one` in the **Title** box, `I found` *n* `ones.` is displayed.
13. On the **State4** form, click the **Service** button.

    The transition form between the **State4** and **State5** state forms opens.
14. On the transition form, click the **OK** button.

The Service orchestration workflow is invoked, and your login ID appears in the **Title** box of the **State5** state form.

Using the Scope, Throw, and Compensate Steps to Handle Faults From Web Services

The **Scope**, **Throw**, and **Compensate** steps in SBM Composer graphically represent *BPEL* [page 680] fault-handling elements. You use these steps to handle Web service faults. This section provides a brief overview of BPEL fault handling and how it is implemented in SBM Composer.

Types of Web Service Faults

The two types of Web service faults are "named" faults and "generic" faults. A named fault is associated with a specific fault situation, such as entering an incorrectly formatted user name or attempting to withdraw money from a bank account that is inactive. A generic fault is returned by a Web service if the service does not specify any named faults. *Web services* [page 701] return faults in SOAP messages that contain SOAP faults. (See About SOAP Messages [page 445].)

If a Web service fault is not handled, or caught, it can stop the entire business process. Unhandled Web service faults are the most common causes of orchestration workflow failures. This section explains the various ways you can use the fault-handling components in SBM Composer to keep your business processes flowing smoothly.

Scope Step

The **Scope** step lets you group related activities. It contains a **FaultHandler** section and a **CompensationHandler** section. Scopes can be nested, that is, one scope can enclose another scope. Orchestration workflows are contained within implicit, or global, scopes.

FaultHandler Section

If a Web service inside of a scope generates a SOAP fault or if an internal BPEL fault occurs, the SBM Orchestration Engine checks whether a fault handler is defined for the scope. The catch branch within the fault handler "catches" the faults so they can be handled. The fault handler can contain an unlimited number of catch branches.

If the SBM Orchestration Engine finds a fault handler, it selects the catch branch with the value that best matches the fault according to the following rules:

- You can right-click a **Service** step in a scope and select **Add Catch Branches to Scope** to add a catch branch for each fault the step can throw. The SBM Orchestration Engine selects the appropriate catch branch when the step throws a fault.
- **ServiceFlowFault** is the only fault that a **Throw** step can throw. The catch branch with **ServiceFlowFault** selected as the **Fault name** on the **General** tab of its Property Editor is selected for **Throw** steps that have a **Fault message** displayed on the **General** tab of the step Property Editor.

  Important: If there are several **Throw** steps within the same **Scope**, each one takes this branch, even if they have different **Fault message** values.
- If no catch or catch all branches are selected, the fault is not caught and is thrown back to the immediately enclosing scope.

CompensationHandler Section

The SBM Orchestration Engine invokes compensation handlers when there is a **Compensate** step in the enclosing scope. You can use any steps in the **Step Palette** to create a compensation handler. For an example, see Using the Compensate Step [page 525].

Throw Step

During the execution of an *orchestration workflow* [page 690], errors might prevent the workflow from completing. For example, suppose you design an orchestration workflow that transfers money between two accounts by invoking a few Web services. However, the first Web service call returns an "invalid account number" fault.

The orchestration workflow stops, unless it contains some logic in the fault handler of the scope that encloses the Web service. For example, you might want to inform the user that he or she entered an invalid account number. The best way to do this by using the **Throw** step to generate a custom error message such as "The account number you entered is not valid. Please try again."

Orchestration Workflows as Web Services

An orchestration workflow itself is invoked as a Web service, although this Web service can only be accessed by the SBM Application Engine. An orchestration workflow is invoked in one of two modes: Request Only (EventNotice) or Request and Respond (EventNoticeWithReply). EventNotice is used when an asynchronous orchestration workflow is invoked, and EventNoticeWithReply is used when a synchronous orchestration workflow is invoked.

A Web service could respond with a SOAP fault if something went wrong during its execution. For each orchestration workflow that is invoked, a WSDL file is generated for its corresponding Web service.

The following sample code is a EventNoticeWithReply Web service operation that is defined in a WSDL file. It shows that the Web service for the orchestration workflow can return a SOAP fault named ServiceFlowFault. All orchestration workflows can return, or throw, a ServiceFlowFault. In fact, this is the only SOAP fault that they can throw.

```
<wsdl:operation name="EventNoticeWithReply">
  <wsdl:input message="tns:XxxxxEventNoticeWithReply" />
  <wsdl:output message="tns:XxxxxEventNoticeWithReplyResponse" />
  <wsdl:fault name="ServiceFlowFault" message="tns:ServiceFlowFault" />
  </wsdl:operation>
```

Why Use a Throw Step?

Use a **Throw** step when you want an orchestration workflow to explicitly signal an internal fault. By inserting a **Throw** step in a synchronous orchestration workflow, you can do the following:

1. Notify a user that the synchronous orchestration workflow encountered a problem and cannot continue.
2. Together with a fault handler, notify a user that the invocation of a Web service within this orchestration workflow returned a SOAP fault.
3. Immediately stop the execution of the scope that encloses the step.

4. Log the contents of the fault that is thrown by the **Throw** step and display it in the Log Viewer.

If a Web service that is invoked within a synchronous orchestration workflow receives a fault as a reply, the user of your process app does not see the fault message, unless you provided a fault handler to catch the fault. Instead, he or she will see the following generic error message in the SBM User Workspace: "An error occurred during the execution of the synchronous orchestration workflow." If you want to display the fault message to the user, enclose the Web service invocation within a scope, catch the fault that is thrown by the Web service, and throw a ServiceFlowFault using a **Throw** step. See Using the Throw Step [page 521] for more information.

If a Web service is invoked within an asynchronous orchestration workflow, you cannot notify a user, as previously described in steps 1 and 2, because asynchronous orchestration workflows do not return anything to the caller, which is the SBM Application Engine. However, an asynchronous orchestration workflow can accomplish the tasks described in items 3 and 4.

How to Use the Throw Step

The procedure for using a **Throw** step is described in Using the Throw Step [page 521].

If a **Throw** step is not enclosed within a scope with a fault handler, the whole workflow is stopped after that step is executed. In addition, the content of its FaultString data element is displayed in the SBM User Workspace if the orchestration workflow is synchronous.

If a **Throw** step is enclosed within a scope that has a fault handler, the fault handler can catch the ServiceFlowFault that is thrown by the **Throw** step. To do this, you can use either a **Catch** branch for a named fault or the **CatchAll** branch. After the ServiceFlowFault is caught, you could use another **Throw** step to throw a new ServiceFlowFault to the SBM Application Engine. The orchestration workflow that you create in Using the Throw Step [page 521] demonstrates how to do this.

Summary

1. The only fault that can be thrown by a **Throw** step is ServiceFlowFault.
2. The content of the FaultString data element of the ServiceFlowFault is displayed in the SBM User Workspace, if the fault is thrown within a synchronous orchestration workflow.
3. The content of the ServiceFlowFault is logged when the fault is thrown and can be used during debugging.
4. After a **Throw** step is executed, the activity in the next enclosing scope is stopped.

Compensate Step

During the execution of an orchestration workflow, you might need to reverse, or compensate, an activity that already completed. For example, say you design an orchestration workflow that transfers money between two accounts by invoking a few Web services. One Web service call in the orchestration workflow withdraws funds from the first account and records the transaction. Another Web service call checks the status of the second account and returns an "inactive account" fault. A third Web service call pays back the funds to the first account because the second account is inactive.

An inner scope contains the Web service that withdraws funds. This scope has a CompensationHandler that calls the Web service that pays back the funds. An outer scope encloses the inner scope. The outer scope contains the Web service that checks whether the second account is valid. The outer scope also contains a FaultHandler, in which a proper **Catch** branch contains the **Compensate** step. When the "inactive account" fault is caught, the **Compensate** step is executed.

Tip: You could also use a **Throw** step to generate a custom error message such as "This account is inactive, so the transfer transaction could not be completed."

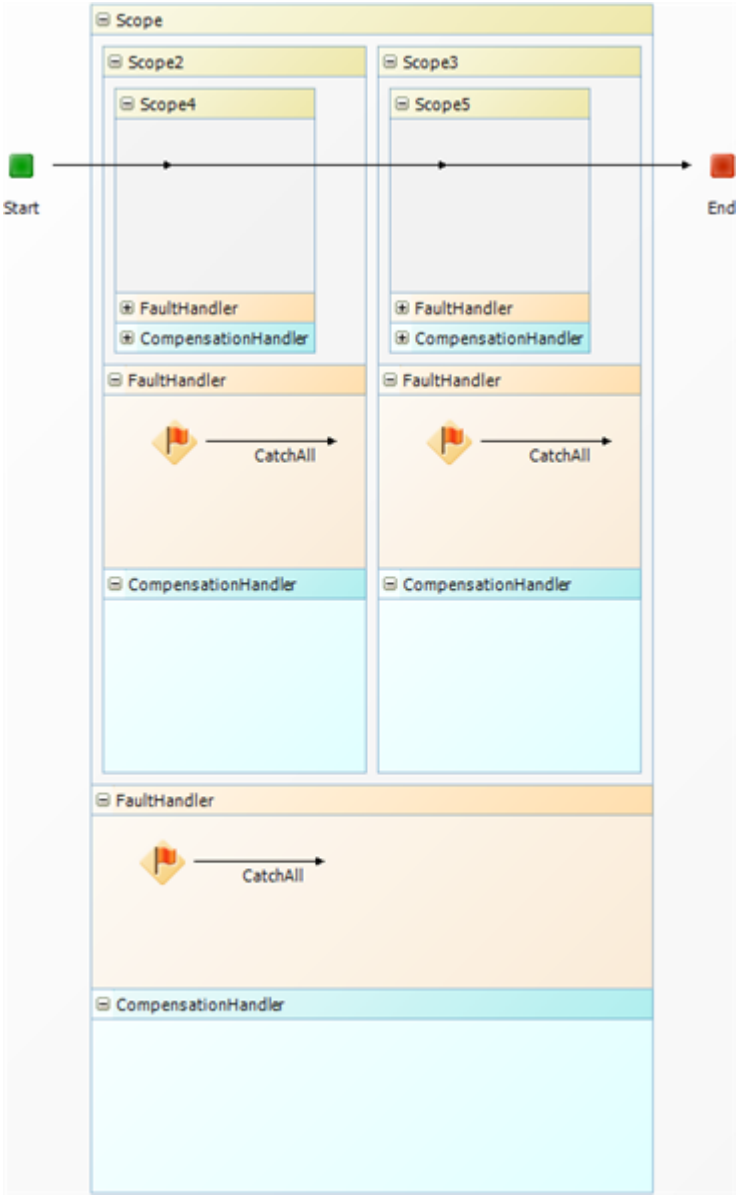
Note: You can only place a **Compensate** step in the **FaultHandler** or **CompensationHandler** section of a scope.

Configuring a Compensate Step

When you configure the **Compensate** step, you can select the Default Scope or a specific scope.

- If you select the Default Scope, the SBM Orchestration Engine checks all immediately enclosed scopes that completed successfully to determine if they have compensation handlers defined in them. If multiple scopes have completed successfully and all have compensation handlers, these scopes are called one by one in reverse order, that is, the one that completed last is called first. For example, scope "Scope" immediately encloses scopes "Scope2" and "Scope3." If there is a **Compensate** step in the **CompensationHandler** of scope "Scope," the SBM Orchestration Engine runs the compensation handlers in scope "Scope3" and then in "Scope2," if they both completed successfully. To change the order in which the scopes are called, you can place more than one **Compensate** step in a **CompensationHandler** section.
- If you select a specific scope, the *BPEL engine* [page 680] only checks that scope to determine if it completed successfully and if it has a compensation handler defined in it. If it completed successfully and has a compensation handler, the compensation handler is called. The compensation handler in this scope can then call other scopes for compensation if it designed to do so.

Figure 1. Enclosed Scopes



Tutorial: Creating a Practice Process App for Fault Handling

In this section, you create a new process app named FaultHandlingProcApp. You will use this process app to practice using the **Scope**, **Throw**, and **Compensate** steps.

You can run the process app at any time after you create an *application workflow* [page 680] and deploy the process app. To run the application workflow (project), follow the instructions in the relevant sections of Running the Fault Handling Process App [page 530].

To create the practice process app:

1. Start SBM Composer.

2. Click the Composer button, and then click **New**.

   The **Create New Process App** dialog box opens.

3. Click **Application Process App**, and then click **Create**.

   The **Configure Process App** dialog box opens.

4. In the **Process app name** and **Application name** boxes, type `FaultHandlingProcApp`, and then click **OK.**

5. In App Explorer, click the **All Items** filter.

6. Under **Tables**, click **FaultHandlingProcessApp**.

7. In the Property Editor, change the **Name** to `FaultHandlingPTable`, and then press the Tab key.

   PTable stands for *primary table* [page 692].

8. In the **System Fields** section of the **Table Palette**, drag and drop a **Description** field onto the **FaultHandlingPTable (Primary)** tab.

   The *Description* field is added to the table, in alphabetical order. The *Description* field will also be added to the state and transition forms. Information that is returned by the orchestration workflows you create in this section is displayed in this field and in the *Title* field.

9. In App Explorer, click **FaultHandlingProcessApp**.

10. On the **FaultHandlingProcApp** tab, change the **Logical name** to `FaultHandlingApp`, and then press the Tab key.

11. Perform the following steps to create **GenericFaultAWF**:

    a. In App Explorer, under **Application Workflows**, click **FaultHandlingProcApp**.

    b. On the **General** tab of the Property Editor, change the **Name** to `GenericFaultAWF`, and then press the Tab key.

    c. In the application workflow editor, click the **Submit** transition.

    d. On the **General** tab of the Property Editor, change the **Name** to `VerifyUser`, and then press the Tab key.

    e. Click the **New** state.

    f. On the **General** tab of the Property Editor, change the **Name** to `IsUserValid`, and then press the Tab key.

12. Perform the following steps to create **NamedFaultAWF**:

    a. In App Explorer, right-click **Application Workflows**, and then click **Add New Workflow**.

    Under **Application Workflows**, **FaultHandlingApp Workflow** should be selected.

    b. On the **General** tab of the Property Editor, change the **Name** to `NamedFaultAWF`, and then press the Tab key.

c. In the application workflow editor, click the **Submit** transition.

d. On the **General** tab of the Property Editor, change the **Name** to `GetTickerSymbol`, and then press the Tab key.

e. Click the **New** state.

f. On the **General** tab of the Property Editor, change the **Name** to `TickerSymbol`, and then press the Tab key.

g. In the **States** section of the **Workflow Palette**, drag **Active** onto the application workflow editor and drop it to the right of the **TickerSymbol** state.

h. Change the **Name** to `BuyRating`, and then press the Tab key.

i. In the **Transitions** section of the **Workflow Palette**, drag **Regular** onto the **TickerSymbol** state, release the mouse button, and then click **BuyRating**.

j. Change the name of the **Transition** to `GetBuyRating`, and then press the Tab key.

13. Perform the following steps to create **ThrowAWF**:

a. In App Explorer, right-click **Application Workflows**, and then select **Add New Workflow**.

FaultHandlingAppWorkflow appears under **NamedFaultAWF**.

b. Click **FaultHandlingAppWorkflow**.

c. On the **General** tab of the Property Editor, change the **Name** to `ThrowAWF`, and then press the Tab key.

d. In the **States** section of the **Workflow Palette**, drag **Active** onto the orchestration workflow editor, and drop it to the right of the **New** state.

e. On the **General** tab of the Property Editor, change **Name** to `TickerSymbol`.

f. In the **Transitions** section of the **Workflow Palette**, drag **Regular** onto the **New** state, release the mouse button, and then click the **TickerSymbol** state.

g. On the **General** tab of the Property Editor, change the **Name** to `GetTickerSymbol`.

14. Perform the following steps to create **CompensateAWF**:

a. In App Explorer, right-click **Application Workflows**, and then select **Add New Workflow**.

FaultHandlingAppWorkflow appears under **ThrowAWF**.

b. Click **FaultHandlingAppWorkflow**.

c. On the **General** tab of the Property Editor, change the **Name** to `CompensateAWF`, and then press the Tab key.

d. In the **States** section of the **Workflow Palette**, drag **Active** onto the orchestration workflow editor, and drop it to the right of the **New** state.

e. On the **General** tab of the Property Editor, change **Name** to `TickerSymbol`.

f. In the **Transitions** section of the **Workflow Palette**, drag **Regular** onto the **New** state, release the mouse button, and then click the **TickerSymbol** state.

g. On the **General** tab of the Property Editor, change the **Name** to `DemoCompensate`.

15. In App Explorer, right-click **FaultHandlingProcessApp**, point to **Add New**, and then select **Orchestration**.

    The **New Orchestration** dialog box opens.

16. In the **Name** box, type `FaultHandlingOrch`, and then click **OK**.

17. Perform the following steps to import the SerenaSampleTickerService:

    a. In App Explorer, under **FaultHandlingOrch**, right-click **Web Services**, and then select **Add New Web Service**.

       The **Web Service Configuration** dialog box opens.

    b. In the **WSDL** box, enter the following URL: `http://serverName/Ticker/services/SerenaSampleTickerService?wsdl`

       Note: In the on-demand environment, *serverName* is the URL that you use to access Application Administrator. For example, the server name for Acme Company is `http://acme.admin.serenaprocessapps.com`. In the on-premise environment, *serverName* is the name of the server (or local machine) that is running the JBoss server. The *serverName* for most local machines is `localhost:8085`.

    c. Click **OK**.

       SerenaSampleTickerService appears in App Explorer under **Web Services**.

18. Save the process app:

    a. On the Quick Access Toolbar, click the **Save locally** button.

       A message box opens reminding you that the design elements have been saved to the *Local Cache* [page 688] only.

    b. Click **OK**.

Using the Scope Step

The **Scope** step includes a **FaultHandler** section and a **CompensationHandler** section. The **FaultHandler** is for handling Web service (SOAP) faults. By default, it has a **CatchAll** branch, which also catches faults generated by the *BPEL engine* [page 680]. You can also add other **Catch** branches. Each **Catch** branch can handle a specific named Web service fault.

You can optionally add a **Throw** step to the **FaultHandler** section and a **Compensation** step to the **CompensationHandler** section. The **Throw** step is explained in Using the Throw Step [page 521], and the **CompensationHandler** is covered in Using the Compensate Step [page 525].

Procedure Summary

1. In App Explorer, select an *orchestration workflow* [page 690] to display it in the orchestration workflow editor.

2. In the **New Items** section of the **Step Palette**, drag a **Scope** step onto the orchestration workflow editor, and drop it onto the line between the **Start** and **End** steps.

You can change the name of the **Scope** step in the Property Editor, and you can also add a description.

3. In the **Step Palette**, drag a **Service** step associated with a Web service onto the orchestration workflow editor, and drop it into the **Scope** section.
4. If the Web service returns named faults, you can perform the following steps to handle them:
   a. Expand the **FaultHandler** section.
   b. Add and configure **Catch** branches one at a time, or automatically add and configure all **Catch** branches for a Web service operation.
      - To add **Catch** branches one at a time:
        1. Right-click the **Fault Handler** step, and then select **Insert New Catch**.

           A new **Catch** branch appears above the **CatchAll** branch.
        2. Select the new **Catch** branch.
        3. On the **General** tab of the Property Editor, specify a Web service fault by selecting it from the **Fault name** list. You can also change the name of the **Catch** branch and provide a description.

           Note: If there is an associated fault message, it is displayed in the **Fault message** box as read-only. If there is no message, the **Fault message** box is empty. See Rules for Configuring the Catch Branch [page 520] for more information.
        4. In the **Step Palette**, you can drag any steps that you want to use to handle the fault into the **Scope** section and drop them onto the **Catch** branch. Then configure the steps as required.
        5. Repeat these steps to add and configure other **Catch** branches.
      - To automatically add all **Catch** branches associated with a Web service operation:
        1. In the top section of the **Scope** step, right-click the **Service** step, and then select **Add Catch Branches to Scope**.

           Catch branches are automatically added to the **FaultHandler** section.
        2. In the **Step Palette**, you can drag any steps that you want to use to handle the faults into the **Scope** section and drop them onto a **Catch** branch. Then configure the steps as required.
   c. To handle faults that are not handled by the **Catch** branches, you can drag any steps that you want to use to handle these faults into the **Scope** section and drop them onto the **CatchAll** branch. Then configure the steps as required.

      You cannot specify a fault name and a fault message for the **CatchAll** branch, because it is invoked if none of the other branches can handle the fault.
5. If the Web service does not return named faults, perform the following steps to handle the faults returned by the Web service:
   a. Expand the **FaultHandler** section.

b. In the **Step Palette**, you can drag any steps that you want use to handle the faults into the **Scope** section and drop them onto the **CatchAll** branch. Then configure the steps as required.

Note: To hide the sections of the **Scope** step as you work on another part of the orchestration workflow, click the minus sign to the left of the **Scope** step.

Tutorial: Creating An Empty Synchronous Orchestration Workflow to Handle Generic Web Service Faults

In this exercise, you add an empty synchronous *orchestration workflow* [page 690] to the FaultHandlingOrch in the FaultHandlingProcApp.

To create an empty orchestration workflow that handles generic Web service faults:

1. In App Explorer, under **Application Workflows**, click **GenericFaultAWF.**
2. In the application workflow editor, right-click the **VerifyUser** transition, and then select **Show Actions**.
3. On the **Actions** tab of the Property Editor, click **New.**

   The **Action Wizard** opens and asks, "Which type of action do you want to execute?"

   Under **Step 1**, **Orchestration Workflow** should be selected.
4. Under **Step 2**, click the **and continue executing** link, and then select **and wait for reply**.
5. Click **Next**.

   The **Action Wizard** asks, "When do you want to call the orchestration workflow?"
6. Under **Step 1**, select **After**.

   Step 2 should read **Invoke an orchestration workflow and wait for reply, after this transition occurs**.
7. Click **Next**.

   The **Action Wizard** asks, "Which orchestration workflow do you want to call?"
8. Under **Step 1**, select **(Add new workflow...)**.

   The **Event with Reply** dialog box opens.
9. In the **Name** box, type `GenericFaultWFWR`.
10. In the **Workflow** box, type `GenericFaultOWF`.
11. In the **Fields used by event** list, select the **Title** check box.
12. In the **Fields returned by event** list, select the **Description** and **Title** check boxes.
13. Click **OK**.

    Under **Step 1**, **GenericFaultWFWR** is selected on the menu and **EventNoticeWithReply** appears in the list. **Step 2** now reads **Invoke an**

orchestration workflow and wait for reply, after this transition occurs, Call GenericFaultWFWR.EventNoticeWithReply.

14. Click **Finish**.

    GenericFaultWFWR appears under **Orchestration Links**, and **GenericFaultOWF** appears under **Orchestration Workflows** in App Explorer.

Tutorial: Practicing With the Scope Step to Handle Generic Web Service Faults

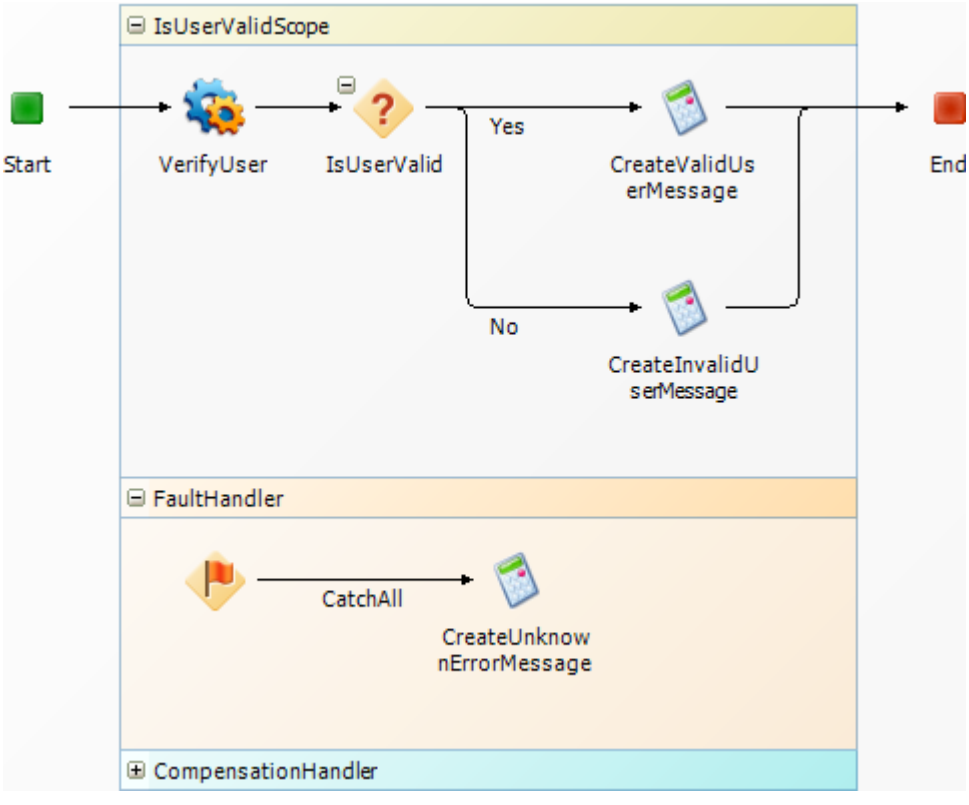
In this exercise, you use a **Scope** step in the GenericFaultOWF *orchestration workflow* [page 690].

Note: Even though the SBM Application Engine actually returns a named fault, for demonstration purposes, this exercise assumes that is does not.

Important: If the server that is running Serena Business Manager does not support Single Sign-On (SSO), you must provide authentication information every time you use an Application Engine Web service operation, as in step 5. This information is usually entered on the **Data Mapping** tab in the **Default** column of the **userId** and **password** data elements under the **auth** data element. If you do not know whether your server supports SSO, ask your system administrator.

After you complete the steps in this exercise, your orchestration workflow should look like the one in the following figure:

Figure 1. GenericFaultOWF



Later, when you run the GenericFaultApp Project, you will alter this orchestration workflow so it returns an error message in the **Description** field.

To use the Scope step in an orchestration workflow to handle generic Web service faults:

1. In App Explorer, under **Orchestration Workflows**, click **GenericFaultOWF**.
2. Create a working data element of type String to hold the message that is passed to the **Description** field as follows:
   a. Click a blank area of the orchestration workflow editor.
   b. On the **Data Mapping** tab of the Property Editor, under **GenericFaultOWF**, right-click **Working Data**, and then select **Add New DataElement**.
   c. Change the name of the new **DataElement** to `Message`.
3. In the **New Items** section of the **Step Palette**, drag a **Scope** step onto the orchestration workflow editor, and drop it between the **Start** and **End** steps.
4. On the **General** tab of the Property Editor, change the **Name** to `IsUserValidScope`, and then press the Tab key.
5. In the **Configured Items** section of the **Step Palette**, drag an **sbmappservices72** Service step onto the orchestration workflow editor, and drop it onto the line inside the top section of the **Scope** step.
6. On the **General** tab of the Property Editor, change the **Name** to `VerifyUser`.
7. From the **Operation** menu, select **IsUserValid**.
8. On the **Data Mapping** tab, locate the **loginId** data element, select the corresponding cell in the **Source elements** column, and then click the down arrow.
9. In the **Select a Source** tool that opens, under **GenericFaultOWF**, **Inputs**, **EventNoticeWithReply**, **Extension**; select **Title**; and then click **OK**.
10. In the **Step Palette**, drag a **Decision** step onto the orchestration workflow editor, and drop it onto the line inside the top section of the **IsUserValidScope** step, to the right of the **VerifyUser** step.

    In steps 11 through 24, you configure this step to decide between two possible outcomes: the user is valid or the user is not valid.
11. In the Property Editor, change the **Name** to `IsUserValid`, and then press the Tab key.
12. Right-click the **IsUserValid** step, and then select **Insert New Branch**.
13. On the **General** tab of the Property Editor, change the **Name** to `Yes`.
14. On the **Options** tab, in the **Rule** section, enter the following expression using the expression editor: `VerifyUser.IsUserValidResponse.return`.

    See About the Expression Editor [page 442].
15. In the **Step Palette**, drag a **Calculate** step onto the orchestration workflow editor, and drop it onto the **Yes** branch.

16. On the **General** tab of the Property Editor, change the **Name** to `CreateValidUserMessage`.
17. On the **Options** tab, in the **Target** section, enter the following using the expression editor: `Message`.
18. In the **Expression** section, type: `"This user is valid."`

    Be sure to include the quotation marks.
19. Select the **Otherwise** branch.
20. On the **General** tab of the Property Editor, change the **Name** to `No`, and then press the Tab key.
21. In the **Step Palette**, drag a **Calculate** step onto the orchestration workflow editor, and drop it onto the **No** branch.
22. On the **General** tab of the Property Editor, change the **Name** to `CreateInvalidUserMessage`.
23. On the **Options** tab, in the **Target** section, enter the following using the expression editor: `Message`
24. In the **Expression** section, type `"This user is not valid."`

    Be sure to include the quotation marks.
25. Select the **End** step.
26. On the **Data Mapping** tab, under **Extension**, locate the **Description** data element, select the corresponding **Source elements** column, and then click the down arrow.
27. In the **Select a Source** tool that opens, under **GenericFaultOWF**, **WorkingData**; select **Message**; and then click **OK**.
28. Expand the **FaultHandler** section of the **IsUserValidScope** step.
29. In the **Step Palette**, drag a **Calculate** step into the **FaultHandler** section and drop it onto the **CatchAll** branch.
30. On the **General** tab of the Property Editor, change the **Name** to `CreateUnknownErrorMessage`.
31. On the **Options** tab, in the **Target** section, enter the following using the expression editor: `Message`.
32. In the **Expression** section, type `"An unknown error occurred at GenericFaultAWF_GenericFaultOWF_VerifyUser."`

    Be sure to include the quotation marks.
33. On the Quick Access Toolbar, click the **Validate** button.
34. In the message box that opens, click **OK**.

    The following two warning messages appear in the Message List:

    - **The required DefaultElement 'GenericFaultOWF\Message' is not mapped or defaulted in 'GenericFaultOWF'**

This message warns you that you did not provide a value for the **Message** working data element. You can ignore the message, or you can set the default value to 0 (zero) to prevent the message from appearing.

- **Compensation handler is empty.**

  You can ignore this message, because a compensation handler is not required for this orchestration workflow.

35. *Publish* [page 692] and deploy the **FaultHandlingProcApp.**

    See Step 6: Publish the Process App [page 570] and Step 7: Deploy the Process App [page 570] for instructions.

36. Turn on debug logging as follows:

    a. On the **Home** tab of the Ribbon, in the **Common Views** group, select the **Log Viewer** check box.

    b. On the **Overview** tab of the Log Viewer, right-click **FaultHandlingApp**, and then select **Debug Logging**.

    c. Right-click **FaultHandlingOrch**, and then select **Debug Logging**.

Tutorial: Creating An Empty Synchronous Orchestration Workflow for the Scope Step to Handle Named Faults

In this exercise, you add an empty synchronous *orchestration workflow* [page 690] to the FaultHandlingOrch in the FaultHandlingProcApp.

To create an empty orchestration workflow that handles named Web service faults:

1. In App Explorer, under **Application Workflows**, click **NamedFaultAWF.**
2. In the application workflow editor, right-click the **GetTickerSymbol** transition, and then select **Show Actions**.
3. On the **Actions** tab of the Property Editor, click **New.**

   The **Action Wizard** opens and asks, "Which type of action do you want to execute?"

   Under **Step 1**, **Orchestration Workflow** should be selected.
4. Under **Step 2**, click the **and continue executing** link, and then select **and wait for reply**.
5. Click **Next**.

   The **Action Wizard** asks, "When do you want to call the orchestration workflow?"
6. Under **Step 1**, select **After**.

   Step 2 should read **Invoke an orchestration workflow and wait for reply, after this transition occurs**.
7. Click **Next**.

   The **Action Wizard** asks, "Which orchestration workflow do you want to call?"
8. Under **Step 1**, select **(Add new workflow...)**.

The **Event With Reply** dialog box opens.

9. In the **Name** box, type `NamedFaultWFWR1`.
10. In the **Workflow** box, type `NamedFaultOWF1`.
11. In the **Fields used by event** list, select the **Title** check box.
12. In the **Fields returned by event** list, select the **Description** and **Title** check boxes.
13. Click **OK**.

    Under **Step 1**, **NamedFaultWFWR1** is selected on the menu and **EventNoticeWithReply** appears in the list. **Step 2** now reads **Invoke an orchestration workflow and wait for reply, after this transition occurs, Call NamedFaultWFWR1.**
14. Click **Finish**.

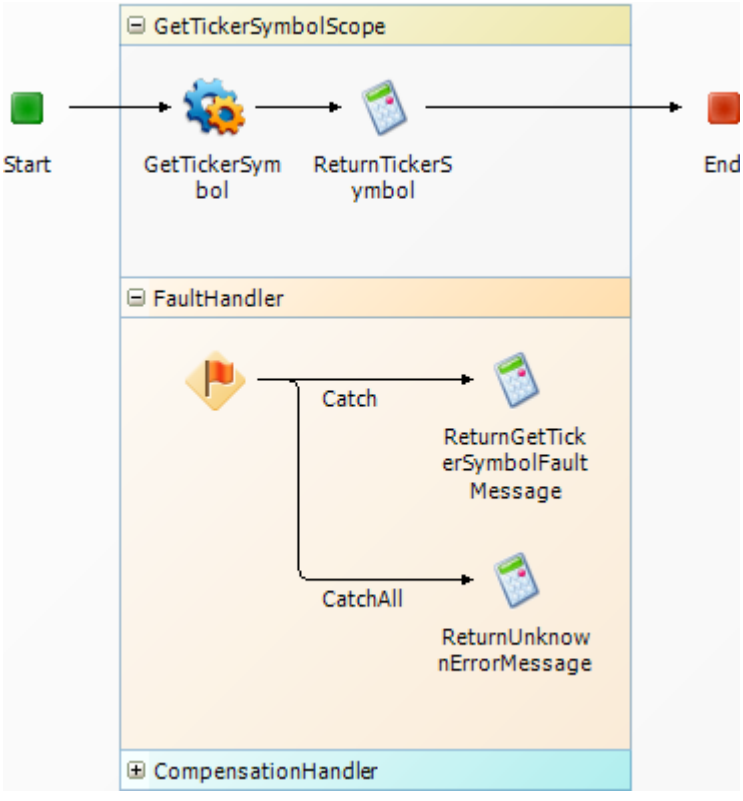
    NamedFaultWFWR1 appears below **GenericFaultWFWR** under **Orchestration Links**, and **NamedFaultOWF1** appears below **GenericFaultOWF** under **Orchestration Workflows** in App Explorer.

Tutorial: Practicing With the Scope Step to Handle Named Web Service Faults

In this exercise, you use a **Scope** step in the **NamedFaultOWF1** orchestration workflow.

After you complete the steps in this exercise, your *orchestration workflow* [page 690] should look like the one in the following figure:

Figure 1. NamedFaultOWF1



To use the Scope step in an orchestration workflow that handles named Web service faults:

1. Under **Orchestration Workflows**, click **NamedFaultOWF1**.
2. Create a working data element of type String to hold the message that is passed to the **Description** field as follows:
   a. Click a blank area of the orchestration workflow editor.
   b. On the **Data Mapping** tab of the Property Editor, under **NamedFaultOWF1**, right-click **Working Data**, and then select **Add New DataElement**.
   c. Change the name of the new **DataElement** to `Message`.
3. In the **New Items** section of the **Step Palette**, drag a **Scope** step onto the orchestration workflow editor, and drop it between the **Start** and **End** steps.
4. Change the name of the **Scope** step to `GetTickerSymbolScope`, and then press the Tab key.
5. In the **Configured Items** section of the **Step Palette**, drag a **SerenaSampleTickerService** Service step onto the orchestration workflow editor, and drop it onto the line inside the top section of the **GetTickerSymbolScope** step.
6. On the **General** tab of the Property Editor, change the **Name** to `GetTickerSymbol`.

7. From the **Operation** menu, select **GetTickerSymbol**.
8. On the **Data Mapping** tab, locate the **company** data element, select the corresponding cell in the **Source elements** column, and then click the down arrow.
9. In the **Select a Source** tool that opens, under **NamedFaultOWF1**, **Inputs**, **EventNoticeWithReply**, **Extension**; select **Title**; and then click **OK**.
10. In the **New Items** section of the **Step Palette**, drag a **Calculate** step onto the orchestration workflow editor, and drop it onto the line inside the top section of the **GetTickerSymbolScope** step, to the right of the **GetTickerSymbol** step.
11. On the **General** tab of the Property Editor, change the **Name** to `ReturnTickerSymbol`.
12. On the **Options** tab, in the **Target** section, enter the following using the expression editor: `EventNoticeWithReply.Extension.Title`.

    See About the Expression Editor [page 442].
13. In the **Expression** section, enter the following expression using the expression editor: `GetTickerSymbol.GetTickerSymbolResponse.GetTickerSymbolResult`.
14. Select the **End** step.
15. On the **Data Mapping** tab, under **Extension**, locate the **Title** data element, select the corresponding cell in the **Source elements** column, and then click the down arrow.
16. In the **Select a source** tool that opens, under **NamedFaultOWF1**, **Inputs**, **EventNoticeWithReply**, **Extension**; select **Title**; and then click **OK**.
17. On the **Data Mapping** tab, under **Extension**, locate the **Message** data element, select the corresponding cell in the **Source elements** column, and then click the down arrow.
18. In the **Select a source** tool that opens, under **NamedFaultOWF1**, **WorkingData**; select **Message**; and then click **OK**.
19. Expand the **FaultHandler** section of the **GetTickerSymbolScope** step.
20. Right-click the **Throw** step, and then select **Insert New Catch**.
21. On the **General** tab of the **Catch** branch, select **SerenaSampleTickerService-GetTickerSymbolFault** on the **Fault Name** menu.

    SerenaSampleTickerService-GetTickerSymbolFault is automatically inserted in the **Fault message** box and is read-only.
22. In the **New Items** section of the **Step Palette**, drag a **Calculate** step into the **FaultHandler** section, and drop it onto the **Catch** branch.
23. On the **General** tab of the Property Editor, change the **Name** to `ReturnGetTickerSymbolFault`.
24. On the **Options** tab, in the **Target** section, enter the following using the expression editor: `Message`.

25. In the **Expression** section, enter the following expression using the expression editor: `Catch.GetTickerSymbolFault.detail`.

    Note: For this step, you must first type `Catch` followed by a period. Then you can use the expression editor to complete the expression. Also, any time you rename a **Catch** branch, you must use the new name in the expression.

26. In the **New Items** section of the **Step Palette**, drag a **Calculate** step into the **FaultHandler** section of the **GetTickerSymbolScope** step, and drop it onto the **CatchAll** branch.

27. On the **General** tab of the Property Editor, change the **Name** to `CreateUnknownErrorMessage`.

28. On the **Options** tab, in the **Target** section, enter the following using the expression editor: `Message`.

29. In the **Expression** section, type `"An unknown error occurred at NamedFaultAWF_NamedFaultOWF1_GetTickerSymbol."`

    Be sure to include the quotation marks.

30. On the Quick Access Toolbar, click the **Validate** button.

31. In the message box that opens, click **OK**.

    The following two warning messages appear in the Message List:

    - **The required DefaultElement 'NamedFaultOWF1\Message' is not mapped or defaulted in 'NamedFaultOWF1'**

      This message warns you that you did not provide a value for the **Message** working data element. You can ignore the message, or you can set the default value to 0 (zero) to prevent the message from appearing.

    - **Compensation handler is empty.**

      You can ignore this message, because a compensation handler is not required for this orchestration workflow.

32. *Publish* [page 692] and deploy the **FaultHandlingProcApp.**

    See Step 6: Publish the Process App [page 570] and Step 7: Deploy the Process App [page 570] for instructions.

Tutorial: Creating an Empty Synchronous Orchestration Workflow for Automatically Adding Catch Branches for Named Faults

In this exercise, you add an empty synchronous *orchestration workflow* [page 690] to the **FaultHandlingOrch** in the **FaultHandlingProcApp**.

To create an empty orchestration workflow for automatically adding Catch branches for named faults:

1. In App Explorer, under **Application Workflows**, click **NamedFaultAWF.**

2. In the application workflow editor, right-click the **GetBuyRating** transition, and then select **Show Actions** on the menu.

3. On the **Actions** tab of the Property Editor, click **New.**

   The **Action Wizard** opens and asks, "Which type of action do you want to execute?"

   Under **Step 1**, **Orchestration Workflow** should be selected.

4. Under **Step 2**, click the **and continue executing** link, and then select **and wait for reply**.

5. Click **Next**.

   The **Action Wizard** asks, "When do you want to call the orchestration workflow?"

6. Under **Step 1**, select **After**.

   Step 2 should read **Invoke an orchestration workflow and wait for reply, after this transition occurs**.

7. Click **Next**.

   The **Action Wizard** asks, "Which orchestration workflow do you want to call?"

8. Under **Step 1**, select **(Add new workflow...)**. The **Event With Reply** dialog box opens.

9. In the **Name** box, type `NamedFaultWFWR2`.

10. In the **Workflow** box, type `NamedFaultOWF2`.

11. In the **Fields used by event** list, select the **Description** and **Title** check boxes.

12. In the **Fields returned by event** list, select the **Description** and **Title** check boxes.

13. Click **OK**.

    Under **Step 1**, **NamedFaultWFWR2** is selected on the menu and **EventNoticeWithReply** appears in the list. **Step 2** now reads **Invoke an orchestration workflow and wait for reply, after this transition occurs, Call NamedFaultWFWR2.EventNoticeWithReply.**

14. Click **Finish**.

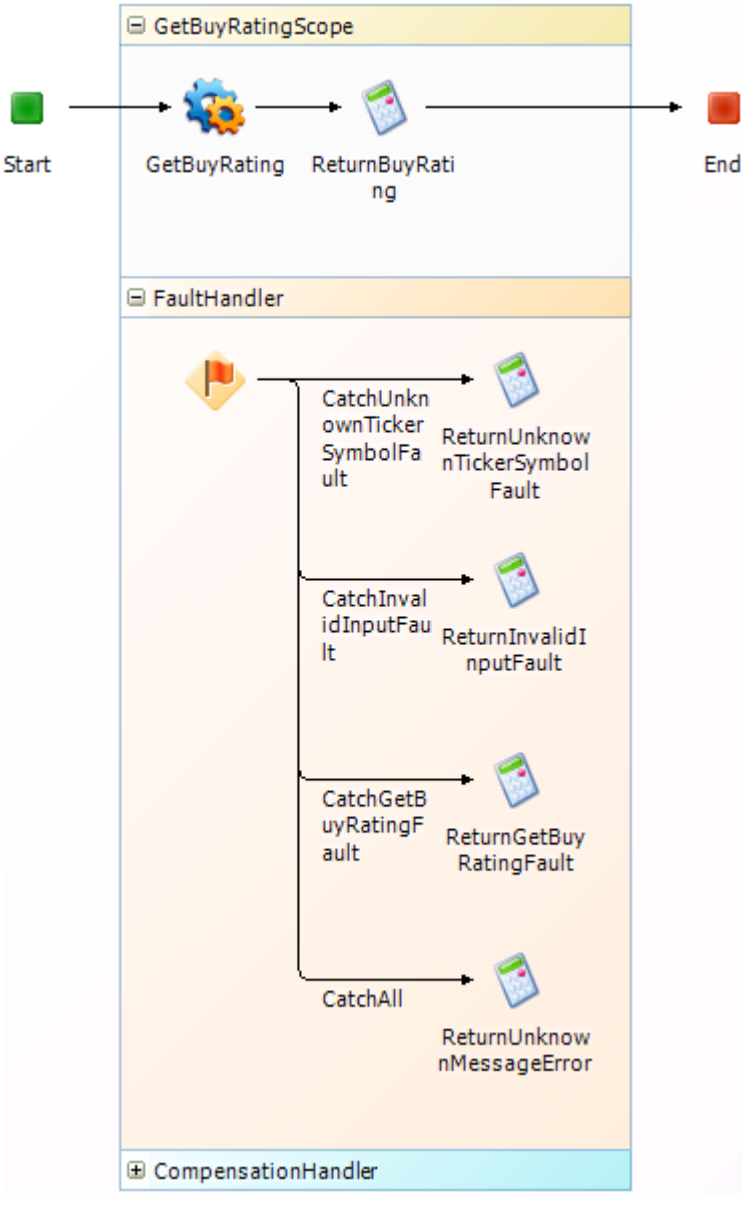
    NamedFaultWFWR2 appears below **NamedFaultWFWR1** under **Orchestration Links**, and **NamedFaultOWF2** appears below **NamedFaultOWF1** under **Orchestration Workflows** in App Explorer.

Tutorial: Practicing Automatically Adding Catch Branches for Named Faults

In this exercise, you use a **Scope** step in the **NamedFaultOWF2** orchestration workflow and you add **Catch** branches to its **FaultHandler** section automatically.

After you complete the steps in this exercise, your *orchestration workflow* [page 690] should look like the one in the following figure:

Figure 1. NamedFaultOWF2



To automatically add Catch branches for named Web service faults:

1. Under **Orchestration Workflows**, click **NamedFaultOWF2**.
2. Create a working data element of type String to hold the message that is passed to the **Description** field as follows:
   a. Click a blank area of the orchestration workflow editor.
   b. On the **Data Mapping** tab of the Property Editor, under **NamedFaultOWF2**, right-click **Working Data**, and then select **Add New DataElement**.

c. Change the name of the new **DataElement** to `Message`.

3. In the **New Items** section of the **Step Palette**, drag a **Scope** step onto the orchestration workflow editor, and drop it between the **Start** and **End** steps.
4. Change the name of the **Scope** step to `GetBuyRatingScope`, and then press the Tab key.
5. In the **Configured Items** section of the **Step Palette**, drag a **SerenaSampleTickerService** Service step onto the orchestration workflow editor, and drop it onto the line inside the top section of the **GetBuyRatingScope** step.
6. On the **General** tab of the Property Editor, change the **Name** to `GetBuyRating`.
7. From the **Operation** menu, select **GetBuyRating**.
8. On the **Data Mapping** tab, locate the **symbol** data element, select the corresponding cell in the **Source elements** column, and then click the down arrow.
9. In the **Select a Source** tool that opens, under **NamedFaultOWF2**, **Inputs**, **EventNoticeWithReply**, **Extension**; select **Title**; and then click **OK**.
10. In the **New Items** section of the **Step Palette**, drag a **Calculate** step onto the orchestration workflow editor, and drop it onto the line inside the top section of the **GetBuyRatingScope** step, to the right of the **GetBuyRating** step.
11. On the **General** tab of the Property Editor, change the **Name** to `ReturnBuyRating`.
12. On the **Options** tab, in the **Target** section, enter the following using the expression editor: `Message`.

    See About the Expression Editor [page 442].
13. In the **Expression** section, enter the following expression using the expression editor: `GetBuyRating.GetBuyRatingResponse.GetBuyRatingResult`.
14. Select the **End** step.
15. On the **Data Mapping** tab, under **Extension**, locate the **Description** data element, select the corresponding cell in the **Source elements** column, and then click the down arrow.
16. In the **Select a source** tool that opens, under **NamedFaultOWF2**, **WorkingData**; select **Message**; and then click **OK**.
17. Expand the **FaultHandler** section of the **GetBuyRatingScope** step.
18. In the top section of the **GetBuyRatingScope** step, right-click the **GetBuyRating** step, and then select **Add Catch Branches to GetBuyRatingScope**.

    SBM Composer scans the **SerenaSampleTickerService** WSDL for any named faults defined for the **GetBuyRating** operation and adds the following three **Catch** branches to the **FaultHandler** section: **CatchUnknownTickerSymbolFault**, **CatchInvalidInputFault**, and **CatchGetBuyRatingFault**.

    Note: If the Web service operation does not return any named faults, the **Add Catch Branches to Scope** option is not available.

19. Select each **Catch** branch, and note on the **General** tab of the Property Editor that the **Fault name** and **Fault message** are automatically configured.
20. In the **New Items** section of the **Step Palette**, drag a **Calculate** step into the **FaultHandler** section, and drop it onto the **CatchUnknownTickerSymbolFault** branch.
21. On the **General** tab of the Property Editor, change the **Name** to `ReturnUnknownTickerSymbolFault`.
22. On the **Options** tab, in the **Target** section, enter the following using the expression editor: `Message`
23. In the **Expression** section, enter the following expression using the expression editor: `CatchUnknownTickerSymbolFault.UnknownTickerSymbolFault.detail`

    You can use `errorCode` rather than `detail` if you specified a decision based on the error code.

    Note: For this step, you must first type `CatchUnknownTickerSymbolFault` followed by a period. Then you can use the expression editor to complete the expression. Also, any time you rename a **Catch** branch, you must use the new name in the expression.
24. In the **New Items** section of the **Step Palette**, drag a **Calculate** step into the **FaultHandler** section, and drop it onto the **CatchInvalidInputFault** branch.
25. On the **General** tab of the Property Editor, change the **Name** to `ReturnInvalidInputFault`.
26. On the **Options** tab, in the **Target** section, enter the following using the expression editor: `Message`.
27. In the **Expression** section, enter the following expression using the expression editor: `CatchInvalidInputFault.InvalidInputFault`.

    Note: For this step, you must first type `CatchInvalidInputFault` followed by a period. Then you can use the expression editor to complete the expression.
28. In the **New Items** section of the **Step Palette**, drag a **Calculate** step into the **FaultHandler** section, and drop it onto the **CatchGetBuyRatingFault** branch.
29. On the **General** tab of the Property Editor, change the **Name** to `ReturnGetBuyRatingFault`.
30. On the **Options** tab, in the **Target** section, enter the following using the expression editor: `Message`.
31. In the **Expression** section, enter the following expression using the expression editor: `CatchGetBuyRatingFault.GetBuyRatingFault.detail`.

    You can use `errorCode` rather than `detail` if you created a system that correlates the message detail with the error code.

    Note: For this step, you must first type `CatchGetBuyRatingFault` followed by a period. Then you can use the expression editor to complete the expression. Also, any time you rename a **Catch** branch, you must use the new name in the expression.

32. In the **New Items** section of the **Step Palette**, drag a **Calculate** step into the **FaultHandler** section of the **GetBuyRatingScope** step, and drop it onto the **CatchAll** branch.
33. On the **General** tab of the **Property Editor**, change the **Name** to `CreateUnknownErrorMessage`.
34. On the **Options** tab, in the **Target** section, enter the following using the expression editor: `Message`.
35. In the **Expression** section, type `"An unknown error occurred at NamedFaultAWF_NamedFaultOWF2_GetBuyRating"`.

    Be sure to include the quotation marks.
36. On the Quick Access Toolbar, click the **Validate** button.
37. In the message box that opens, click **OK**.

    The following two warning messages appear in the Message List:

    - **The required DefaultElement 'NamedFaultOWF2\Message' is not mapped or defaulted in 'NamedFaultOWF2'**

      This message warns you that you did not provide a value for the **Message** working data element. You can ignore the message, or you can set the default value to 0 (zero) to prevent the message from appearing.

    - **Compensation handler is empty**

      You can ignore this message, because a compensation handler is not required for this orchestration workflow.

    - **The required DataElement 'GetBuyRating\GetBuyRating\type' is not mapped or defaulted in 'NamedFaultOWF2**

      You can ignore this message.
38. *Publish* [page 692] and deploy the **FaultHandlingProcApp.**

    See Step 6: Publish the Process App [page 570] and Step 7: Deploy the Process App [page 570] for instructions.

Rules for Configuring the Catch Branch

When you configure a **Catch** branch, you specify a fault name, a fault message, or both. Before you can determine which item or items to choose, you need to know how the Web service that you are invoking returns faults. This information is usually provided in the Web service documentation.

In SBM Composer, the rules are applied as follows:

- If the Web service returns a fault with no associated message, after you select a **Fault name**, the **Fault message** box is empty.
- If the Web service returns a fault with a name and an associated message, after you select a **Fault name**, the name of the associated message is displayed in the **Fault message** box. The message is read-only.

Using the Throw Step

You use the **Throw** step to return a fault named ServiceFlowFault in an *orchestration workflow* [page 690].

Procedure Summary

1. In App Explorer, select an orchestration workflow to display it in the orchestration workflow editor.
2. In the **New Items** section of the **Step Palette**, drag a **Throw** step onto the orchestration workflow editor, and drop it wherever you want to throw a ServiceFlowFault.
3. On the **General** tab of the Property Editor, you can change the name of the **Throw** step and provide a description.
4. On the **Data Mapping** tab, you can enter an optional fault code in the **Default** column of the **FaultCode** data element.

   You can use the fault code as a short way to represent a long fault string.
5. Still on the **Data Mapping** tab, you can provide a value for the **FaultString** data element. You can map a value in the **Source elements** column, or you can enter a String value, such as a message, in the **Default value** column.

Tutorial: Creating an Empty Synchronous Orchestration Workflow for the Throw Step

In this exercise, you add an empty synchronous *orchestration workflow* [page 690] to the **FaultHandlingOrch** in the **FaultHandlingProcApp**.

To create an empty synchronous orchestration workflow that uses the Throw step:

1. In App Explorer, under **Application Workflows**, click **ThrowAWF**.
2. In the application workflow editor, right-click the **GetTickerSymbol** transition, and then select **Show Actions**.
3. On the **Actions** tab of the Property Editor, click **New**.

   The **Action Wizard** opens and asks, "Which type of action do you want to execute?"

   Under **Step 1**, **Orchestration Workflow** should be selected.
4. Under **Step 2**, click the **and continue executing** link, and then select **and wait for reply**.
5. Click **Next**.

   The **Action Wizard** asks, "When do you want to call the orchestration workflow?"
6. Under **Step 1**, select **After**.

   Step 2 should read **Invoke an orchestration workflow and wait for reply, after this transition occurs**.
7. Click **Next**.

   The **Action Wizard** asks, "Which orchestration workflow do you want to call?"

8. Under **Step 1**, select **(Add new workflow...)**.

   The **Event With Reply** dialog box opens.

9. In the **Name** box, type `ThrowWFWR`.
10. In the **Workflow** box, type `ThrowOWF`.
11. In the **Fields used by event** list, select the **Title** check box.
12. In the **Fields used by event** list, select the **Description** and **Title** check boxes.
13. Click **OK**.

    Under **Step 1**, **ThrowWFWR** is selected on the menu and **EventNoticeWithReply** is selected in the list. **Step 2** now reads **Invoke an orchestration workflow and wait for reply, after this transition occurs, Call ThrowWFWR**.

14. Click **Finish**.

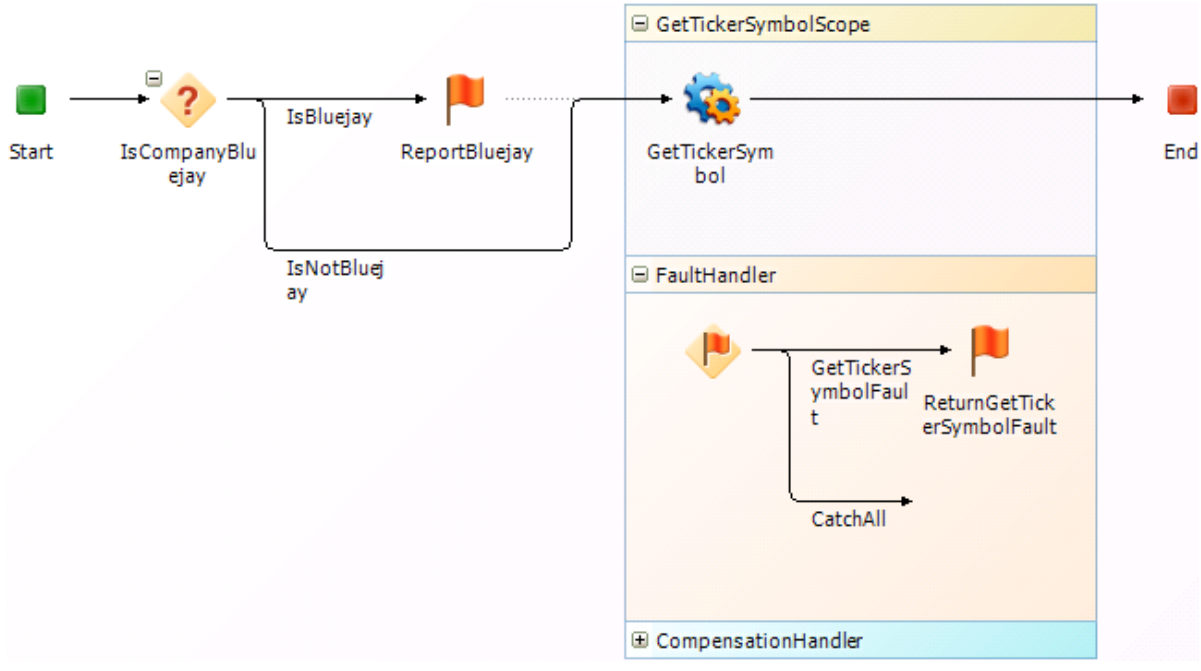
    ThrowWFWR appears below **NamedFault2WRWR** under **Orchestration Links**, and **ThrowOWF** appears below **NameFault2OWF** under **Orchestration Workflows** in App Explorer.

Tutorial: Practicing With the Throw Step

In this exercise, you use a **Scope** step in the **NamedFaultOWF2** *orchestration workflow* [page 690] and add **Catch** branches to its **FaultHandler** section automatically.

After you complete the steps in this exercise, your orchestration workflow should look like the one in the following figure:

Figure 1. ThrowOWF



To use the Throw step in an orchestration workflow that throw custom and named faults:

1. In App Explorer, under **Orchestration Workflows**, click **ThrowOWF**.
2. In the **New Items** section of the **Step Palette**, drag a **Decision** step onto the orchestration workflow editor, and drop it onto the line between the **Start** and **End** steps.

   In steps 3 through 10, you configure this step to decide between two possible outcomes: the company is Bluejay Bird Supply or the company is not Bluejay Bird Supply.
3. On the **General** tab of the Property Editor, change the **Name** to `IsCompanyBluejay`.
4. Right-click the **IsCompanyBluejay** step, and then select **Insert New Branch**.
5. On the **General** tab of the Property Editor, change the **Name** to `IsBluejay`.
6. On the **Options** tab of the Property Editor, enter the following expression in the **Rule** section using the expression editor:
   ```
   EventNoticeWithReply.Extension.Title="Bluejay Bird Supply"
   ```
   See About the Expression Editor [page 442].

7. In the **Step Palette**, drag a **Throw** step onto the orchestration workflow editor and drop it on the **IsBluejay** branch.

   This step is used to display an error message in the SBM User Workspace when the **Title** field contains "Bluejay Bird Supply." It is also used to display an optional error code that is not displayed in the SBM User Workspace, but can be used, for example, for debugging purposes.

8. On the **General** tab of the Property Editor, change the **Name** to `ReportBluejay`, and then press the Tab key.

9. On the **Data Mapping** tab, locate the **FaultCode** data element and enter the following in the corresponding cell of the **Default value** column: `ERROR-BLUEJAY`.

10. Locate the **FaultString** data element, and enter the following in the corresponding **Default value** column: `Bluejay Bird Supply is a credit risk.`

11. In the **New Items** section of the **Step Palette**, drag a **Scope** step onto the orchestration workflow editor, and drop it between the **ReportBluejay** and **End** steps.

    This step is used to enclose the **GetTickerSymbol** Web service operation and to catch any faults that it generates.

12. On the **General** tab of the Property Editor, change the **Name** to `GetTickerSymbolScope`.

13. In the **Configured Items** section of the **Step Palette**, drag a **SerenaSampleTickerService** step into the top section of the **GetTickerSymbolScope** step, and drop it. This step is used to get the ticker symbol for the company name passed from the **Title** field, or to return a fault for all other company names except Bluejay Bird Supply (see step 4).

14. On the **General** tab of the Property Editor, change the **Name** to `GetTickerSymbol`.

15. On the **Operation** menu, select **GetTickerSymbol**.

16. On the **Data Mapping** tab, locate the **company** data element, select the corresponding cell in the **Source elements** column, and then click the down arrow.

17. In the **Select a Source** tool that opens, under **ThrowOWF**, **Inputs**, **EventNoticeWithReply**, **Extension**; select **Title**; and then click **OK**.

18. Expand the **FaultHandler** section.

19. Right-click the **Fault Handler** step, and then select **Insert New Catch**.

    This **Catch** branch handles the SerenaSampleTickerService-GetTickerSymbolFault. All other faults generated by the **GetTickerSymbol** step are handled by the **CatchAll** branch.

20. On the **General** tab of the Property Editor for the **Catch** branch, change the **Name** to `GetTickerSymbolFault`.

21. Select **SerenaSampleTickerService-GetTickerSymbolFault** on the **Fault Name** menu.

    SerenaSampleTickerService-GetTickerSymbolFault is automatically inserted in the **Fault message** box and is read-only.

22. In the **Step Palette**, drag a **Throw** step into the **FaultHandler** section and drop it onto the **GetTickerSymbolFault** branch.

    This step is used to return a customized fault message in the SBM User Workspace. It is also used to generate an optional error code.

23. On the **General** tab of the Property Editor, change the **Name** to `ReturnGetTickerSymbolFault`.

24. On the **Data Mapping** tab, locate the **FaultCode** data element and enter the following in the corresponding cell in the **Default value** column: `ERROR-SYSTEM`.

25. Locate the **FaultString** data element, select the corresponding cell in the **Source elements** column, and then click the down arrow.

26. In the **Select a Source** tool that opens, under **ThrowOWF**, **GetTickerSymbolFault**, **Outputs**, **GetTickerSymbolFault**; select **detail**; and then click **OK**.

27. Select the **End** step.

    The contents of the **Title** field are passed to the **End** step. This step is used to display a valid ticker symbol in the **Description** field in the SBM User Workspace. Although Bluejay Bird Supply is listed in the SerenaSampleTickerService, its symbol is not returned because it was caught by the **ReportBluejay** Throw step.

28. On the **Data Mapping** tab, under **Extension**, locate the **Description** data element, select the corresponding cell in the **Source elements** column, and then click the down arrow.

29. In the **Select a Source** tool that opens, under **GetTickerSymbol**, **Outputs**, **GetTickerSymbolResponse**; select **GetTickerSymbolResult**; and then click **OK**.

30. On the Quick Access Toolbar, click the **Validate** button.

31. In the message box that opens, click **OK**.

    The following a warning message appears in the Message List: **Compensation handler is empty.**

    You can ignore this message, because a compensation handler is not required for this orchestration workflow.

32. *Publish* [page 692] and deploy the **FaultHandlingProcApp.**

    See Step 6: Publish the Process App [page 570] and Step 7: Deploy the Process App [page 570] for instructions.

Using the Compensate Step

You use the **Compensate** step to reverse, or undo, the operations in an enclosed scope that already completed.

Procedure Summary

1. In App Explorer, select an *orchestration workflow* [page 690] that with nested **Scope** steps.

2. If necessary, expand the outermost **Scope** step so that all sections are visible.

3. In the **Step Palette**, drag a **Compensate** step onto the orchestration workflow editor and drop it in the **FaultHandler** or **CompensationHandler** section of the outermost **Scope** step.

   If you want to compensate for a specific named fault only, place the **Compensate** step in a **FaultHandler** section with a named fault. Otherwise, place the **Compensate** step in the **CompensationHandler** section.

4. In the Property Editor of the **Compensate** step, select the **Scope** to be compensated. You can also enter a description.

5. To specify the compensation actions, add other steps from the **Step Palette** to the **CompensationHandler** section, to the right of the **Compensate** step.

6. If you are using nested scopes, you can use a **Throw** step to pass a fault from any inner scopes to an outer scope.

Tutorial: Creating an Empty Asynchronous Orchestration Workflow for the Compensate Step

You use the **Compensate** step in combination with nested **Scope** steps. The *Web services* [page 701] or actions defined in the **Compensate** step should compensate or roll back whatever was done in the main part of the inner nested **Scope** step. This is particularly useful if these actions have taken some time to complete.

In this exercise, you add an empty asynchronous *orchestration workflow* [page 690] to the **FaultHandlingOrch** in the **FaultHandlingProcApp**.

To create an empty asynchronous orchestration workflow that uses the Compensate step:

1. In App Explorer, under **Application Workflows**, click **CompensateAWF**.
2. In the application workflow editor, right-click the **DemoCompensate** transition, and then select **Show Actions**.
3. On the **Actions** tab of the Property Editor, click **New**.

   The **Action Wizard** opens and asks, "Which type of action do you want to execute?"

   Under **Step 1**, **Orchestration Workflow** should be selected and **Step 2** should read **Invoke and orchestration workflow and continue executing**.

4. Click **Next**.

   The **Action Wizard** asks, "What do you want to affect?"

   Under **Step 1**, **This item** should be selected and **Step 2** should read **Invoke an orchestration workflow and continue executing, affect this item**.

5. Click **Next**.

   The **Action Wizard** asks, "Which condition do you want to check?"

   Under **Step 1**, **Unconditionally** should be selected and **Step 2** should still read **Invoke an orchestration workflow and continue executing, affect this item**.

6. Click **Next**.

   The **Action Wizard** asks, "Which orchestration workflow to you want to invoke?"

7. Under **Step 1**, select **(Add new workflow...)**.

   NewDemoCompensateWorkflow appears under **Step 1** and should be selected. **Step 2** should read **Invoke an orchestration workflow and continue executing, affect this item, invoke event (FaultHandlingPTable:CompensateAWF_New_DemoCompensate)**.

   SubmitGetTickerSymbolWorkflow appears below **ThrowSyncOWF** under **Orchestration Workflows** in App Explorer.

8. Click **Finish**.

9. Click **NewDemoCompensateWorkflow**, and then, on the **General** tab of the Property Editor, change the **Name** to `CompensateOWF`.

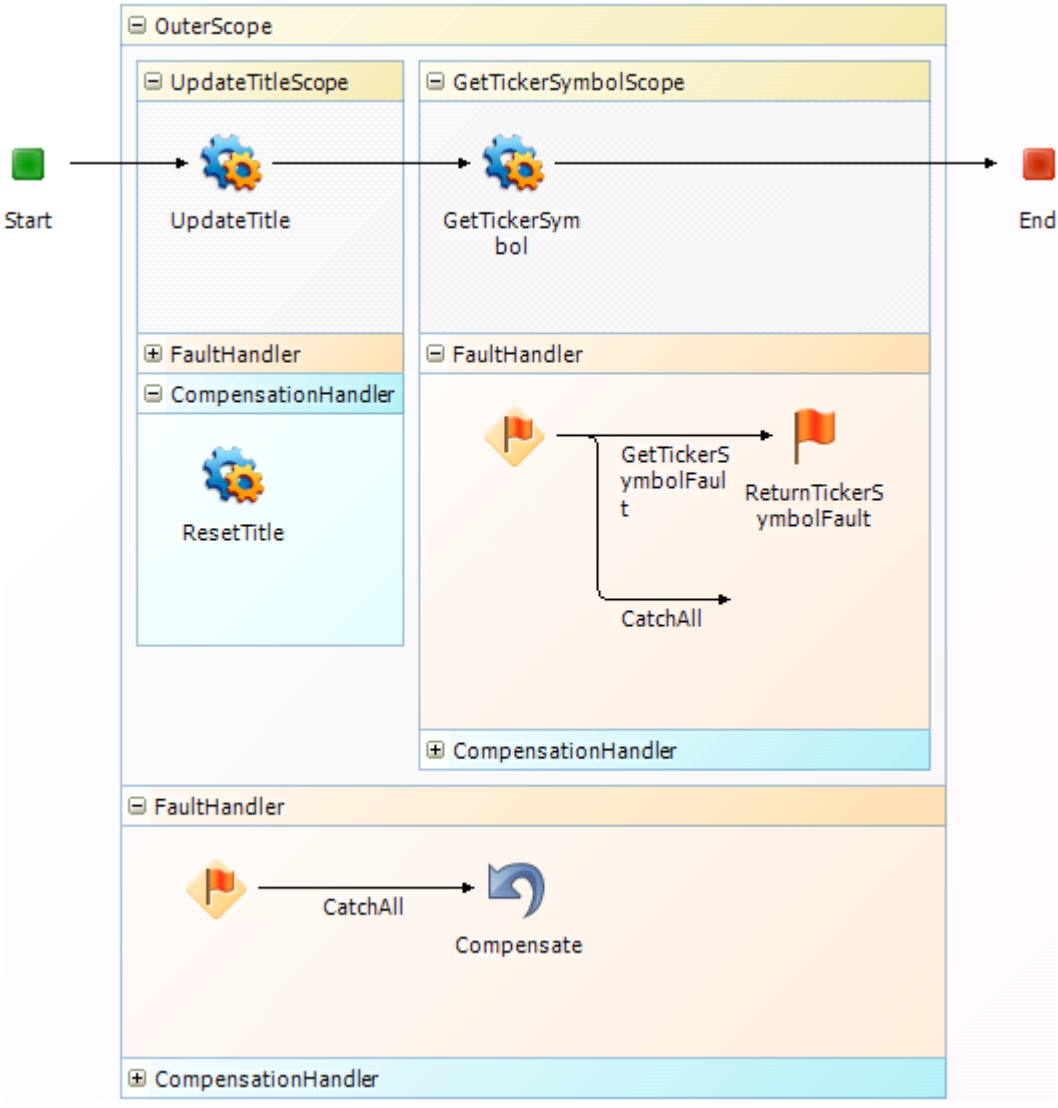
Tutorial: Practicing with the Compensate Step

In this exercise, you use a **Compensate** step in the **CompensateOWF** orchestration workflow.

Note: For this tutorial, the purpose of the SerenaSampleTickerService Web service is to return a named fault, not to get the results of any of its operations.

After you complete the steps in this exercise, your *orchestration workflow* [page 690] should look like the one in the following figure:

Figure 1. CompensateOWF



To use the Compensate step in an asyncronous orchestration workflow:

1. In App Explorer, select **CompensateOWF**.
2. In the **New Items** section of the **Step Palette**, drag a **Scope** step onto the orchestration workflow editor, and drop it onto the line between the **Start** and **End** steps.
3. On the **General** tab of the Property Editor, change the **Name** to `OuterScope`, and then press the Tab key.
4. In the **New Items** section of the **Step Palette**, drag a **Scope** step onto the orchestration workflow editor, and drop it inside the top section of the **OuterScope** step.

5. On the **General** tab of the Property Editor, change the **Name** to `UpdateTitleScope`.
6. In the **Configured Items** section of the **Step Palette**, drag an **sbmappservices72** Service step into the **UpdateTitleScope** step, and drop in the top section.
7. On the **General** tab of the Property Editor, change the **Name** to `UpdateTitle`.
8. From the **Operation** menu, select **TransitionItem.**
9. On the **Data Mapping** tab, expand **item**, **id**; locate the **tableIdItemId** data element; select the corresponding cell in the **Source elements** column; and then click the down arrow.
10. In the **Select a source** tool that opens, under **CompensateOWF**, **Inputs**, **EventNotice**, **Extension**; select **ItemId_TableRecId**; and then click **OK**.
11. Locate the **title** data element, and enter the following in the corresponding cell in the **Default value** column: `Updated by CompensationOWF.`
12. Expand the **CompensationHandler** of the **UpdateTitleScope** step.
13. In the **Configured Items** section of the **Step Palette**, drag an **sbmappservices72** Service step into the **CompensationHandler** of the **UpdateTitleScope** step.
14. On the **General** tab of the Property Editor, change the **Name** to `ResetTitle`.
15. From the **Operations** menu, select **TransitionItem**.
16. On the **Data Mapping** tab, expand **item**, expand **id**; locate the **tableIdItemid** data element; select the corresponding cell in the **Source elements** column; and then click the down arrow.
17. In the **Select a source** tool that opens, under **CompensateOWF**, **Inputs**, **EventNotice**, **Extension**; select **ItemId_TableRecId**; and then click **OK**.
18. Locate the **description** data element, and enter the following in the corresponding **Default value** column: `Returning Title field to its original value.`
19. Collapse the **UpdateTitleScope** step.
20. In the **New Items** section of the **Step Palette**, drag a **Scope** step into the top section of the **OuterScope** step, and drop it on the right of the **UpdateTitleScope** step.
21. On the **General** tab of the Property Editor, change the **Name** to `GetTickerSymbolScope.`
22. In the **Configured Items** section of the **Step Palette**, drag a **SerenaSampleTickerService** Service step into the top section of the **GetTickerSymbolScope** step.
23. On the **General** tab of the Property Editor, change the **Name** to `GetTickerSymbol`.
24. On the **Operation** menu, select **GetTickerSymbol**.
25. On the **Data Mapping** tab, locate the **company** data element; select the corresponding cell in the **Source elements** column; and then click the down arrow.

26. In the **Select a source** tool that opens, under **CompensateOWF**, **Inputs**, **EventNotice**, **Extension**; select **Title**; and then click **OK**.
27. Expand the **FaultHandler** section of the **GetTickerSymbolScope** step.
28. Right-click the **Fault Handler** step, and then select **Insert New Catch**.
29. Select **SerenaSampleTickerService-GetTickerSymbolFault** from the **Fault name** list.
30. On the **General** tab of the Property Editor, change the **Name** to `GetTickerSymbolFault`.

    SerenaSampleTickerService-GetTickerSymbolFault automatically appears in the **Fault message** box.
31. In the **New Items** section of the **Step Palette**, drag a **Throw** step into the **FaultHandler** section of the **GetTickerSymbolScope** step, and drop it onto the **GetTickerSymbolFault** branch.
32. On the **Data Mapping** tab, locate the **FaultString** data element, select the corresponding cell in the **Source elements** column; and then click the down arrow.
33. In the **Select a source** tool that opens, under **GetTickerSymbolFault**, **Outputs**, **GetTickerSymbolFault**; select **detail**; and then click **OK**.
34. Expand the **FaultHandler** section of the **OuterScope** step.
35. In the **New Items** section of the **Step Palette**, drag a **Compensate** step into the **FaultHandler** section, and drop it onto the **CatchAll** branch.

    On the **General** tab of the Property Editor, the **Scope** should be **[Default Scope]**.
36. On the Quick Access Toolbar, click the **Validate** button.
37. In the message box that opens, click **OK**.

    The following warning message appears in the Message List: **Compensation handler is empty.**

    You can ignore this message, because a compensation handler is not required for this orchestration workflow.
38. *Publish* [page 692] and deploy the **FaultHandlingProcApp.**

    See Step 6: Publish the Process App [page 570] and Step 7: Deploy the Process App [page 570] for instructions.

Running the Fault Handling Process App

This section contains exercises for running the various FaultHandlingProcApp projects (application workflows) in the SBM User Workspace. The steps in each tutorial are correlated with the way elements in the orchestration workflows and the application workflows interact.

Following is a list of the tutorials in this section.

- SerenaSampleTickerService Company Names and Ticker Symbols [page 531]
- Tutorial: Running the GenericFaultAWF Project [page 531]

- Tutorial: Altering the GenericFaultOWF to Return a Web Service Fault [page 532]
- Tutorial: Running the GenericFaultAWF Project and Invoking the CatchAll Branch [page 532]
- Tutorial: Running the NamedFaultAWF Project and Invoking a Catch Branch [page 534]
- Tutorial: Running the ThrowAWF Project [page 535]
- Tutorial: Running the CompensateAWF Project [page 537]

SerenaSampleTickerService Company Names and Ticker Symbols

Several of the tutorials in this section use the SerenaSampleTickerService. The companies listed on this Web service and their ticker symbols are shown in the following table. In some of the steps, you are required to use the values in this table.

Company Name	Ticker Symbol
Aftabitorium	AFT
Bluejay Bird Supply	BBS
Carol Creations	CCR
Meg N Ah Electric Co.	MAC
Rob & Bert Manufacturing	RBM
Tim Buck 2 Entertainment	TBE

Tutorial: Running the GenericFaultAWF Project

In this exercise, you run the GenericFaultAWF Project in the SBM User Workspace.

To run the GenericFaultAWF Project:

1. Log on to the SBM User Workspace.
2. Select the **FaultHandlingApp** tab.

   If the **FaultHandlingApp** tab is not visible, click the **More** tab, and then select **FaultHandlingApp** in the list.
3. If **Submit View** appears in the *navigation pane* [page 689], click **Submit to my Preferred Projects**. Otherwise, click the **Submit View** icon at the bottom of the navigation pane.
4. In the **Submit Tree**, click **GenericFaultAWF Project**.

   The **Submit** transition form opens.
5. In the **Title** field, enter your user ID, and then click **OK**.

   The message "This user is valid" is returned in the **Description** field.

6. In **Submit View**, click **Submit to my Preferred Projects**.
7. In the **Submit Tree**, click **GenericFaultAWF Project**.
8. In the **Title** field of the **Submit** transition form, enter an invalid user ID such as `InvalidUser`, and then click **OK**.

   The message "This user is not valid" is returned in the **Description** field.

Tutorial: Altering the GenericFaultOWF to Return a Web Service Fault

In this exercise, you alter the **GenericFaultOWF** orchestration workflow by mapping an invalid user ID. When the *orchestration workflow* [page 690] is invoked, a Web service fault is generated.

To alter the GenericFaultOWF to return a Web service fault:

1. Open SBM Composer.
2. Click the Composer button, and then click **Open**.

   The **Open Process App** dialog box opens.
3. Select **FaultHandlingProcApp**, and then select **Open**.
4. In App Explorer, click **FaultHandlingProcApp**, and then click the **All Items** filter.
5. Under **Orchestration Workflows**, click **GenericFaultOWF**.
6. In the orchestration workflow editor, select the **VerifyUser** step.
7. On the **Data Mapping** tab of the **Property Editor**, expand the **auth** data element, locate the **userId** data element, and enter an invalid user ID such as `zzzz` in the corresponding **Default value** column.
8. *Publish* [page 692] and deploy the **FaultHandlingProcApp.**

   See Step 6: Publish the Process App [page 570] and Step 7: Deploy the Process App [page 570] for instructions.

Tutorial: Running the GenericFaultAWF Project and Invoking the CatchAll Branch

In this exercise, you run the altered GenericFaultAWF Project in the SBM User Workspace.

To run the GenericFaultAWF Project and return an error message:

1. Log on to the SBM User Workspace.
2. Select the **FaultHandlingPro** tab.

   If the **FaultHandlingPro** tab is not visible, click the **More** tab, and then select **FaultHandlingPro** in the list.
3. If **Submit View** appears in the *navigation pane* [page 689], click **Submit to my Preferred Projects**. Otherwise, click the **Submit View** icon at the bottom of the navigation pane.
4. In the **Submit Tree**, click **GenericFaultAWF Project**.

   The **Submit** transition form opens.

5. Enter some text in the **Title** field, and then click **OK**.

   The message `An unknown error occurred at GenericFaultAWF_GenericFaultOWF_VerifyUser` is returned in the **Description** field.

   Note: The *orchestration workflow* [page 690] will fail when you invoke it, so it does not matter what you enter in the **Title** field. However, you must enter something, because this is a required field.

6. To find out more about the error, you can return to the **FaultHandlingProcessApp** in SBM Composer and look for the associated SOAP fault in the Log Viewer as follows:

   a. If the **Log Viewer** tab is not visible, on the Ribbon, in the **Common Views** group, select the **Log Viewer** check box.

   b. On the **Overview** tab of the Log Viewer, click **Refresh**.

      GenericFaultOWF should be selected.

   c. On the **Details** tab, select the latest run on the **Run** tab.

   d. Look for the error message that begins with `A fault occurred during the execution of the orchestration…`

   e. Right-click anywhere in the message row, and then select **Show Message**.

      The **Message Detail** dialog box opens and shows the content of the SOAP message.

   f. Click the **Previous** button until you see the message that contains the SOAP fault. This message begins as follows: `A Web service was invoked at Service step VerifyUser, and now the Orchestration Engine is receiving the following message.`

   g. In the SOAP message that follows, locate the SOAP fault, which begins with `<SOAP-ENV:Fault.`

      The received SOAP message for the **GenericFaultApp Project** should contain the following information:

```
…
<SOAP-ENV:Fault xmlns:SOAP-ENV='http://schemas.xmlsoap.org/soap/envelope/'>
  <faultcode>SOAP-ENV:Server</faultcode>
  <faultstring>Invalid User ID or Password</faultstring>
  <detail>
    <ae:AEWebservicesFault xmlns:ae='urn:sbmappservices72'>Invalid
      User ID or Password</ae:AEWebservicesFault>
  </detail>
</SOAP-ENV:Fault>
…
```

      Note that the SOAP fault indicates an invalid user ID or password.

   h. Click the **Previous** button again until you see the SOAP message that contains the source of the error. This message begins as follows: `A Web service is being invoked at Service step VerifyUser, and the Orchestration Engine is sending the following message.`

The sent SOAP message for the **GenericFaultApp Project** should contain the following information:

```
…
xmlns:ns8='urn:sbmappservices72'><ns8:auth><ns8:userId>ZZZZZ</ns8:userId>
  </ns8:auth><ns8:loginId
…
```

Note the incorrect entry for **userID**.

7. After you complete this part of the tutorial, you can restore **GenericFaultOWF** to its original, valid state by deleting the invalid **userId** value and redeploying the process app. If you do not do this, you can still run the other projects in the **FaultHandlingProcessApp**.

Tutorial: Running the NamedFaultAWF Project and Invoking a Catch Branch

In this exercise, you run the NamedFaultAWF Project in the SBM User Workspace.

To run the NamedFaultAWF Project:

1. Log on to the SBM User Workspace.
2. Select the **FaultHandlingApp** tab.

   If the **FaultHandlingApp** tab is not visible, click the **More** tab, and then select **FaultHandlingApp** in the list.
3. If **Submit View** appears in the *navigation pane* [page 689], click **Submit to my Preferred Projects**. Otherwise, click the **Submit View** icon at the bottom of the navigation pane.
4. In the **Submit Tree**, click **NamedFaultAWF Project**.

   The **Submit** transition form opens.
5. In the **Title** field, enter one of the business names from the table in SerenaSampleTickerService Company Names and Ticker Symbols [page 531], and then click **OK**.

   The ticker symbol is returned in the **Title** field of the state form.
6. Click **GetBuyRating**.

   The **GetBuyRating** transition form opens.
7. Click **OK**.

   The buy rating is returned in the **Description** field of the state form. The **Description** field should contain the following message: "This company has a strong-buy rating."
8. In **Submit View**, click **Submit to my Preferred Projects**.
9. In the **Submit Tree**, click **NamedFaultAWF Project**.
10. In the **Title** field, type some invalid input such as **Acme Company**, and then click **OK**.

The **SerenaSampleTickerService-GetTickerSymbolFault** message appears in the **Description** field. If you typed `Acme Company`, the message should read as follows: "No information is available for 'acme company'. The companies listed on this service are 'Rob & Bert manufacturing', 'Aftabitorium', 'Meg N Ah Electric Co.', 'BlueJay Bird Supply', 'Tim Buck 2 Entertainment', and 'Carol Creations'." If you click **GetBuyRating**, the **NamedFaultOWF2** orchestration workflow is not invoked and the information in the **Title** and **Description** fields does not change.

Tutorial: Running the ThrowAWF Project

In this exercise, you run the ThrowAWF Project in the SBM User Workspace.

To run the ThrowAWF Project:

1. Log on to the SBM User Workspace.
2. Select the **FaultHandlingApp** tab.

   If the **FaultHandlingApp** tab is not visible, click the **More** tab, and then select **FaultHandlingApp** in the list.
3. If **Submit View** appears in the *navigation pane* [page 689], click **Submit to my Preferred Projects**. Otherwise, click the **Submit View** icon at the bottom of the navigation pane.
4. In the **Submit Tree**, click **ThrowAWF Project**.

   The **Submit** transition form opens.
5. In the **Title** field, type one of the company names from the table in SerenaSampleTickerService Company Names and Ticker Symbols [page 531], except Bluejay Bird Supply, and then click **OK**.

   The **ThrowAWF Project** state form opens, and the text appears in the **Title** field.
6. Click **GetTickerSymbol**.

   The **Submit** transition form opens.
7. Click **OK**.

   The **ThrowOWF** orchestration workflow is invoked. The data from the **Title** field is passed to the **IsCompanyBluejay** Decision step, which chooses a branch based on its rule. In this case, the data is not "Bluejay Bird Supply," so it selects the **IsNotBlueJay** branch and passes the data to the SerenaSampleTickerService Web service.

   The Web service's **GetTickerSymbol** operation is invoked, and the Web service returns a ticker symbol for the company name that you typed in the **Title** field. The ticker symbol appears in the **Description** field of the state page.
8. In the navigation pane, click **Submit to my Preferred Projects**.
9. In the **Submit Tree**, click **ThrowAWF Project**.

   The **Submit** transition form opens.
10. In the **Title** field, type some text, such as `zzzz`. (Do not use any of the company names from the table in SerenaSampleTickerService Company Names and Ticker Symbols [page 531], including Bluejay Bird Supply.) Click **OK**.

The **ThrowAWF Project** state form opens, and the text appears in the **Title** field.

11. Click **GetTickerSymbol**.

    The *transition form* [page 700] between the **Submit** and **TickerSymbol** states opens.

12. Click **OK**.

    The **ThrowOWF** orchestration workflow is invoked. The data from the **Title** field is passed to the **IsCompanyBluejay** Decision step, which chooses a branch based on its rule. In this case, the data in the **Title** field is not Bluejay Bird Supply, so it selects the **IsNotBluejay** branch and passes the data to the SerenaSampleTickerService Web service.

    The Web service's **GetTickerSymbol** operation is invoked, and the Web service determines that the data is not valid. The Web service generates the **GetTickerSymbolFault**, which is caught by the **GetTickerSymbolFault** Catch branch.

    The **Catch** branch passes the fault detail to the **ReturnGetTickerSymbolFault** Throw step, which throws a fault and the error text to the SBM Application Engine.

    The SBM Application Engine stops the workflow and displays the following error message on the state form, below the transition buttons: `Error occurred during web service invocation: No information is available for 'ZZZZ'. The companies listed on this service are 'Rob & Bert manufacturing', 'Aftabitorium', 'Meg N Ah Electric Co.', 'BlueJay Bird Supply', 'Jay Buck 2 Entertainment', and 'Carol Creations'.` (The text after the colon is the fault detail.)

13. In the navigation pane, click **Submit to my Preferred Projects**.

14. In the `Submit Tree`, click **ThrowAWF Project**.

    The **Submit** transition form opens.

15. In the **Title** field, type **`Bluejay Bird Supply`**, and then click **OK**.

    The **ThrowAWF Project** state form opens, and the text appears in the **Title** field.

16. Click **GetTickerSymbol**.

    The **Submit** transition form opens.

17. Click **OK**.

    The **ThrowOWF** orchestration workflow is invoked. The data from the **Title** field is passed to the **IsCompanyBluejay** Decision step, which chooses a branch based on its rule. In this case, the data is "Bluejay Bird Supply." The **Decision** step selects the **IsBluejay** branch and passes the data to the **ReportBluejay** Throw step, which throws a fault and the text mapped in the **FaultString** data element to the SBM Application Engine.

    The SBM Application Engine stops the workflow and displays the following error on the state page, below the transition buttons: `Error occurred during web service invocation: SOAPFaultCode: ns1:ERROR_BLUEJAYSOAP Fault String: Bluejay Bird Supply is a credit risk.`

Tutorial: Running the CompensateAWF Project

In this exercise, you run the CompensateAWF Project in the SBM User Workspace.

To run the CompensateAWF Project:

1. Log on to the SBM User Workspace.
2. Select the **FaultHandlingPro** tab.

   If the **FaultHandlingPro** tab is not visible, click the **More** tab, and then select **FaultHandlingPro** in the list.
3. If **Submit View** appears in the *navigation pane* [page 689], click **Submit to my Preferred Projects**. Otherwise, click the **Submit View** icon at the bottom of the navigation pane.
4. In the **Submit Tree**, click **CompensateAWF Project**.

   The **Submit** transition form opens.
5. In the **Title** field, type any of the company names from the table in SerenaSampleTickerService Company Names and Ticker Symbols [page 531], and then click **OK**.

   The **CompensateAWF Project** state form opens, and the company name appears in the **Title** field.
6. Click **DemoCompensate**.

   The **Submit** transition form opens.
7. Click **OK**, and then click the **Reload Item** button.

   The **CompensateOWF** orchestration workflow is invoked. The data from the **Title** field is passed to the **UpdateTitle** Service step. The **UpdateTitle** Service step is invoked and passes the text that is mapped to the **Title** field, "Updated by CompensateOWF," to the **GetTickerSymbol** Service step. The Web service's **GetTickerSymbol** Service step is invoked, and the text is returned to the SBM Application Engine, which displays "Updated by CompensateOWF" in the **Title** field of the state page.
8. In the navigation pane, click **Submit to my Preferred Projects**.
9. In the **Submit Tree**, click **CompensateAWF Project**.

   The **Submit** transition form opens.
10. In the **Title** field, type any text that is not contained the company names from the table in SerenaSampleTickerService Company Names and Ticker Symbols [page 531], and then click **OK**.

    The **CompensateAWF Project** state form opens, and the text you typed appears in the **Title** field.
11. Click **DemoCompensate**.

    The **Submit** transition form opens.
12. Click **OK**, and then click the **Reload Item** button.

The **CompensateOWF** orchestration workflow is invoked. The Web service's GetTickerSymbol operation is invoked, and the Web service determines that the data is not valid. The Web service generates the **GetTickerSymbolFault**, which is caught by the **GetTickerSymbolFault** Catch branch. This branch passes the fault detail to the **ReturnTickerSymbolFault** Throw step, which throws a fault to the **Compensate** step in the **FaultHandler** section of the **OuterScope**. The **Compensate** step looks for any enclosed, successfully completed scopes and finds the **UpdateTitle** scope. The **CompensationHandler** section for this scope contains a **Service** step to invoke the SBM Application Engine **TransitionItem** operation, which rolls back the change to the **Title** field that happened in the **UpdateTitle** scope. Note that this scope changed the **Title** field to "Updated by CompensateOWF," but the **Compensate** step is now rolling back that change. It also changes the **Description** field to "Returning Title to its original value." The Application Engine displays the original data from the **Title** field in the **Title** field of the state form and "Returning Title to its original value." in the **Description** field.

Raising External Events

See Raising External Events [page 556] for a procedure that shows you how to raise external events.

Chapter 28: Orchestration Use Cases

The use cases in this section describe how to implement common use cases that use orchestrations.

This section contains the following use cases:

- Building Dynamic Arrays [page 539]
- Getting Values from a Multi-User Field [page 547]
- Creating Primary Items Based on Multi-Relational Field Values [page 551]
- Raising External Events [page 556]

Building Dynamic Arrays

In SBM Composer, you can build an array with a fixed number of array elements at *design time*. This is known as a *fixed array*.

However, sometimes you must determine the number of array elements at *runtime* (that is, when an *orchestration workflow* [page 690] runs). For example, you want to send telephone numbers to an SMS (Short Message Service) Web service. You use a **Service** step in an orchestration workflow to get the telephone numbers of those users who were selected on a form in the SBM User Workspace.

Because you do not know the number of selected users and telephone numbers in advance, you build a *dynamic array*. For an orchestration workflow to dynamically add elements to an array, you use a **Calculate** step to target each element in the array in sequential order, beginning with 1.

This section includes use cases that illustrate ways to make Web service calls that accept a dynamically-sized list of elements. They are for demonstration purposes only, and do not represent realistic scenarios.

Important: If you do not use dynamic arrays, you must make multiple Web service calls with one element in each array, or create an array in advance that has a fixed length (for example, a 20-element array). These methods have a negative impact on performance.

CAUTION: Array elements must be ordered chronologically. Do not split arrays or interject an array element into the middle of an array. For example, you cannot interject an array element between two existing array elements, and you cannot create a seventh array element if there is no sixth array element.

Use Case: Creating an Array to Use in a Subsequent Service Step

In this use case, you get an array of file names from an *auxiliary table* [page 680] that stores Source Code Management (SCM) *application* [page 679] information, and then create a new item for each file in another auxiliary table that stores Issue Defect Management (IDM) information.

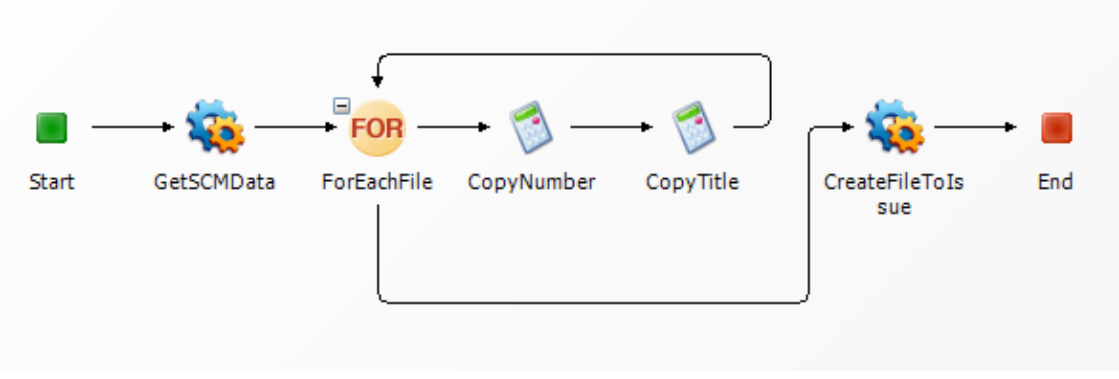
Note: Similar data can come from any external Web service. That is, instead of coming from an auxiliary table, the data about files and associated issues can come from your SCM application.

The *orchestration workflow* [page 690] loops through each file and then processes it as follows:

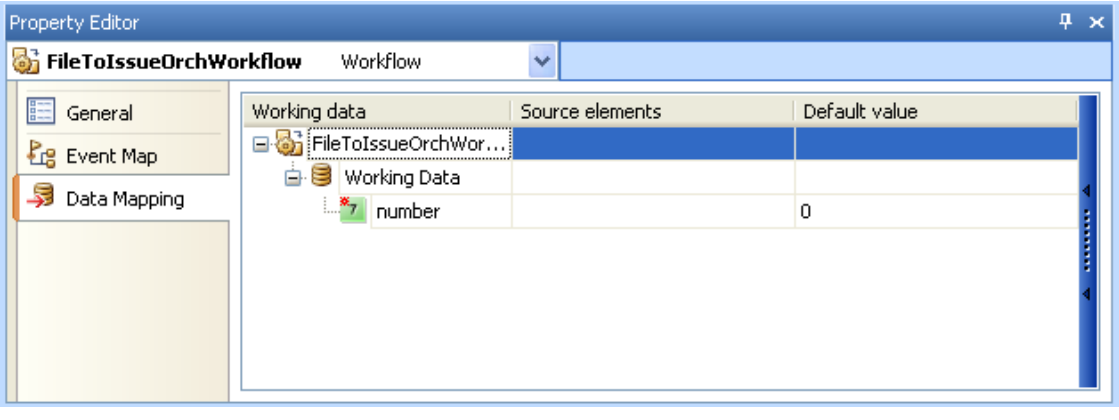
1. Gets an array of files from an auxiliary table.
2. Adds the file names to an input element in a subsequent **Service** step.
3. Creates new items in another auxiliary table.

To create an array and use it in a subsequent Service step:

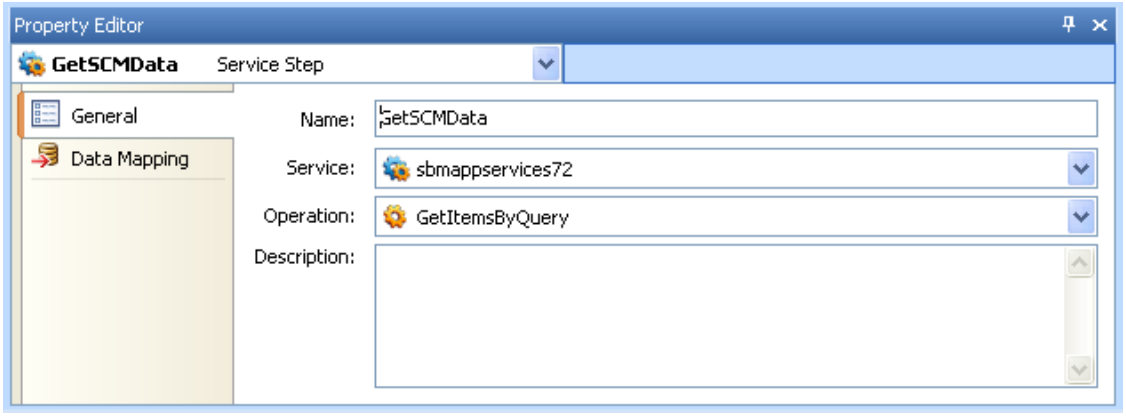
1. Add steps to the orchestration workflow as shown below.

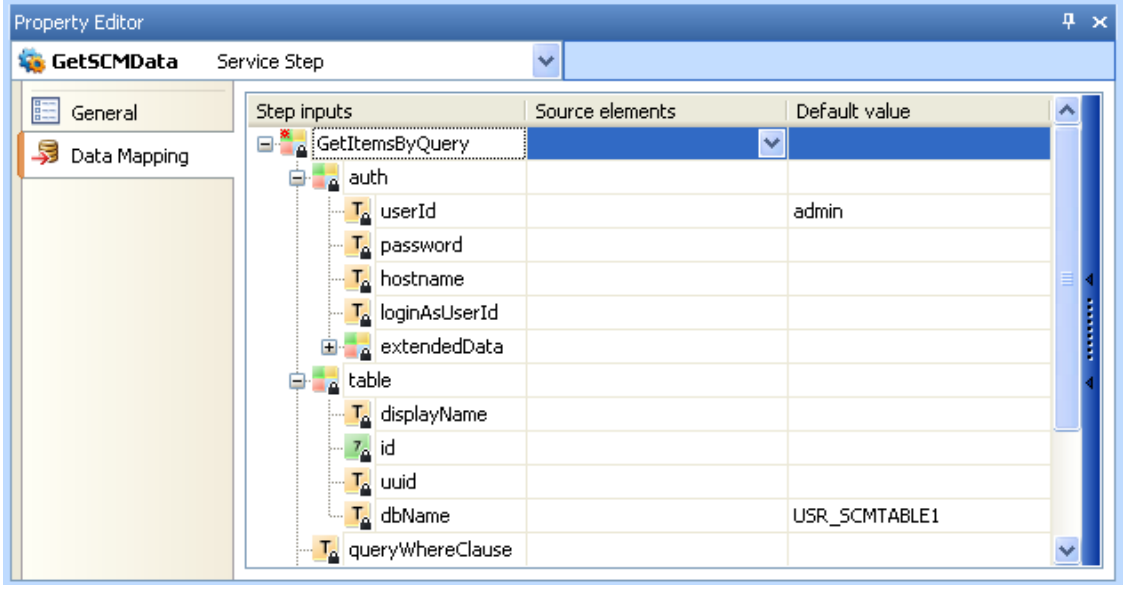


2. In the **Working Data** for the workflow, add a variable to store the number of loops through the **ForEachFile** step, using the type "Integer." Set the default value of the variable to `0`.

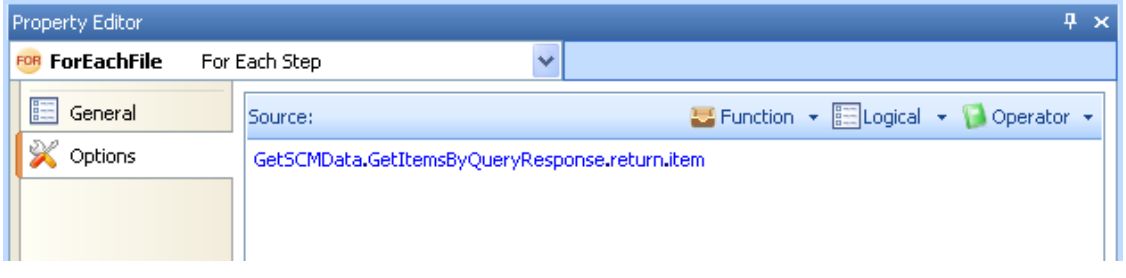


3. Configure the **GetSCMData** step to get a list of file names from the auxiliary table for the SCM application.

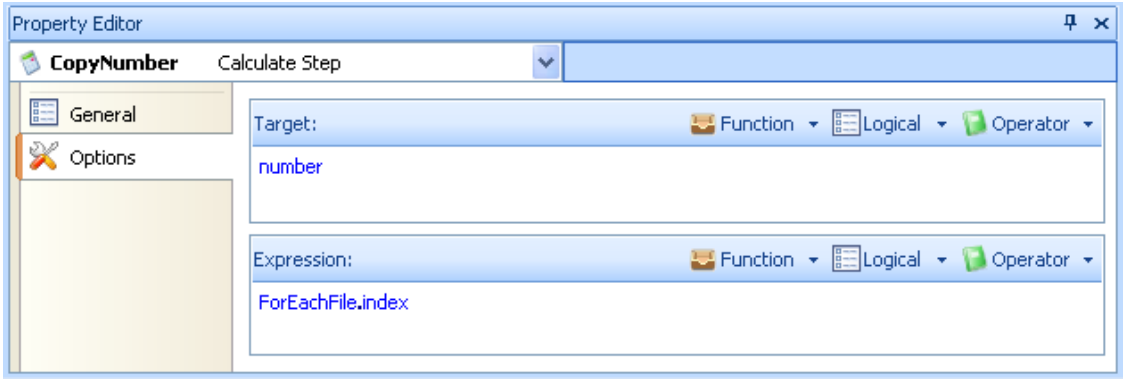




4. Configure the **ForEachFile** step to repeat the operations in the loop for each file returned in the **GetSCMData** response.

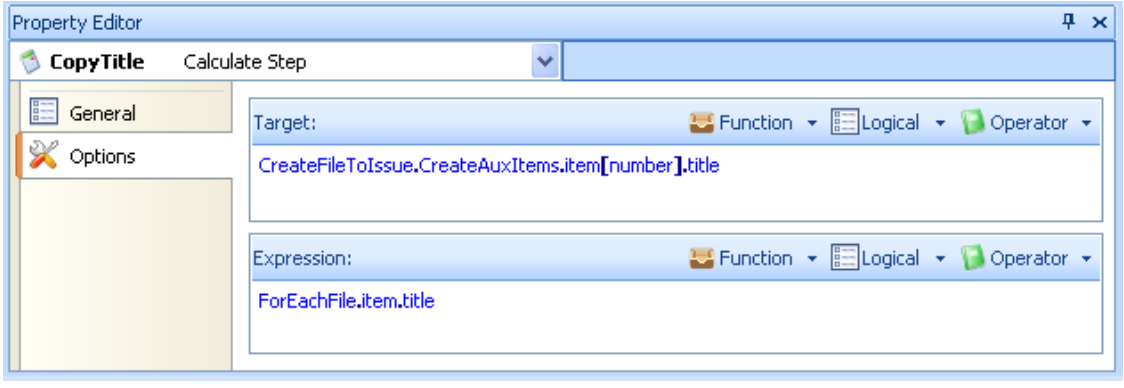


5. Configure the **CopyNumber** step to store the number of times the orchestration workflow loops through the **ForEachFile** step.

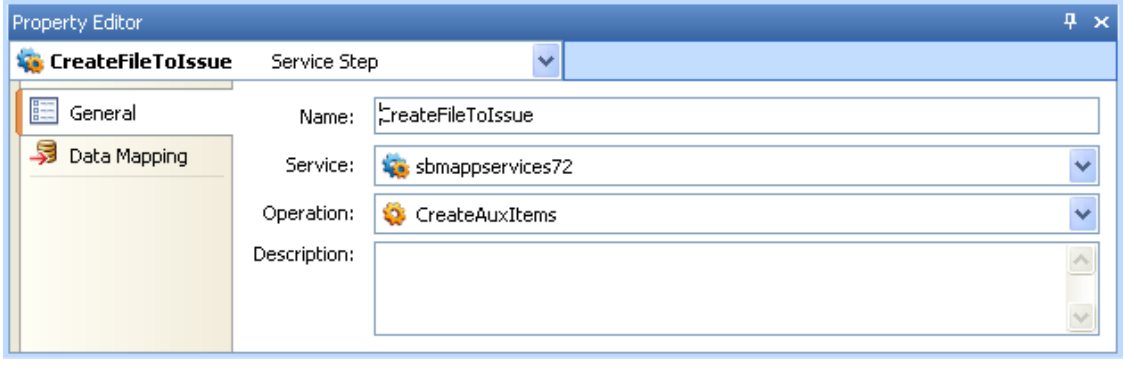


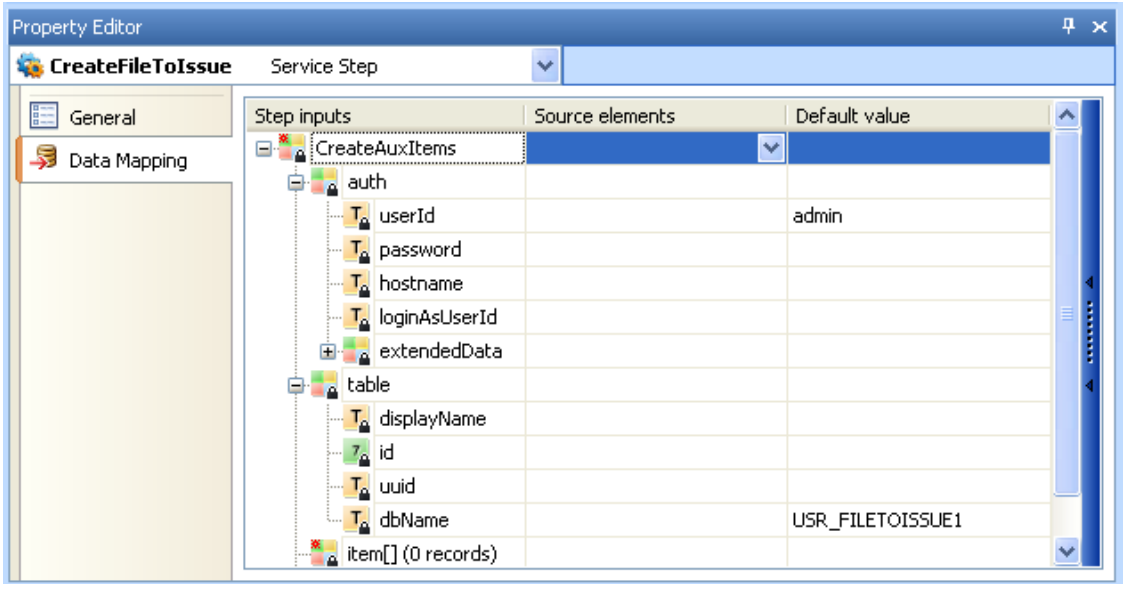
Note: This step must be included. You must copy the number of loops into the working data; you cannot use the **ForEachFile** step by itself.

6. Configure the **CopyTitle** step to set the value of the **title** in the **item[]** array in the **CreateFiletoIssue** step. This value is a file name returned by the **GetSCMData** step.



7. Configure the **CreateFileToIssue** step to create a new item in another auxiliary table for each file that was returned by the **GetSCMData** step.





Use Case: Populating Custom Fields

In this use case, you add two *Text* custom fields to the primary table: *Cities* and *Name*. Then you dynamically create an extended field list in a **Service** step that populates these fields in the SBM User Workspace.

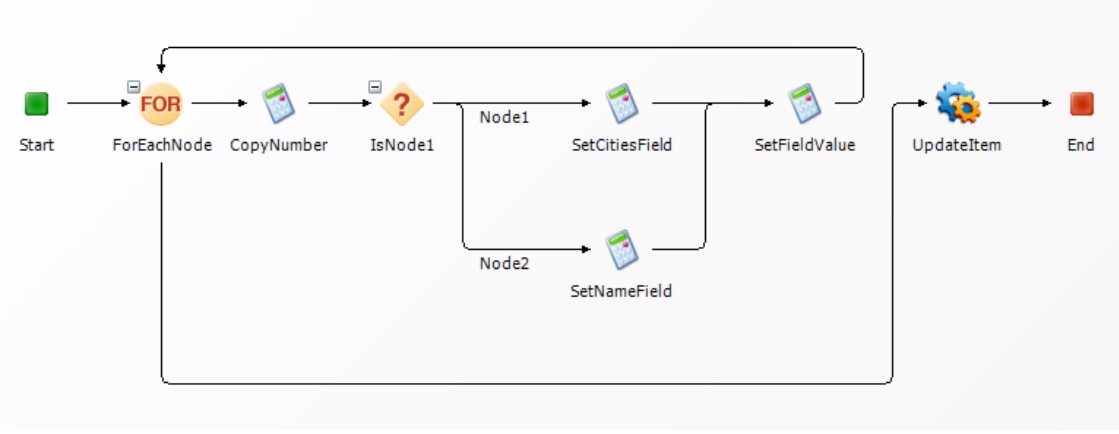
Note: In this use case, you use working data as input to the **ForEachNode** loop, and the **ForEachNode** loop creates an extended field list as an input to the **UpdateItem** step. The same logic can be used when an *event* [page 685] contains array records (that is, you have arrays included in the **Extension** data in an *event definition* [page 685]). Instead of using the working data as the input to the **ForEachNode** step, you can use the **EventNotice**.

The *orchestration workflow* [page 690] loops through each node and processes it as follows:

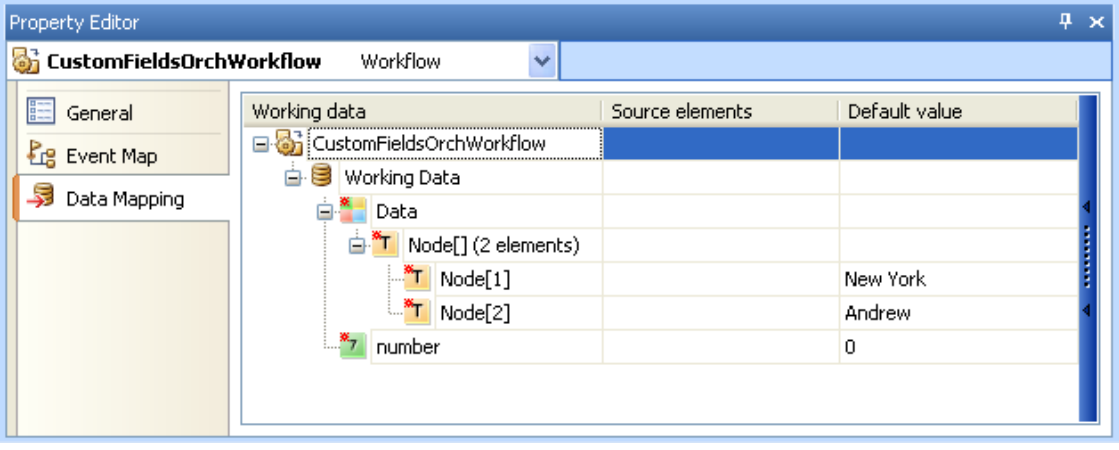
1. Sets the database name of the first field in the extended field list.
2. Sets the value of the first field in the extended field list.
3. Sets the database name of the second field in the extended field list.
4. Sets the value of the second field in the extended field list.
5. Updates the fields in the SBM item with information in the extended field list.

To dynamically create an extended field list that populates *custom fields* [page 682]:

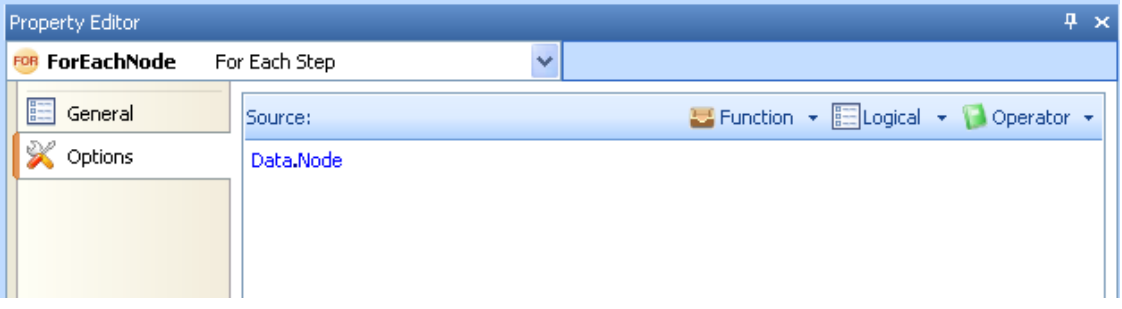
1. Add steps to the orchestration workflow as shown below.



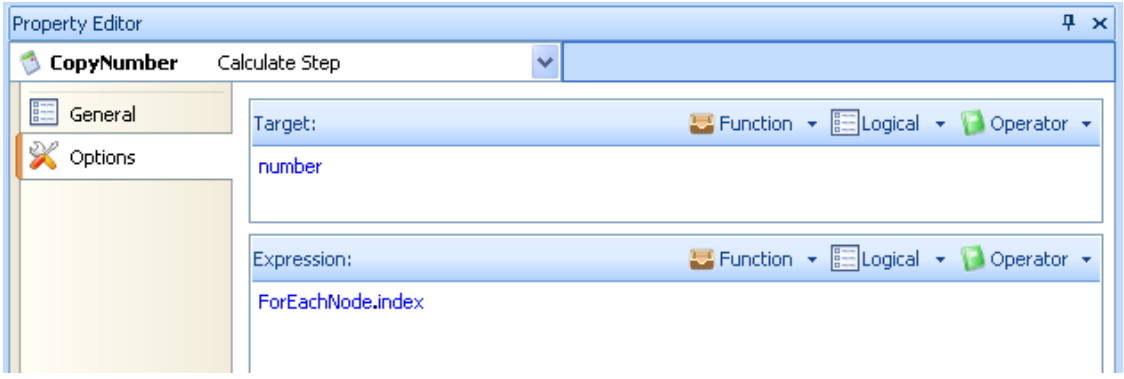
2. Define the working data for the orchestration workflow:
   a. Create an array with two array elements. Set a default value for each array element.
   b. Add a variable to store the number of loops through the **ForEachNode** step, using the type "Integer." Set the default value of the variable to `0`.



3. In the **ForEachNode** step, enter an expression that describes the source of the data to be processed. In this example, the **Node** array is the source of the data.

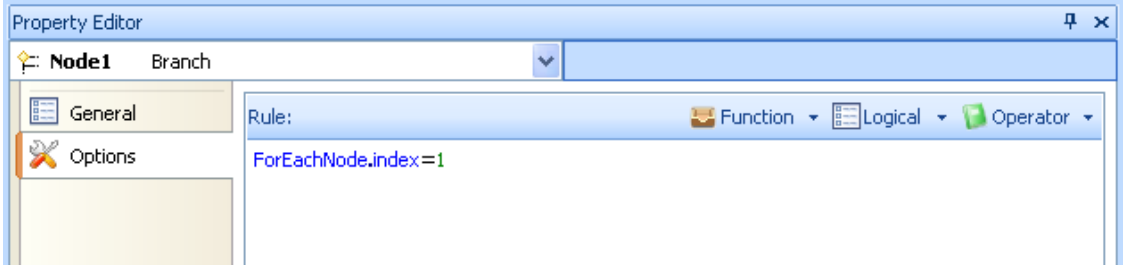


4. Configure the **CopyNumber** step to store the number of times the orchestration workflow loops through the **ForEachNode** step.

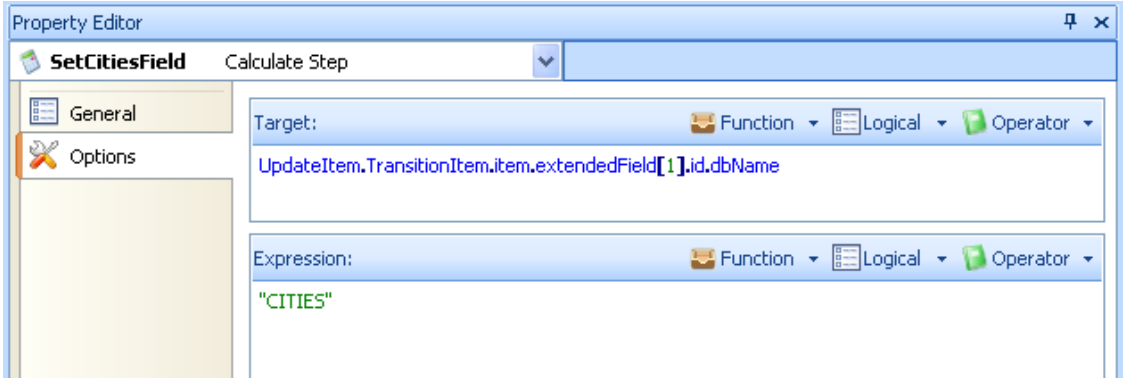


Note: This step must be included. You must copy the number of loops into the working data; you cannot use the **ForEachNode** step by itself.

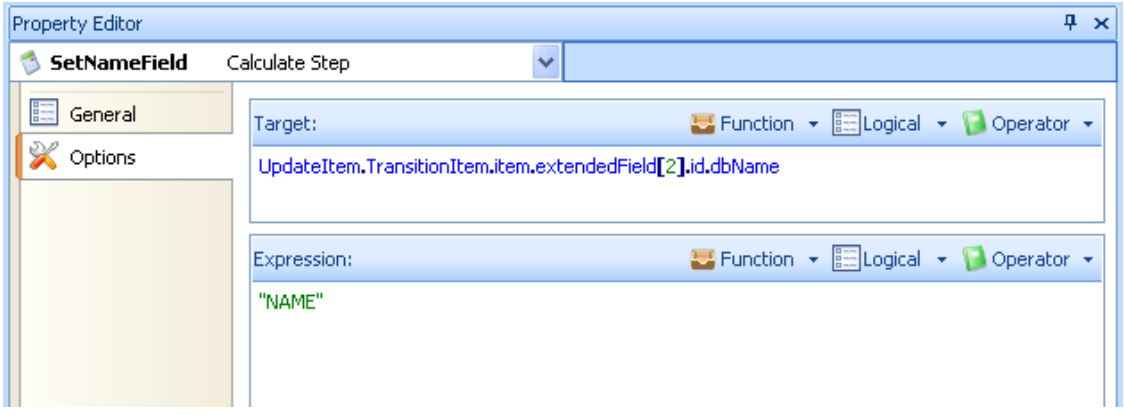
5. Configure the **Node1** branch to set the index value for the "cities" node to "1."



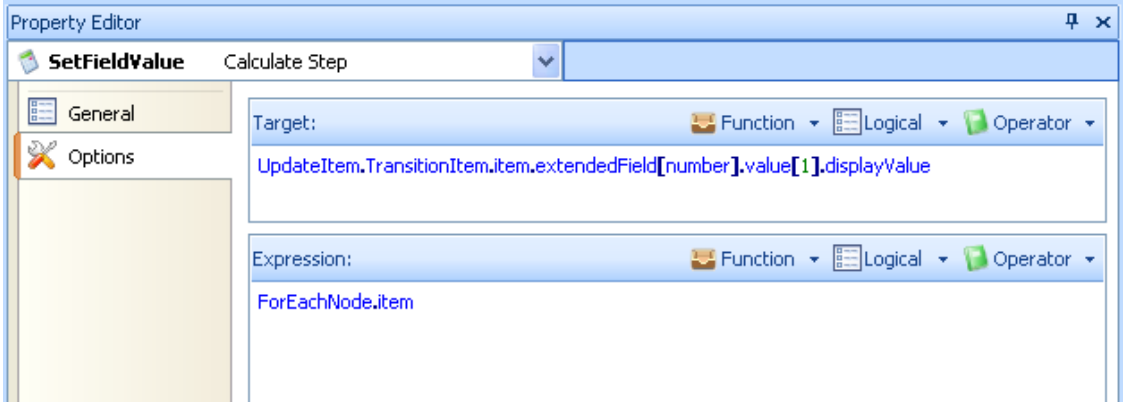
6. Configure the **SetCitiesField** step to set the database name of the first extended field to CITIES.



7. Configure the **SetNameField** step to set the database name of the second extended field to NAME.

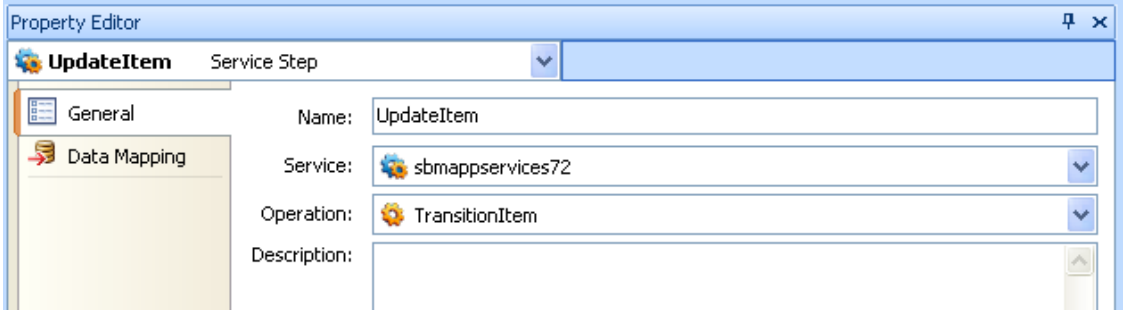


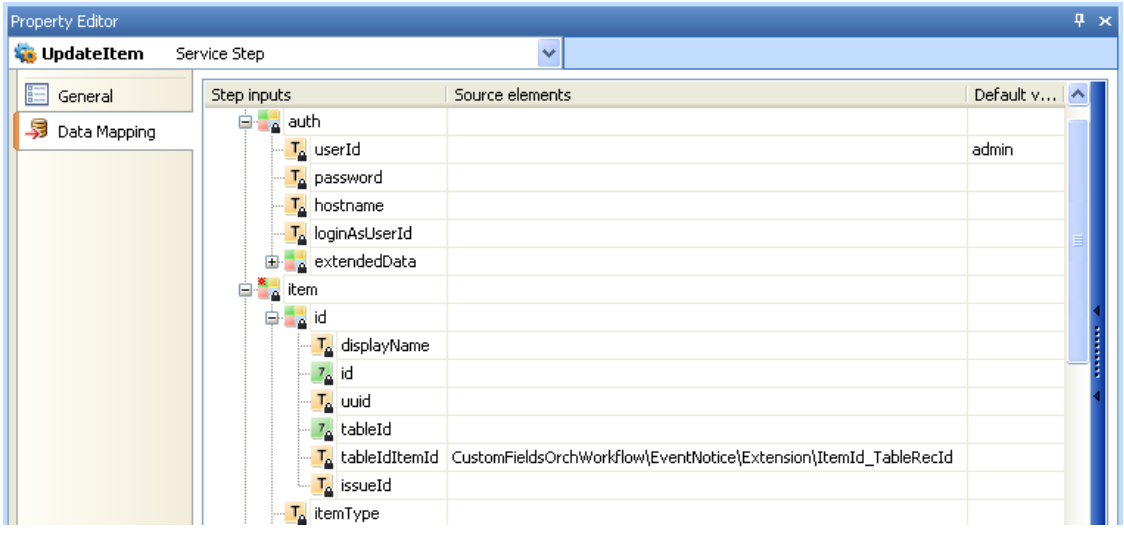
8. Configure the **SetFieldValue** step to set the value of the applicable extended field.



Note: In this use case, you use the first array element to create the CITIES extended field and the second array element to create the NAME extended field. The index used in the **Target** expression for the **SetCitiesField** and **SetNameField** steps shows the first array element using **[1]**, and the second array element using **[2]**. The order in which these fields are defined in the *application table* [page 679] does not matter; you can use any order in the orchestration workflow. However, after the order is set, the values for those fields must be in the same order. For example, after **[1]** is assigned "CITIES," its value will be **extendedField[1].value[1].displayValue**.

9. Configure the **UpdateItem** step to populate the custom fields in SBM items.





Getting Values from a Multi-User Field

This procedure describes a way to get values from a *Multi-User* field so you can manipulate the values. In the example used in this topic, the item is updated with a comma-delimited list containing the login IDs of the users. This list could instead be sent to an external product that expects the list.

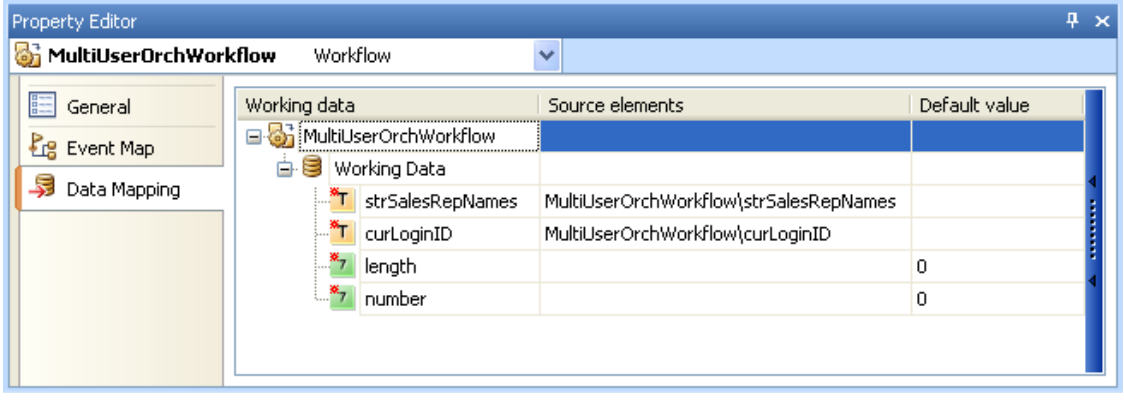
Important: This procedure can also be used with *Multi-Selection* and *Multi-Relational* fields.

The *orchestration workflow* [page 690] loops through each selection in the *Multi-User* field and processes it as follows:

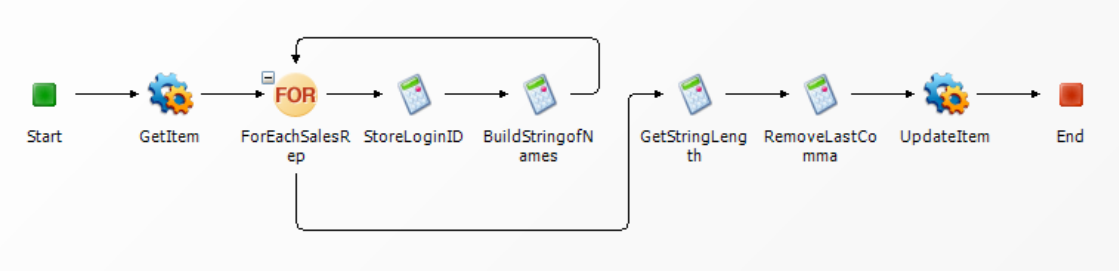
1. Gets sales representatives that were selected in an item.
2. Stores their login IDs to a string.
3. Appends that string to another working data string. The working data string will be formatted as a comma-delimited list of display values, and will be mapped to a field that will be display the list in the SBM item.

To get values from a *Multi-User* field:

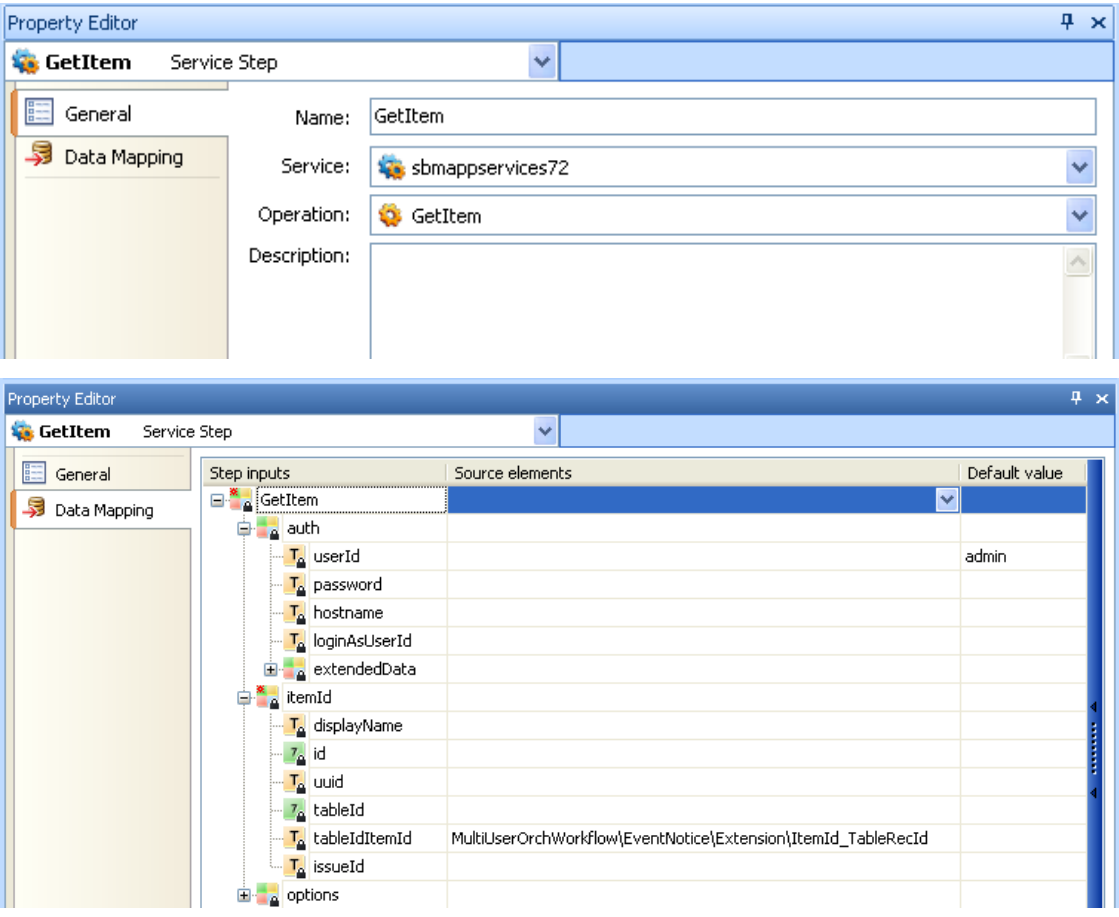
1. Add variables to the working data as shown in the following illustration.



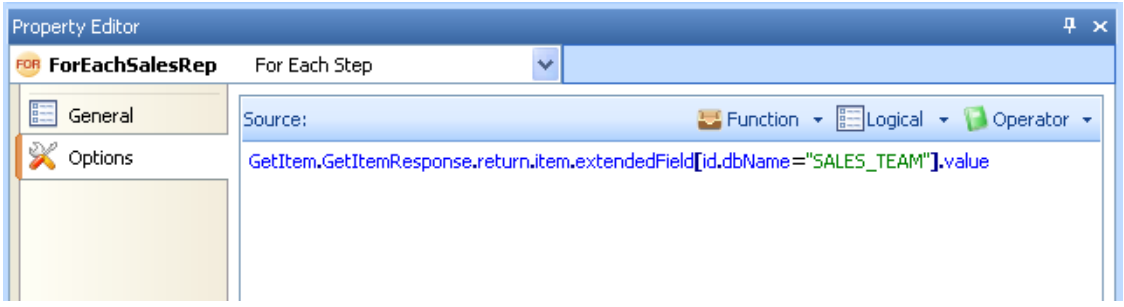
2. In the orchestration workflow, add steps to create the logic for getting and formatting the login IDs of the sales representatives selected in the item.



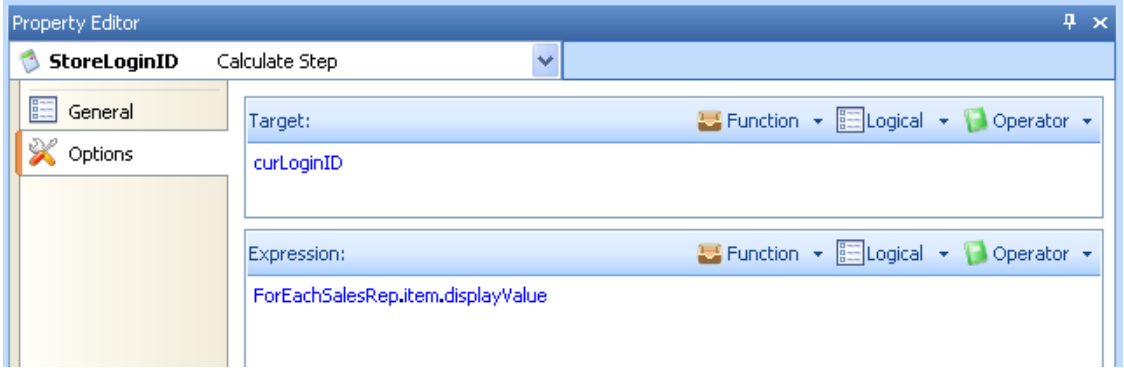
3. Configure the **GetItem** step to get the current item.



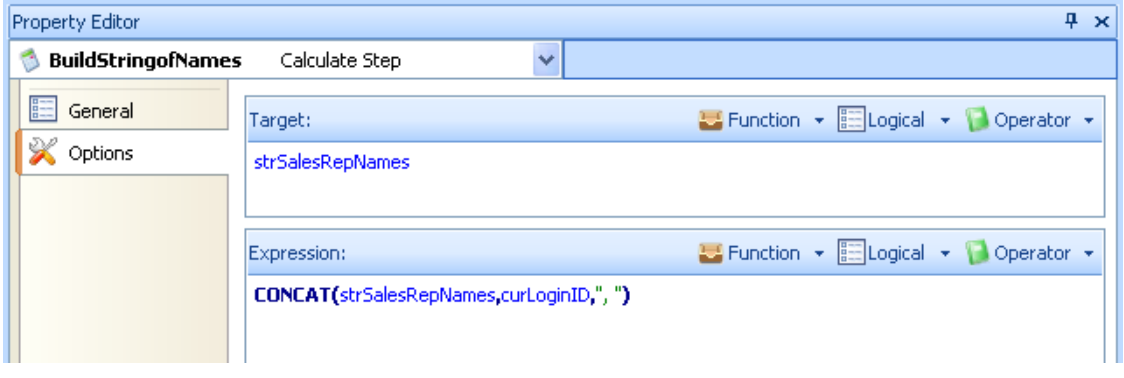
4. Configure the **ForEachSalesRep** step get the login ID for each selected sales representative in the *Sales Team* field.



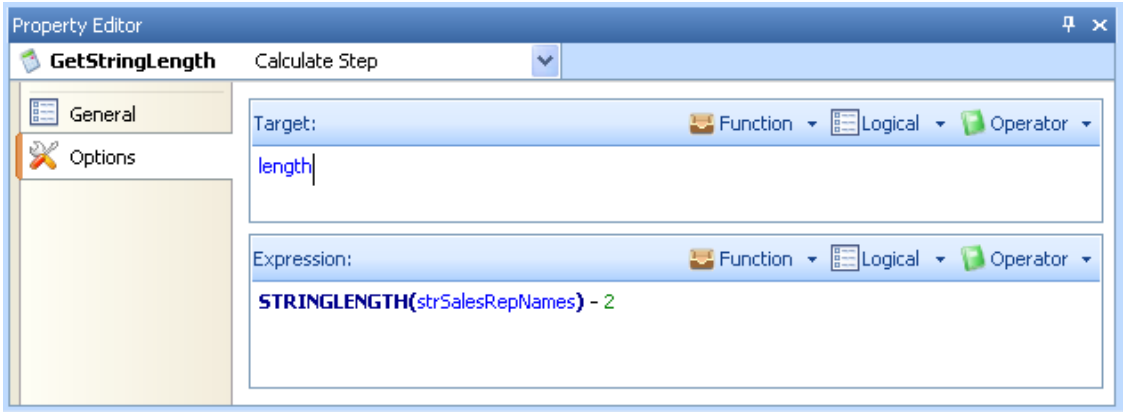
5. Configure the **StoreLoginId** step to store the "displayValue" for each sales representative. The "displayValue" represents the login ID and user object, not the display name.



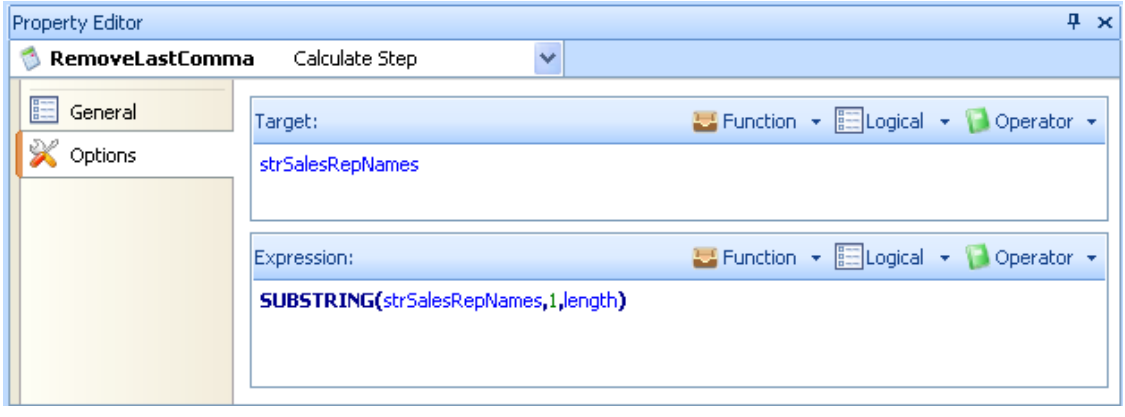
6. Configure the **BuildStringOfNames** step to append the login ID to the string of login IDs.



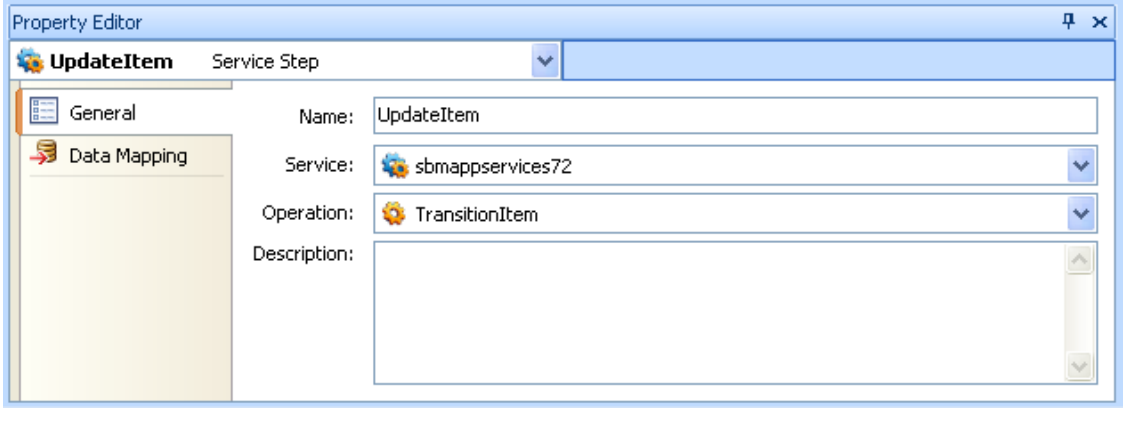
7. Configure the **GetStringLength** step to get the length of the string you want to keep. (The last two characters in the string, a comma and a space, will be removed.)

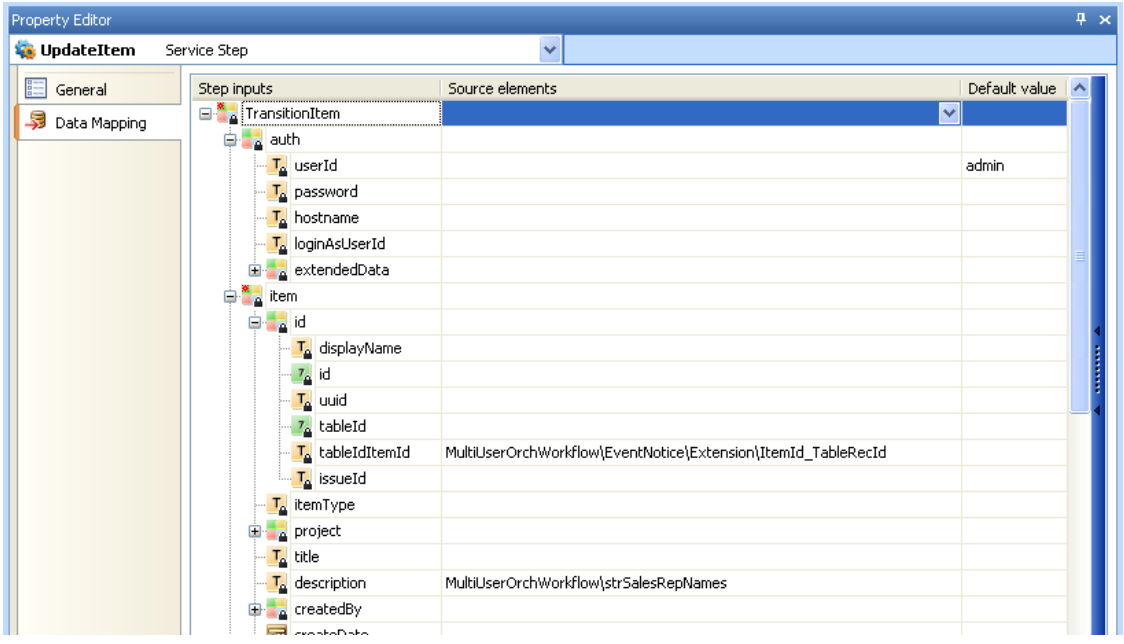


8. Configure the **RemoveLastComma** step to remove the last two characters (a comma and a space) from the string after all of the login IDs are added.



9. Configure the **UpdateItem** step to add the string to the *Description* field.





Creating Primary Items Based on Multi-Relational Field Values

You can create a set of items based on the values that are selected in a *Multi-Relational* field.

A tester does the following:

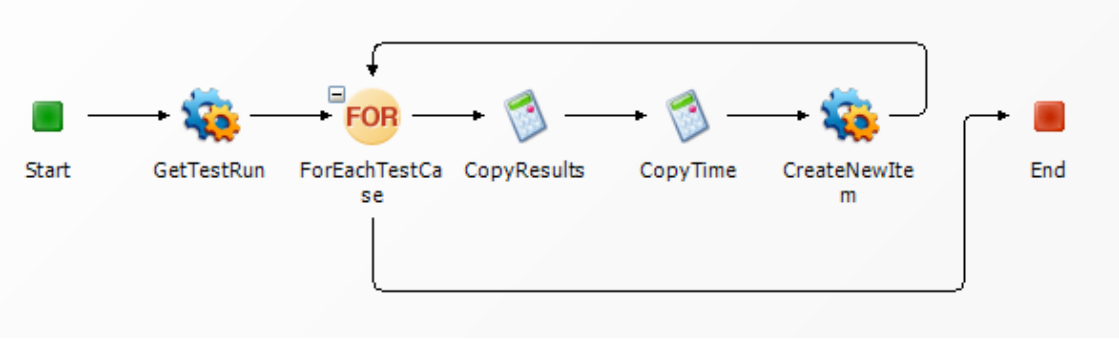
1. Submits a new issue to begin the testing process.
2. Selects associated test cases from the test case library. (The *Test Cases* field is a *Multi-Relational* field that references the `TEST_CASES` auxiliary table.)
3. Copies information to create a working copy of the test case. The working copy is used to create a new item.

The *orchestration workflow* [page 690] does the following:

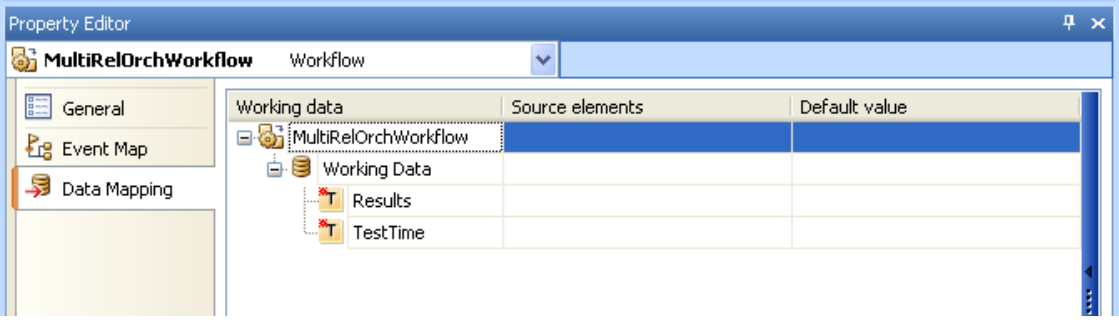
1. Gets the current item.
2. Gets the first test case that the user selected in the *Test Cases* field.
3. Copies information to create a new item.
4. Creates the new item.
5. Repeats this process for each selected test case. The new items are displayed as subtasks in the current item.

To create primary items based on *Multi-Relational* field values:

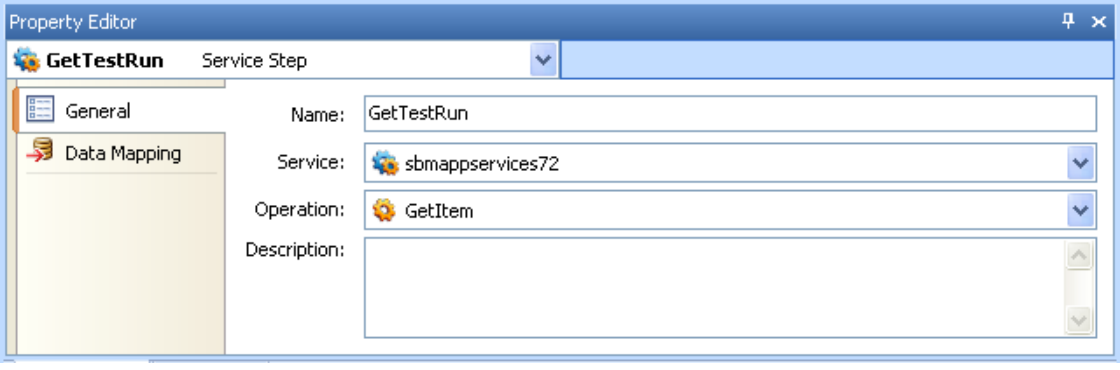
1. Add steps to the orchestration workflow as shown below.

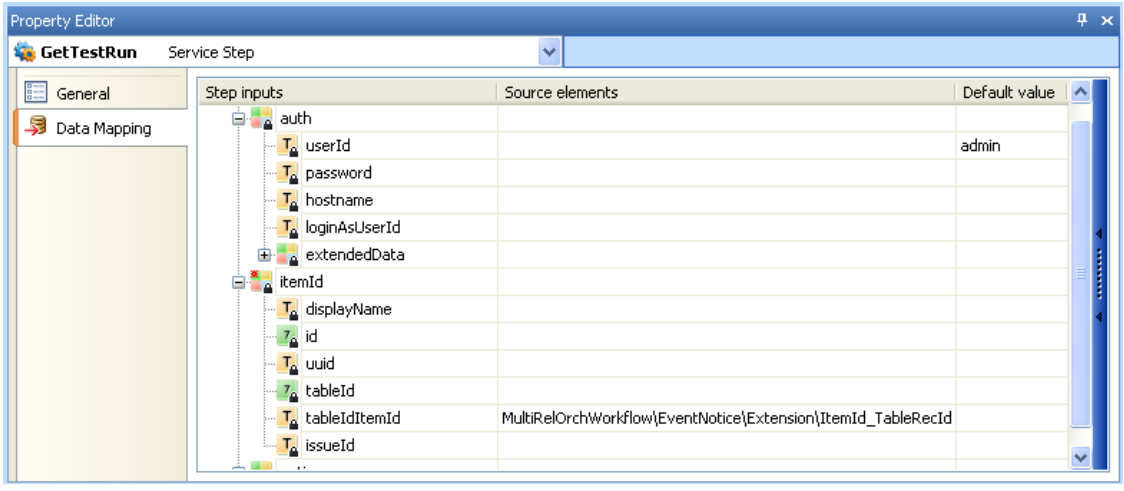


2. Define the working data for the orchestration workflow as shown in the following illustration.



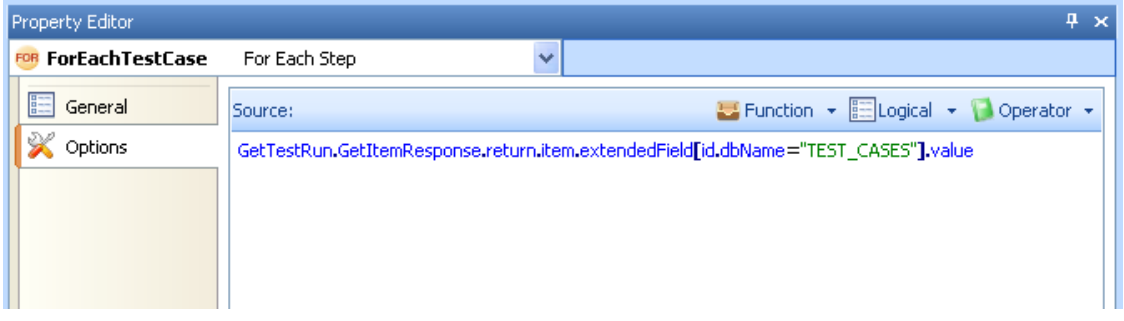
3. Configure the **GetTestRun** step to get the the current item.



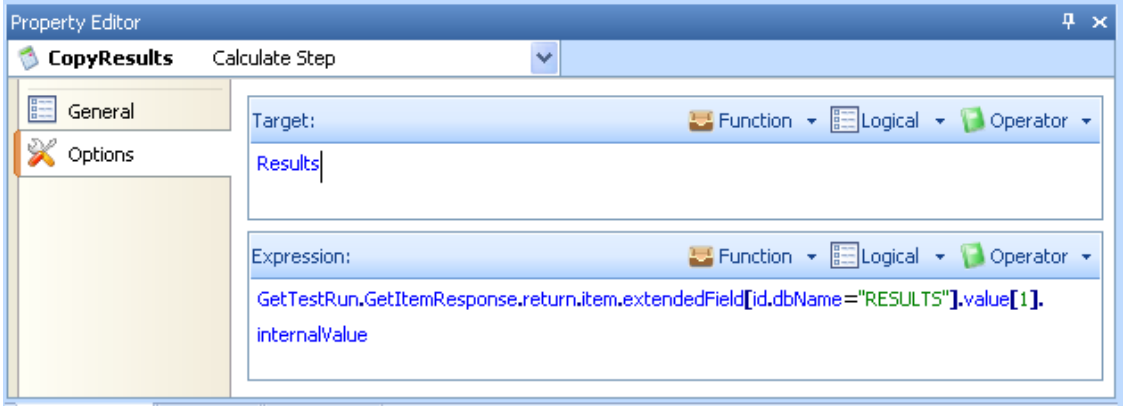


4. In the **ForEachTestCase** step, enter an expression that describes the source of the data to be processed in this loop. In this example, the step processes each test case the tester selected in the *Test Cases* field in the item.

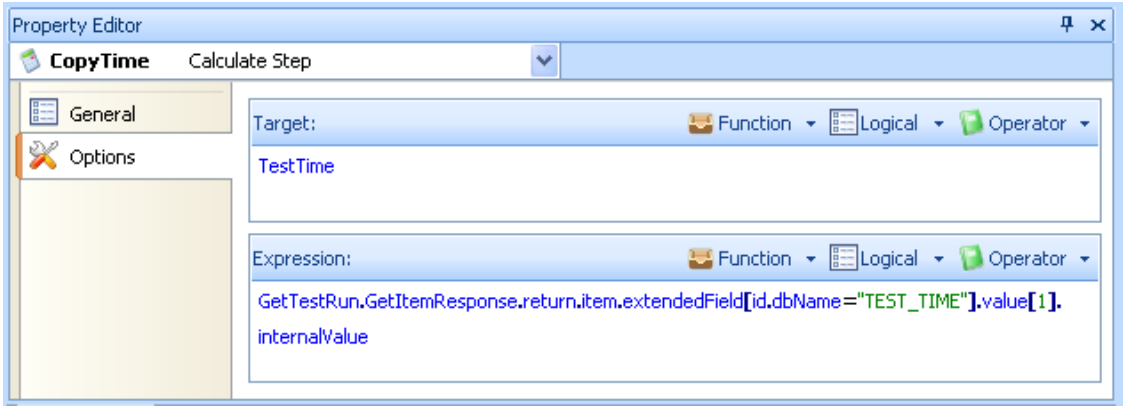
   Attention: This is the step that gets the *Multi-Relational* field values.



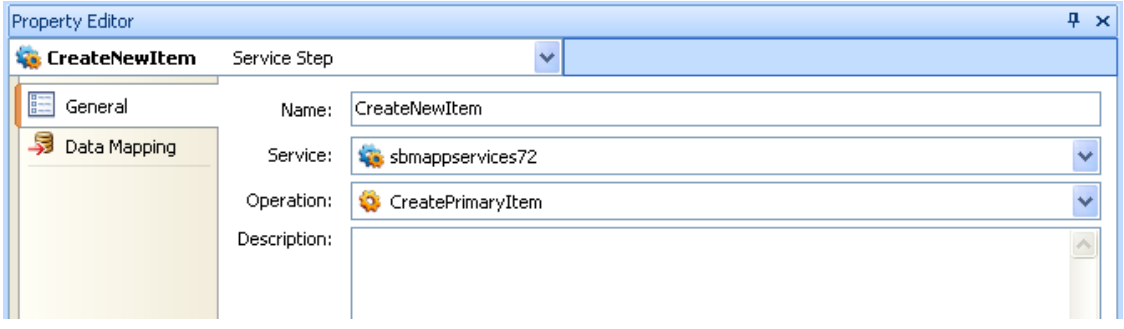
5. Configure the **CopyResults** step to copy the value of the *Results* field.

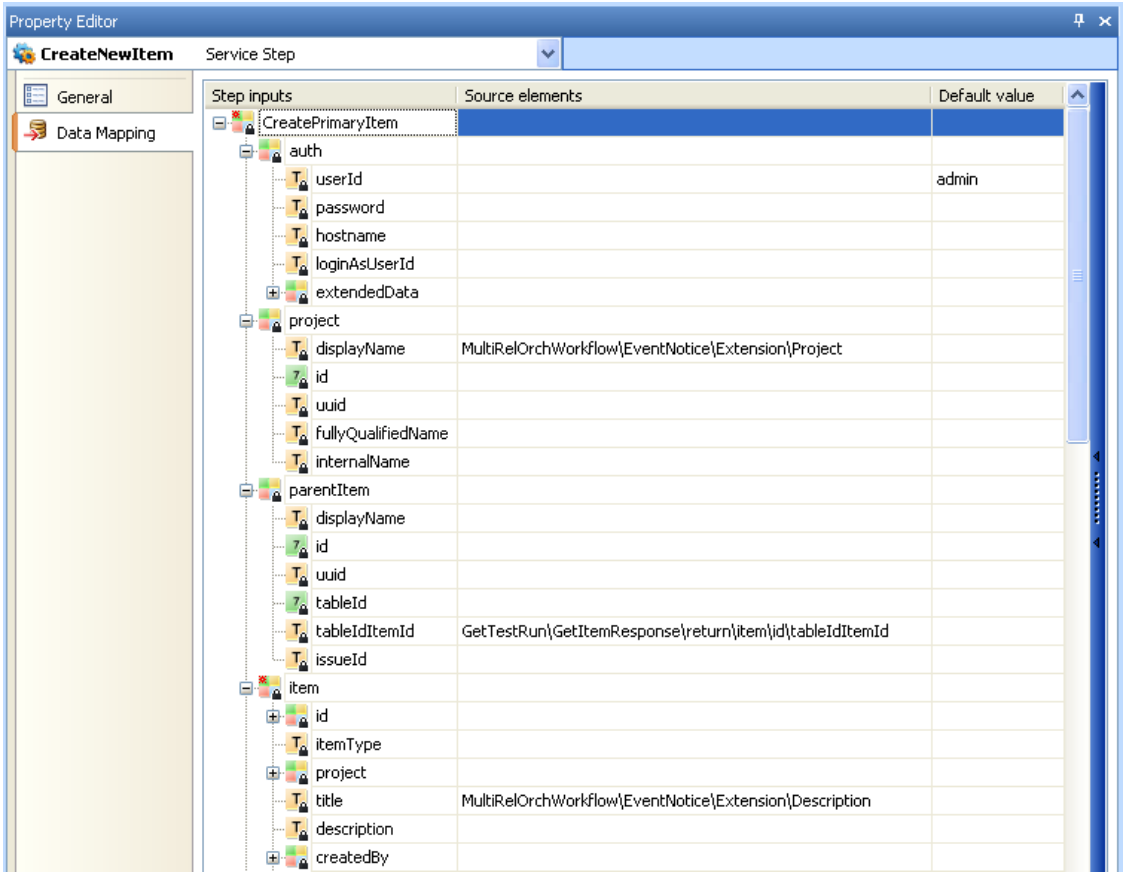


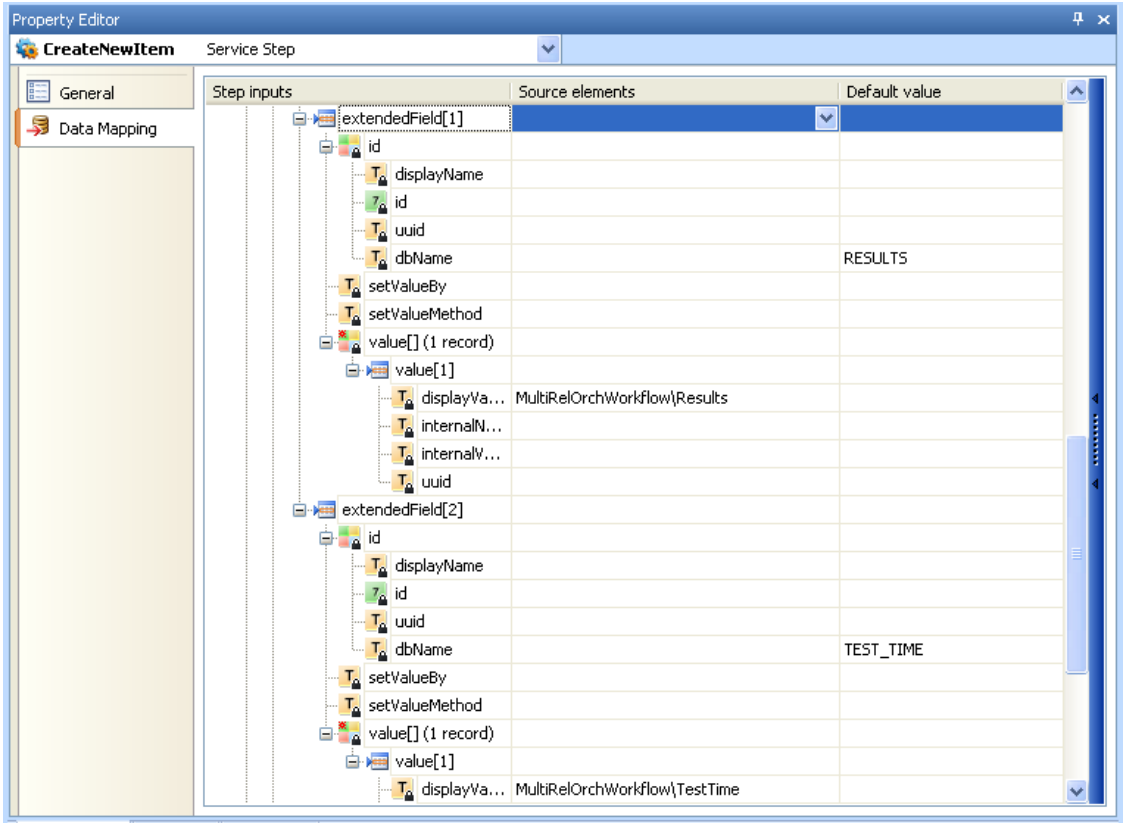
6. Configure the **CopyTime** step to copy the value of the *Test Time* field.



7. Configure the **CreateNewItem** step to create each new item and display the new items in the current item. The title of the new items is based on what was entered in the *Description* field.




Property Editor
CreateNewItem
Service Step
General
Data Mapping
Step inputs
Source elements
Default value
CreatePrimaryItem
auth
userId
admin
password
hostname
loginAsUserId
extendedData
project
displayName
MultiRelOrchWorkflow\EventNotice\Extension\Project
id
uuid
fullyQualifiedName
internalName
parentItem
displayName
id
uuid
tableId
tableIdItemId
GetTestRun\GetItemResponse\return\item\id\tableIdItemId
issueId
item
id
itemType
project
title
MultiRelOrchWorkflow\EventNotice\Extension\Description
description
createdBy



The workflow repeats the steps in the **ForEachTestCase** loop until a new item is created for each of the selected test cases.

Raising External Events

Note: This topic assumes that you understand events and event definitions. For information about them, see About Events [page 437] and About Application Links and Event Definitions [page 439].

Any external product that is capable of calling a Web service can raise events in SBM by calling a corresponding Web service. SBM Composer provides the ability to automatically generate a `.wsdl` file that you can use in the external product to call the Web service.

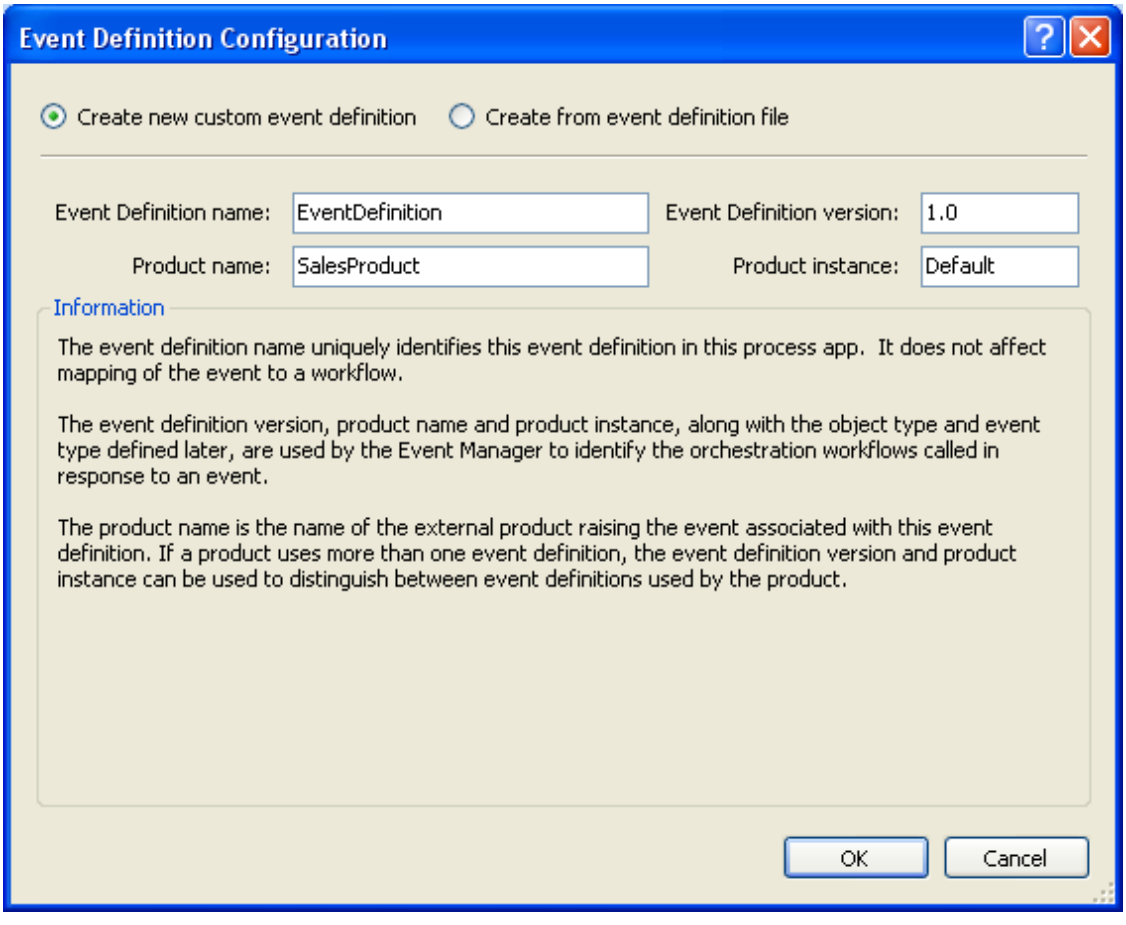
For example, suppose you use Salesforce.com to track customer prospects, and want to have a new item created in SBM whenever a potential customer is initially contacted. You can raise an external event that causes an asynchronous *orchestration workflow* [page 690] to run whenever this event occurs. The orchestration workflow creates the item in SBM.

Because the event provides the inputs for the workflow, the `.wsdl` file must be designed specifically for the event and its associated orchestration workflows. You start by defining a custom event definition that defines the data you want to pass to the workflow. This event definition defines both the event that the external product will raise and the inputs for the orchestration workflow. The event definition determines the content of the `.wsdl` file. When you click the **Export external event WSDL** button in the event definition Property Editor, the `.wsdl` file that will be used to raise the event is created.

The following procedure describes a way to raise an external event that updates a field in a SBM item. In this procedure, a sales product sends an external event to update the *Discount Percent* field. In SBM Composer, you define two custom data fields in a custom event definition: **ItemID** and **DiscountPercent**. The orchestration workflow that is invoked when the external event is raised updates the item with the specified discount amount.

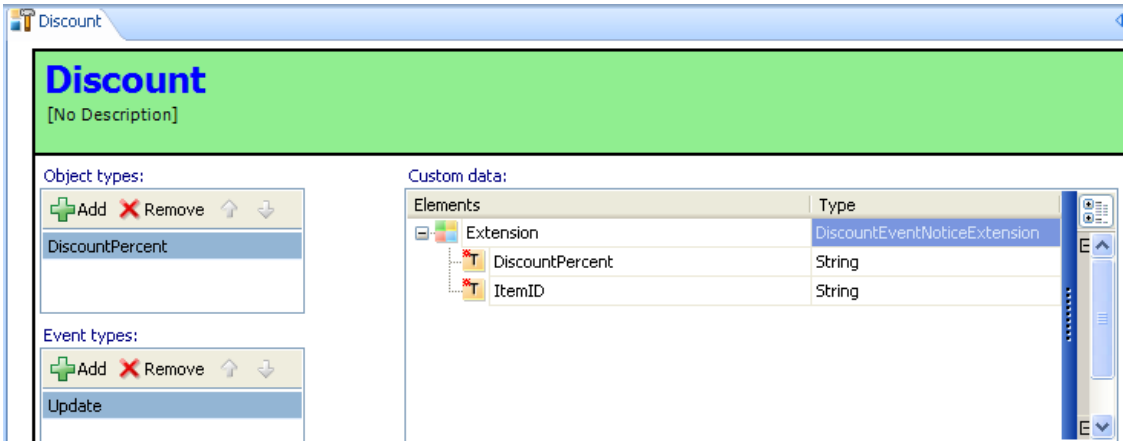
To update the discount percent value in a SBM item:

1. Create a new orchestration process app.
2. In App Explorer, right-click **Application Links**, and then select **Add New Event Definition**.
3. In the **Event Definition Configuration** dialog box that opens, create a new custom event definition as shown in the following illustration, and then click **OK**.

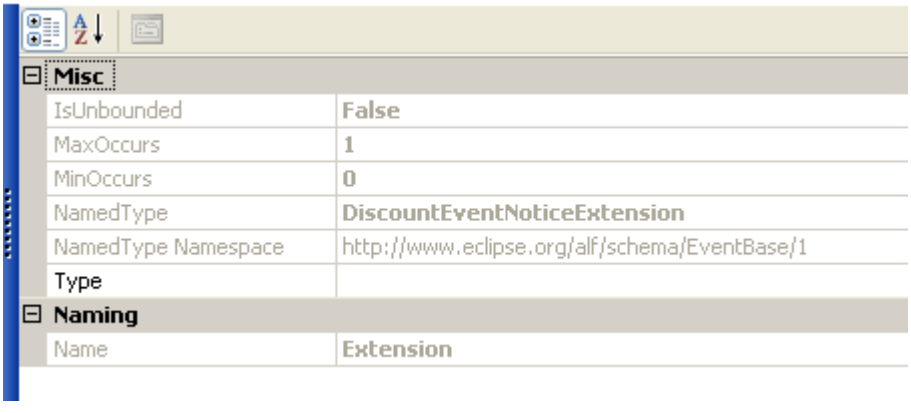


4. Define values in the event definition editor that opens as shown in the following illustration. See Creating a New Custom Event Definition [page 475] for more information.

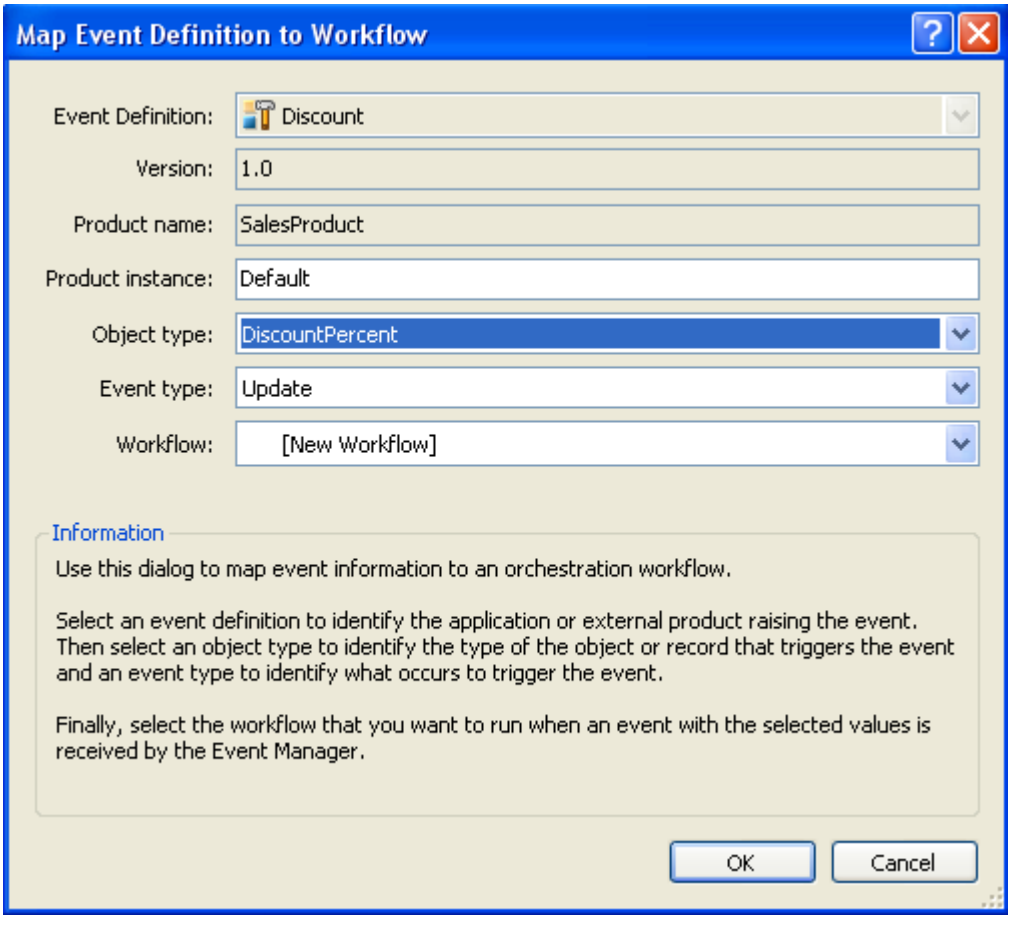
   The **Extension** element is automatically added to the **Custom data**. Its **Type** is the event definition name with **EventNoticeExtension** appended to it.



The properties for the **Extension** element include its named type and namespace.

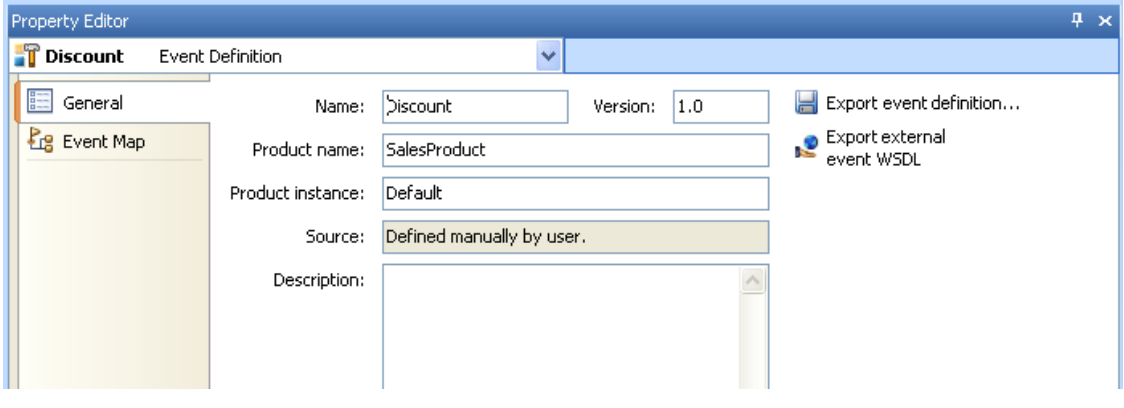


5. On the **Event Map** tab of the event definition Property Editor, click **Add**.
6. In the **Map Event Definition to Workflow** dialog box that opens, select **[New Workflow]** and then click **OK**.

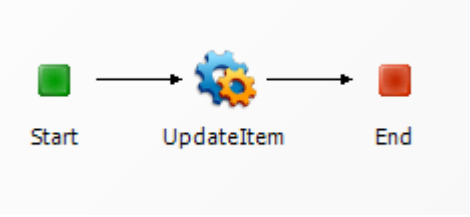


A new orchestration workflow is added to App Explorer.

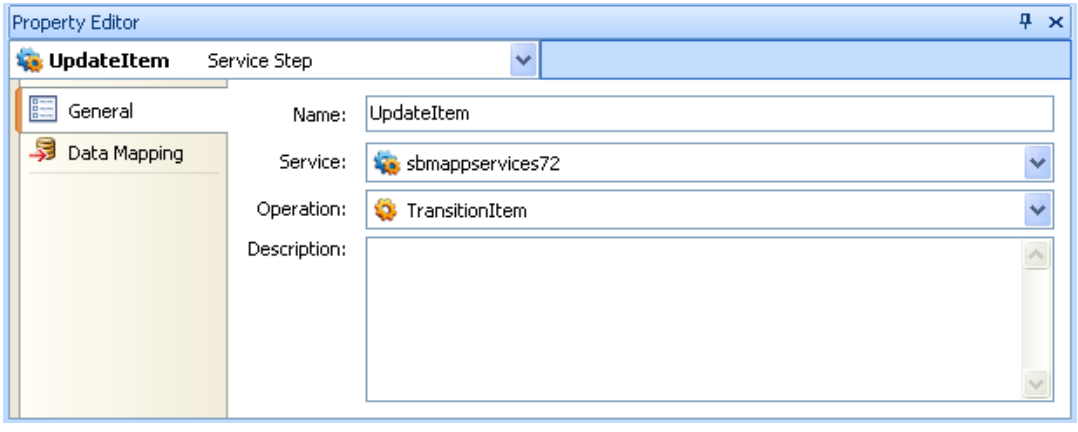
7. On the **General** tab of the event definition Property Editor, click **Export external event WSDL**, and save the `.wsdl` file to the file system of your computer.



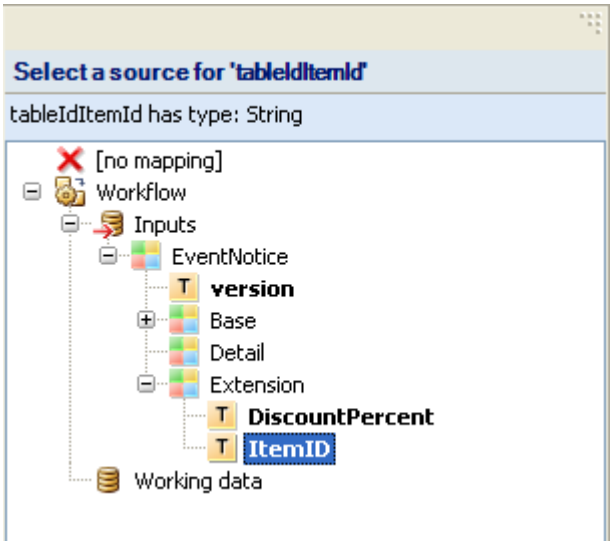
8. Click the orchestration workflow that was added to App Explorer and add a **Service** step to it as shown below. This is the orchestration workflow that will be invoked when the external event is raised.



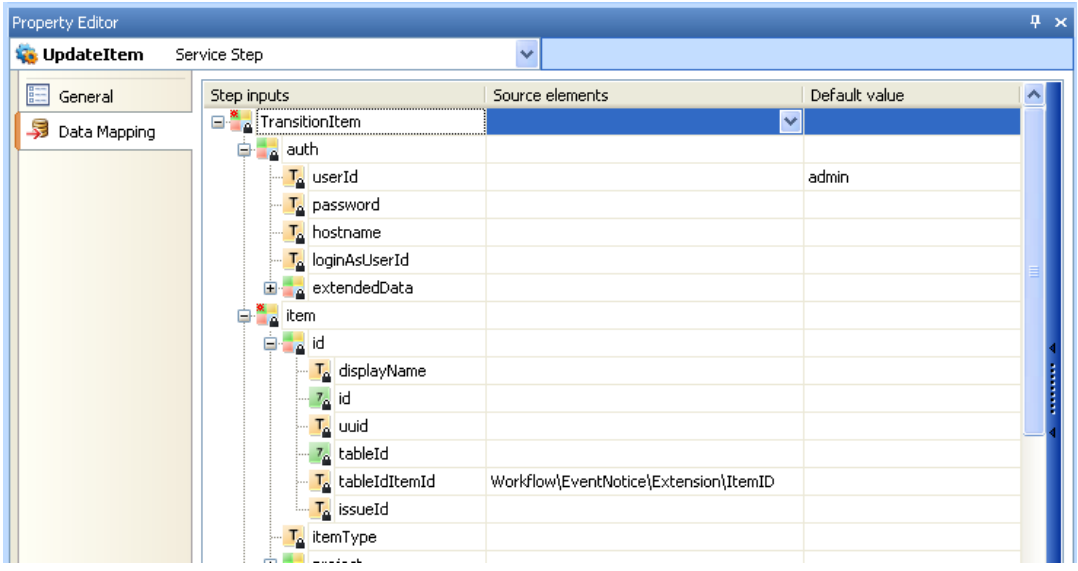
9. Configure the **UpdateItem** step to update the discount amount:
   a. On the **General** tab of the step Property Editor, select the Web service and operation shown in the following illustration.



   b. Click the **Data Mapping** tab of the step Property Editor.
   c. Expand **id** under **item**, click in the **Source elements** column for the **tableIdItemId** step input, and then select **ItemID** under **Extension** in the **Select a source** dialog box that opens. (This is one of the custom data fields you added to the custom event definition.)

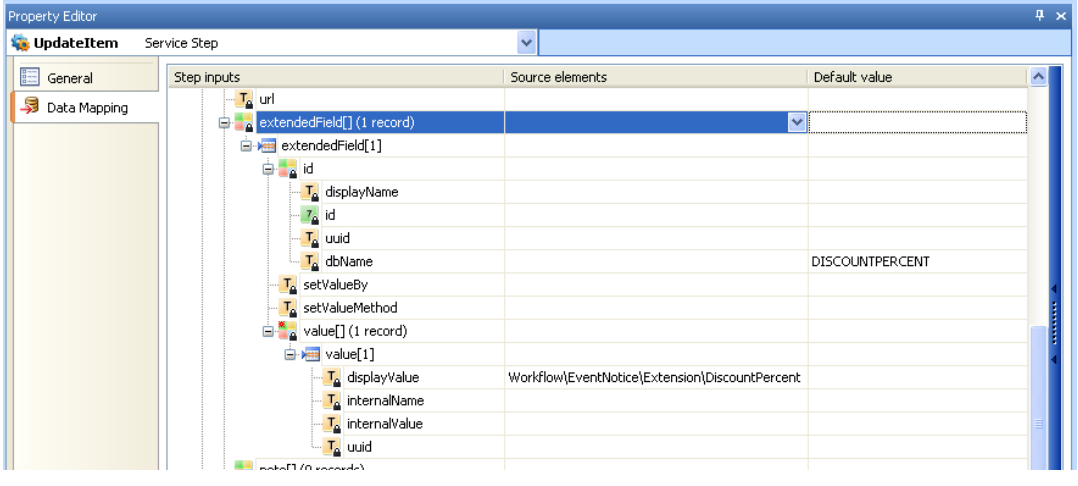


   d. Click **OK**. The path is added to the **Source elements** column.



e. Right-click the **extendedField[]** step input, and then select **Add Array Record**. Expand **extendedField[1]**, and then type `DISCOUNTPERCENT` in the **Default value** column for the **dbName** step input. This is the database name of the field that will be updated.

f. Expand the **value[]** step input, right-click, and then select **Add Array Record**. Expand **value[1]** and then and then click in the **Source elements** column for the **displayValue** step input. In the **Select a source** tool that opens, select **DiscountPercent** under **Extension**. (This is the other custom data field you added to the custom event definition.) Click **OK**. The path is added to the **Source elements** column.



10. Deploy the process app.

11. Use the exported `.wsdl` file to create a SOAP request to raise an external event. You can use any third-party product that can create and send SOAP requests to do this. If you are an advanced user, you can alternatively create the SOAP request manually and then send it in an e-mail message. For more information, see Raising an External Event through E-mail [page 590].

Chapter 29: Orchestration Tutorials

The exercises in this tutorial demonstrate how to use SBM Composer to create orchestrations. This tutorial takes approximately 1/2 hour to complete, and requires that you first complete the Chapter 6: Process App Tutorial [page 91], which also takes approximately 1/2 hour.

In the Orchestration tutorial, you create two synchronous orchestration workflows, and one asynchronous *orchestration workflow* [page 690] with a Web service. All three workflows are necessary for the process app to work correctly.

This tutorial contains the following steps:

Step 1: Create an Orchestration

To create an orchestration:

1. In App Explorer, right-click **MyProcessApp**, point to **Add New**, and then select **Orchestration**.
2. In the **New Orchestration** dialog box, enter `MyOrch` in the **Name** box, and then click **OK**.
3. Click **OK** to close the **New Orchestration** dialog box.

   Application Links, **Orchestration Workflows**, **Type Library**, and **Web Services** appear under the new **MyOrch** heading in App Explorer.
4. Continue to Step 2: Create a Synchrous Pre-Transition Orchestration Workflow [page 564].

Step 2: Create a Synchrous Pre-Transition Orchestration Workflow

In this exercise, you use the **Action Wizard** to create the *SynchBefore* orchestration. SynchBefore is a synchronous pre-transition *orchestration workflow* [page 690]. When SynchBefore is invoked from Submit transition, it updates (adds text to) the **Title** box of the transition page before the page loads.

To create a pre-transition orchestration workflow with reply:

1. In App Explorer, under the **Application Workflows** heading, select **MyAppWorkflow**.
2. In the application workflow editor, right-click the **Submit** transition, and select **Show Actions**.
3. On the **Actions** tab of the Property Editor, click **New.**

   The **Action Wizard** asks, "Which type of action do you want to execute?"
4. From the list of *action* [page 679] types in the **Step 1** box, select **Orchestration Workflow**.
5. In the **Step 2** box, click the **and continue executing** link, select **and wait for reply** from the list, and then click **Next**.

   The **Action Wizard** asks, "When do you want to call the orchestration workflow?"
6. In the **Step 1** box, select **Before**, and then click **Next**.

   You do not do anything in the **Step 2** box, which reads "Invoke an orchestration workflow and wait for reply, | before this transaction occurs".

   The **Action Wizard** asks, "Which orchestration workflow do you want to invoke?"
7. On the menu under **Step 1**, select **(Add new workflow...)**.

   You do not do anything in the **Step 2** box, which still reads "Invoke an orchestration workflow and wait for reply, | before this transaction occurs".
8. In the **Event with Reply** dialog box, do the following:
   a. Change the **Name** to `MyOrchWFWR` (which is short for MyOrch Workflow with Reply).
   b. From the **Orchestration** list, select **MyOrch.**
   c. In the **Workflow** box, change the name to `SynchBefore`.
   d. In the **Fields used by event** column, select the **Title** check box.
   e. In the **Fields returned by event** column, select the **Title** check box.
   f. Click **OK.**

      You do not do anything in the **Step 2** box, which now reads "Invoke an orchestration workflow and wait for reply, | before this transaction occurs, | Call MyOrchWFWR | If form is invalid, don't rerun this service."
9. In the **Action Wizard**, without changing anything in the **Step 2** box, click **Finish**.

A new icon appears next to the transition name in the application workflow editor indicating that an action is associated with the transition. The action is also listed on the **Actions** tab of the **Property Editor**.

In addition, in App Explorer, under the **MyOrch** heading, **SynchBefore** appears under **Orchestration Workflows**, and **MyOrchWFWR** appears under the **MyApp** heading, under **Orchestration Links**.

10. In App Explorer, under **MyOrch**, under **Orchestration Workflows**, select **SynchBefore**.
11. In the **New Items** section of the **Step Palette**, drag and drop a **Calculate** step onto the line between the **Start** and **End** steps.
12. On the **Options** tab of the Property Editor, in the **Target** section, enter the following expression: `EventNoticeWithReply.Extension.Title`
13. In the **Expression** section, enter the following function: `STRING("Replace this text with something unique, 25 characters or less.")`

    For more information about creating expressions, see About the Expression Editor [page 442].
14. In the orchestration workflow editor, select the **End** step.
15. On the **Data Mapping** tab of the Property Editor, expand the **Extension** data element, locate the **Title** data element, select the corresponding cell in the **Source elements** column, and then click the down arrow.
16. In the **Select a Source** tool, expand **SynchBefore**, **Inputs**, **EventNoticeWithReply**, and **Extension**; select **Title**; and then click **OK.**

    SynchBefore\EventNoticeWithReply\Extension\Title appears in the **Source elements** column.
17. Continue to Step 3: Create a Synchronous Post-Transition Orchestration Workflow [page 565].

Step 3: Create a Synchronous Post-Transition Orchestration Workflow

In this exercise, you use the **Action Wizard** to create the *SynchAfter* orchestration workflow. SynchAfter is a synchronous post-transition *orchestration workflow* [page 690]. When SynchAfter is invoked from the Submit transition, it appends the text in the **Title** box of the transition page before the state form loads. The new appended text appears in the **Title** box on the state form.

To create a synchronous post-transition orchestration workflow:

1. In App Explorer, under the **Application Workflows** heading, select **MyAppWorkflow**.
2. In the application workflow editor, right-click the **Submit** transition, and then select **Show Actions**.
3. On the **Actions** tab of the Property Editor, click **New.**

   The **Action Wizard** asks, "Which type of action do you want to execute?"

4. From the list of *action* [page 679] types in the **Step 1** box, select **Orchestration Workflow**.
5. In the **Step 2** box, click the **and continue executing** link, select **and wait for reply** from the list, and then click **Next**.

   The **Action Wizard** asks, "When do you want to call the orchestration workflow?"
6. In the **Step 1** box, select **After**, do not change anything in the **Step 2** box, and then click **Next**.

   The **Action Wizard** asks, "Which orchestration workflow do you want to call?"
7. In the list under **Step 1**, select **(New workflow...)**.
8. In the **Event with Reply** dialog box that opens:
   a. Change the **Name** to "MyOrchWFWR2" (which is short for MyOrch Workflow With Reply #2).
   b. From the **Orchestration** list, select **MyOrch.**
   c. In the **Workflow** box, change the name to `SynchAfter`.
   d. In the **Fields used by event** column, select the **Title** check box.
   e. In the **Fields returned by event** column, select the **Title** check box.
   f. Click **OK.**
9. In the **Action Wizard**, click **Finish**.

   The action is listed on the **Actions** tab of the Property Editor.

   In addition, in App Explorer, under the **MyOrch** heading, **SynchAfter** appears under **Orchestration Workflows**, and **MyOrchWFWR2** appears under the **MyApp** heading, under **Orchestration Links**.
10. In App Explorer, under **MyOrch**, under **Orchestration Workflows**, select **SynchAfter**.
11. In the **New Items** section of the **Step Palette**, drag a **Calculate** step onto the line between the **Start** and **End** steps.
12. On the **Options** tab of the **Property Editor**, in the **Target** section, enter the following expression: `EventNoticeWithReply.Extension.Title.`
13. In the **Expression** section, enter the following function:
    `CONCAT(EventNoticeWithReply.Extension.Title, " Appended by SynchAfter.")`

    Note: Be sure to include the space after the comma.

    For more information about creating expressions, see About the Expression Editor [page 442].
14. In the orchestration workflow editor, select the **End** step.

15. On the **Data Mapping** tab of the **Property Editor**, expand the **Extension** data element, locate the **Title** data element, select the corresponding cell in the **Source elements** column, and then click the down arrow.
16. In the **Select a source** tool, expand **SynchAfter**, **Inputs**, **EventNoticeWithReply**, and **Extension**; select **Title**; and then click **OK.**

    SynchAfter\EventNoticeWithReply\Extension\Title appears in the **Source elements** column.
17. In App Explorer, under the **MyApp** heading, under **Orchestration Links**, select **MyOrchWFWR2**.
18. On the **MyOrchWFWR2** tab, the **Title** check boxes of the **Fields used by event** and **Fields returned by event** columns should be selected.
19. Continue to Step 4: Create an Asynchronous Orchestration Workflow [page 567].

Step 4: Create an Asynchronous Orchestration Workflow

In this exercise, you add an event without reply as a transition *action* [page 679] on the **Close** transition of **MyAppWorkflow**.

Asynch is an asynchronous orchestration workflow. When **Asynch** is invoked from the **Close** transition, it creates a new item in MyOtherAppProject. The **Title** box of the new item contains the text that was added by the **SynchBefore** and **SynchAfter** orchestration workflows and some additional text.

To create an event without reply:

1. Add a second *application workflow* [page 680] called **MyOtherAppWorkflow**:
   a. In App Explorer, right-click the **Application Workflows** heading, and then select **Add New Workflow**.
   b. On the **General** tab of the Property Editor, change the **Name** field to `MyOtherAppWorkflow`, and then press TAB.
2. In App Explorer, under the **Application Workflows** heading, select **MyAppWorkflow**.
3. In the application workflow editor, right-click the **Close** transition, and select **Show Actions**.
4. On the **Actions** tab of the Property Editor, click **New.**

   The **Action Wizard** asks, "Which type of action do you want to execute?"
5. From the list of action types in the **Step 1** box, select **Orchestration Workflow**.
6. Without changing anything, click **Next**.

   The **Action Wizard** asks, "What do you want to affect?"
7. Without changing anything, click **Next**.

   The **Action Wizard** asks, "Which condition do you want to check?"
8. Without changing anything, click **Next**.

The **Action Wizard** asks, "Which orchestration workflow do you want to invoke?"

9. In Step 1, select **(New workflow...)**.

   AssignedCloseWorkflow appears and is selected in the **Step 1** list.

10. Without changing anything in the **Step 2** box, click **Finish**.

    A new icon appears next to the transition name in the application workflow editor indicating that an action is associated with the transition. The action is also listed on the **Actions** tab of the Property Editor.

    In addition, in App Explorer, under the **MyOrch** heading, **AssignedCloseWorkflow** appears under **Orchestration Workflows** and **MyAppALFServiceFlow** appears under **Event without Reply**.

11. In App Explorer, under **Application Links**, select **MyAppALFServiceFlow**.

12. On the **General** tab in the Property Editor, change the **Name** to `MyOrchWFNR` (which is short for MyOrch Workflow No Reply), and then press the Tab key.

13. Under **MyOrch**, under **Orchestration Workflows**, select **AssignedCloseWorkflow**.

14. On the **General** tab of the Property Editor, change the **Name** to `Asynch`, and then press the Tab key.

15. In the **New Items** section of the **Step Palette**, drag and drop a **Calculate** step onto the line between the **Start** and **End** steps.

16. On the **Options** tab in the Property Editor, in the **Target** section, enter the following expression: `EventNotice.Extension.Title.`

17. In the **Expression** section, enter the following string function:
    ```
    CONCAT(EventNotice.Extension.Title," This issue was created by Asynch.")
    ```

    Note: Be sure to include the space after the comma.

    For more information about creating expressions, see About the Expression Editor [page 442].

18. In the **New Items** section of the **Step Palette**, drag a **Service** step onto the line between the **Calculate** and **End** steps.

    sbmappservices72 appears under **Web Services**, under the **MyOrch** heading.

    SBM Composer automatically adds the SBM Web Service (sbmappservices72) to new orchestrations. (For more information, see the *Serena® Business Manager Web Services Developer's Guide*.)

19. On the **General** tab of the Property Editor, from the **Service** list, select **sbmappservices72**. You can also add another Web service by selecting **(add new service...)** on the **Service** menu. In the **Web Service Configuration** dialog box that opens, select the Web service's WSDL file on the **WSDL** menu or click the browse button.

20. From the **Operation** menu, select the **CreatePrimaryItemWithName** Web service operation.
21. On the **Data Mapping** tab, locate the **fullyQualifiedProjectName** data element, and then select the corresponding cell in the **Default value** column.
22. In the **Default value** column, enter the following text: `MyOtherAppProject`. This is the name of the project into which the new item will be submitted.
23. Expand the **item** data element, locate the **title** data element, select the corresponding cell in the **Source elements** column, and then click the down arrow.
24. In the **Select a source** tool, expand **Asynch**, **Inputs**, **EventNotice**, and **Extension**; select **Title**; and then click **OK.**

    Note: If you are not using Single Sign-On (SSO), you might need to map the **auth** data element also. (For more information about authentication and SSO, see the *Serena® Business Manager Web Services Developer's Guide*.)
25. Continue to Step 5: Validate the Process App [page 569].

Step 5: Validate the Process App

SBM Composer validates a process app before publishing it, but you can also validate a process app at any time by performing steps 1 and 2 in the following exercise.

Tip: While you are creating orchestration workflows, you should validate often to catch potential problems before you attempt to *publish* [page 692] and deploy a process app.

To validate a process app:

1. On the **Deployment** tab of the Ribbon, click the **Validate** button.

   If the process app is ready to be published, a message box opens advising you that the process app passed validation and that there are no errors, warnings, or messages.
2. Click **OK**.
3. To see what a validation error looks like:

   a. In App Explorer, under the **MyOrch** heading, under **Orchestration Workflows**, select any of the orchestration workflows.

   b. In the **New Items** section of the **Step Palette**, drag a **Calculate** step onto the orchestration workflow editor, and drop it anywhere on the line between the **Start** and **End** steps.

   c. Click **Validate** again.

      A message box opens advising that the validation failed and showing the number of any errors, warnings, or messages.

   d. Click **OK**.

      Details about the errors, warnings, and messages appear in the Message List. If the Message List is not available, on the **Home** tab of the Ribbon, in the **Common Views** area, select the **Message List** check box. If this check box is

already selected and the details are still not visible, select the **Message List** tab in the area under the editor pane.

For more information about the Message List, see Using the Message List [page 411]. For information about validation errors, see Troubleshooting Orchestrations Using the Message List [page 573].

4. To return the *orchestration workflow* [page 690] to its valid condition, delete the **Calculate** step that you just added.
5. On the **Deployment** tab of the Ribbon, click the **Validate** button again.

   The message box that opens should show that the process app passed validation.
6. Click **OK**.
7. Continue to Step 6: Publish the Process App [page 570].

Step 6: Publish the Process App

To *publish* [page 692] a process app:

1. Click the Composer button, and then click **Publish**.
   - If the process app was not saved yet, or if any part of the process app changed since the last time it was saved, a message box opens reminding you to save the changes. Click **OK**.
   - If any of the design elements in the process app are not checked in to the SBM Application Repository, a message reminds you to check them in. Click **OK**.
2. When the **Check In Design Elements** dialog box opens, you can enter an optional comment in the **Comment** box.
3. Click **OK**.

   The **Publish Process App** dialog box opens.
4. You can enter an optional comment in the **Comment** box.
5. In the **Label** box, you can modify the text that identifies this version of the process app.
6. Under **Visibility**, select the **Allow others to deploy this process app** check box.
7. Click **OK**.

   A message box opens indicating whether the process app published successfully or the operation failed. (See Troubleshooting Orchestrations Using the Message List [page 573] for information about publish operation errors.)
8. Click **OK**.
9. Continue to Step 7: Deploy the Process App [page 570].

Step 7: Deploy the Process App

To deploy a process app:

1. On the **Deployment** tab of the Ribbon, click the **Deploy** button.

The **Deploy Process App** dialog box opens.

2. From the **Environment** list, select a runtime environment, and then click **Deploy**.

   In the Message List, a message appears notifying you that the *deployment* [page 683] started successfully.

3. Wait for the deployment to complete.

   - If the deployment succeeds, the following message appears in the Message List: "Deployment of 'MyProcessApp' has completed."
   - If the deployment fails, the following message appears in the Message List: "Deployment of 'MyProcessApp' has aborted or failed."

   If the Message List is not available, on the **Home** tab of the Ribbon, in the **Common Views** area, select the **Message List** check box. If this check box is already selected and the details are still not visible, select the **Message List** tab in the area under the editor pane.

   For more information about the Message List, see Using the Message List [page 411]. For information about deployment errors, see Troubleshooting Orchestrations Using the Message List [page 573].

4. Continue to Step 8: Run the Process App [page 571].

Step 8: Run the Process App

In this exercise, you test **MyProcessApp** by running it in the SBM User Workspace.

To run a process app in the SBM User Workspace:

1. Log on to the SBM User Workspace.

   Tip: You can click the **User Workspace** button on the **Launch** tab of the Ribbon to do this.

2. Select the **MyApp** tab.

   If the **MyApp** tab is not visible, click the **More** tab, and then select **MyApp** in the list.

3. On the **Task Page**, under **Submit**, click the **Submit a new MyApp** link.

4. In the **Submit Tree**, click the **MyAppProject** link.

   The first transition page opens, and the title contains the text that you specified in the **Calculate** step of the **SynchBefore** *orchestration workflow* [page 690]: `Replace this text with something unique, 25 characters or less.`

5. Delete the text in the **Title** box, enter some unique text (fewer than 25 characters).

6. Select a user from the **Manager** list, and then click **OK**.

   The first state form opens, and the title now contains the text you entered in the previous step followed by the text that you specified in the **Calculate** step of the **SynchAfter** orchestration workflow: `Appended by SynchAfter.`

7. Click the **Assign** button.

The second *transition form* [page 700] opens.

8. Select a user from the **Employee** list, and then click **OK**.
9. Click **Close** and then click **OK** on the page that opens. The **Asynch** orchestration workflow is invoked, and it creates a new item in MyAppProject.
10. To locate the new item, click the **Search** filter.
11. In the **Projects** field, select **MyOtherAppProject.**
12. Click **Search.**

    Tip: You can search for all items in the **MyApp** *application* [page 679] by typing an * in the search box.

13. In the **Search Results** list, find the new item.

    The **Title** column contains the following: `[Your unique text] Appended by SynchAfter. This issue was created by Asynch.`

14. To view the item, click the link in its **Item Id** column.

    The new item appears in the area under the list.

Orchestration Reminder List

As you begin creating your own orchestration workflows, keep these points in mind:

- Make sure you have access to SBM Composer, the SBM User Workspace, and the SBM System Administrator.
- If you plan to use *Web services* [page 701], make sure you can specify the location for each Web service's WSDL file.
- Make sure that the process app you created or selected contains an *application* [page 679], that it has a workflow defined, and that the workflow contains at least one transition between states.
- Before creating an *orchestration workflow* [page 690], make sure you know what data must be passed from the application to the orchestration workflow.

Chapter 30: Troubleshooting Orchestration Workflows

This section contains the following topics about the resources and tools that you can use to troubleshoot orchestration workflows. You can also use the **Scope**, **Throw**, and **Compensate** steps to help you troubleshoot orchestration workflows (see Using the Scope, Throw, and Compensate Steps to Handle Faults From Web Services [page 498].

- Troubleshooting Orchestrations Using the Message List [page 573]
- Troubleshooting Orchestrations Using the Log Viewer [page 573]
- Troubleshooting Orchestrations Using Error Messages [page 578]
- Limitations on WSDL Files [page 578]
- Debugging for Development and Support [page 579]

Troubleshooting Orchestrations Using the Message List

You can use the Message List to troubleshoot orchestration workflows before you *publish* [page 692] and deploy them. To learn how to use the Message List, go to Using the Message List [page 411].

Following are some typical situations that generate error and warning validation messages for orchestration workflows:

- You did not map a required data element or provide a default value for it in the **Select a Source Tool**. (Error)
- You failed to enter a required expression or you entered an invalid expression, for example, in the **Target** section of a **Calculate** step or in the **Rule** section of a **While** step. (Error)
- You did not provide a compensation handler for a scope. (Warning)
- You did not specify a default value for a working data element. (Warning)

Troubleshooting Orchestrations Using the Log Viewer

This section demonstrates how to work with Log Viewer messages to troubleshoot, or debug, orchestration workflows. You should already know how to use the Log Viewer, which is explained in Using the Log Viewer [page 415].

Note: Unhandled Web service faults are the most common causes of *orchestration workflow* [page 690] failures. (See Using the Scope, Throw, and Compensate Steps to Handle Faults From Web Services [page 498].)

If a process app that contains an orchestration workflow does not perform properly, there might be a problem with the orchestration workflow. You can use the Log Viewer to

troubleshoot orchestration workflow-related runtime errors. The entries in the Log Viewer provide an audit trail that you can use to diagnose these problems.

For example, you can determine why incorrect values were copied to working data elements or **Service** step source elements during the execution of **Calculate** steps. To do this, turn Debug Logging on, and select **User messages only** in the Message Filter Dialog Box [page 420]. The messages, which begin with "copying value," will show the actual values that the **Calculate** steps copied.

Note: If a **Calculate** step is copying an XML element, the message will show the value in XML format.

Web Service Faults

External *Web services* [page 701] are called by **Service** steps in orchestration workflows by sending request SOAP messages. (To the SBM Orchestration Engine, the SBM Application Engine is an external Web service.) Web services also reply with SOAP messages. If an error occurs during the processing of a request, a response SOAP message is returned that contains a SOAP fault. Because orchestration workflows are actually Web services that are invoked by other SBM components, they send and receive SOAP messages as well. (See About SOAP Messages [page 445].)

In SBM, the SBM Application Engine invokes a synchronous *orchestration workflow* [page 690] with a SOAP message that is sent to the SBM Orchestration Engine. The SBM Orchestration Engine returns a response SOAP message to the SBM Application Engine at the **End** step.

The SBM Application Engine or an external event invokes an asynchronous orchestration workflow with a request SOAP message that it sends to the *Event Manager* [page 685]. The Event Manager relays the message to the SBM Orchestration Engine. For asynchronous orchestration workflows, no response is returned.

During the execution of an orchestration workflow, SOAP messages returned from an external Web service might contain a SOAP fault. The first step in debugging an orchestration workflow using the Log Viewer is to look for errors that indicate a SOAP fault. The last error that occurred in the orchestration workflow sequence usually contains the most useful information.

Following are some typical situations that generate error messages for orchestration workflows in the Log Viewer:

- You tried to pass an invalid value to the Web service.
- You did not specify the correct path for the WSDL.
- You failed to enter a required expression or you entered an invalid expression, for example, in the **Target** section of a **Calculate** step or in the **Rule** section of a **While** step.

Debugging Orchestration Workflows

This section presents the steps that you should take to debug the orchestration workflows in a process app using the Log Viewer and includes an example of a problem with a **Service** step. The example demonstrates one of the most common errors that cause the execution of an *orchestration workflow* [page 690] to stop. This problem occurs when a Web service returns a response SOAP message that contains a SOAP fault during execution of the workflow.

Note: If you are using the Log Viewer correctly and you don't see any messages for the orchestration workflow that you are debugging, the workflow might not have been invoked. To check for this problem, you should look at the messages for the associated *application workflow* [page 680].

To debug orchestration workflows using the Log Viewer:

1. In the Log Viewer, click **Refresh** to see the latest messages.
2. On the **Overview** tab, look for errors in any of the orchestration workflows invoked by the application workflow (project) that you submitted to.

   Errors are indicated by numbers in the first position within the brackets next to the name of the application workflow. For example, `UserVerificationOWF [1/0/0/7]` indicates that there is one error messages, no warning messages, no info messages, and seven debug messages in the Log Viewer for this orchestration workflow.

   Note: If there are no error messages, you should also look at any other messages. Find the message with the most severe logging level that is the farthest in the orchestration workflow sequence.
3. Select the orchestration workflow in the list, and then click the **Details** tab.

   In the example, you would click `UserVerificationOWF`.
4. Locate any error messages on the list.

   You would usually look for the error that occurred during the latest run, that is, the one with the highest number.

   In the example, the following series of error message is displayed. (By default, all Log Viewer messages appear in this format.) From the messages, you can see that something went wrong with **Service** step `VerifyUser` when the orchestration workflow that contains it was executed.

Date	Run	Associated Item	Text
2/20/2…	1	VerifyUser	A Web service is being invoked at Service step VerifyUser, and the…
2/20/2…	1	VerifyUser	A Web service was invoked at Service step VerifyUser, and now the…

Date	Run	Associated Item	Text
2/20/2…	1	VerifyUser	A fault occurred during the execution of the orchestration workflo…

5. Right-click the error message row, and then select **Show Message** on the menu.
6. The **Message Details** dialog box opens and displays the entire error message. (Steps 7 and 8 relate to Service steps only.)

   Following is the entire text of the example error message:

   ```
   A fault occurred during the execution of the orchestration workflow at
   Service step VerifyUser. To identify the fault, find the "A Web service was
   invoked" message listed just before this message. View the message (which was
   returned by the Orchestration Engine). Next, find the SOAP fault, which
   begins with "<SOAP-ENV:Fault." To handle a SOAP fault, you can place the
   Service step that calls the Web service inside a Scope step and catch the
   fault in a Catch branch of the FaultHandler. For more information about SOAP
   faults and how to handle them, see the "Working with Orchestrations" section
   of the documentation. [org.jbpm.bpel.integration.client.SoapClient]
   ```

7. Perform the next step in the message instructions. To complete the example, the step is repeated here. In the **Message Details** dialog box, click the **Previous** button to display the `"A Web service was invoked"` message, and then locate the SOAP fault.

   Following is the entire text of the previous (earlier) example message. The SOAP fault is indicated by bold text. Note that the message indicates an invalid User ID or Password.

   ```
   A Web service was invoked at Service step VerifyUser, and now the
   Orchestration Engine is receiving the following message:

       <SOAP-ENV:Envelope
       xmlns:SOAP-ENC='http://schemas.xmlsoap.org/soap/encoding/'
       xmlns:SOAP-ENV='http://schemas.xmlsoap.org/soap/envelope/'
       xmlns:ae='urn:sbmappservices72'
       xmlns:c14n='http://www.w3.org/2001/10/xml-exc-c14n#'
       xmlns:ds='http://www.w3.org/2000/09/xmldsig#'
       xmlns:wsse='http://docs.oasis-open.org/wss/2004/01/oasis-2004
       01-wss-wssecurity-secext-1.0.xsd'
       xmlns:wsu='http://docs.oasis-open.org/wss/2004/01/oasis-20040
       1-wss-wssecurity-utility-1.0.xsd'
       xmlns:xsd='http://www.w3.org/2001/XMLSchema'
       xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'><SOAP-E
       NV:Header></SOAP-ENV:Header><SOAP-ENV:Body><SOAP-ENV:Fault
       xmlns:SOAP-ENV='http://schemas.xmlsoap.org/soap/envelope/'><f
       aultcode>SOAP-ENV:Server</faultcode><faultstring>Invalid User
       ID or Password</faultstring><detail><ae:AEWebservicesFault
       xmlns:ae='urn:sbmappservices72'>Invalid User ID or
       Password</ae:AEWebservicesFault></detail></SOAP-ENV:Fault></S
   ```

```
OAP-ENV:Body></SOAP-ENV:Envelope>[org.jbpm.bpel.integration.s
oap.SerenaSoapUtil]
```

8. Often a SOAP fault is returned because some invalid data was sent to the Web service. To see the data that was sent by the Service step, click **Previous** to view the previous message.

   To see the invalid User ID or Password that was sent by the **VerifyUser** Service step call to the SBM Application Engine, click **Previous** to view the previous message.

   Following is the entire text of the previous (earlier) example message. The values that were sent with the Web service call are indicated by bold text.

   ```
   A Web service is being invoked at Service step VerifyUser, and the
   Orchestration Engine is sending the following message:

   <env:Envelope
   xmlns:env='http://schemas.xmlsoap.org/soap/envelope/'
   xmlns:xsd='http://www.w3.org/2001/XMLSchema'><env:Header><n:S
   ecurity SOAP-ENV:mustUnderstand='0'
   xmlns:n='http://docs.oasis-open.org/wss/2004/01/oasis-200401-
   wss-wssecurity-secext-1.0.xsd'
   xmlns:SOAP-ENV='null'></n:Security></env:Header><env:Body><de
   faultNS1:IsUserValid
   xmlns:bpws='http://schemas.xmlsoap.org/ws/2003/03/business-pr
   ocess/' xmlns:defaultNS='urn:sbmappservices72'
   xmlns:defaultNS1='urn:sbmappservices72'
   xmlns:ns8='urn:sbmappservices72'><ns8:auth><ns8:userId>ZZZZ</ns
   8:userId><ns8:password>zzzz</ns8:password></ns8:auth><ns8:log
   inId xmlns:SOAP-ENC='http://schemas.xmlsoap.org/soap/encoding/'
   xmlns:SOAP-ENV='http://schemas.xmlsoap.org/soap/envelope/'
   xmlns:ns='http://www.eclipse.org/alf/schema/EventBase/1'
   xmlns:s='http://www.eclipse.org/alf/schema/EventBase/1'
   xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
   xsi:type='xsd:string'
   xmlns:xsd='http://www.w3.org/2001/XMLSchema'>XYZ</ns8:loginId
   ></defaultNS1:IsUserValid></env:Body></env:Envelope>[org.jbpm
   .bpel.integration.soap.SerenaSoapUtil]
   ```

9. To go to the source of the problem, right-click in the error message row, and then select **Go to Location** on the menu.

   SBM Composer opens the appropriate *design element* [page 683] in the workflow editor and selects the design element associated with the error.

   In the example, SBM Composer opens **UserVerificationOWF** in the workflow editor and selects the **VerifyUser** Service step.

10. Correct the problem in the design element. (If the error was caused by sending invalid data, configure the Service step to provide the correct data.)

    In the example, you would enter a valid User ID and Password in the **userID** and **password** data elements under **auth**.

Troubleshooting Orchestrations Using Error Messages

This section contains a list of error messages. Each entry in the list provides the actual text of the message, an explanation of its cause, and a suggested solution. These messages are likely to appear in the Log Viewer, but they might also be displayed in error messages boxes or in other areas of SBM Composer.

`Execution of the orchestration workflow is stopped because the size of the Web service response has exceeded the allowable limit of {0} at Service step {1}.`

{0} is the maximum allowable size, and {1} contains the name of the associated **Service** step.

> **Explanation:** A Web service response is limited to a certain size in Serena Business Manager. If the response to a Web service that is invoked by means of a **Service** step in an *orchestration workflow* [page 690] exceeds this limit, the orchestration workflow is stopped. In addition, an error message containing a SOAP fault appears in the Log Viewer for the associated **Service** step. This error message also indicates the maximum allowable size for the response.
>
> **Solution:** To handle the fault, you should place the **Service** step within a **Scope** step and catch the fault in the **Catch All** branch of the **FaultHandler**. Also, you should consider reducing the size of the response SOAP message's payload by redesigning the way you invoke the **Service** step. For example, you could add filtering if the Web service supports it.

Limitations on WSDL Files

If you are having problems with the orchestration workflows in your process apps, it might be because the WSDL files you are using are not fully supported by SBM Composer. Although SBM Composer supports most WSDL files, it has some limitations. Contact your Web service provider to determine if any of the following restrictions apply:

- Only SOAP encoding may be used (not REST).
- Multi-part WSDL files are not supported.
- Recursive elements or types cannot be imported into an *application* [page 679].
- Derived complex types are not completely supported in application workflows; however, they can be used in orchestration workflows.
- Operation overloading, where the same name can apply to different operations in different situations, is not supported.
- Attribute references are not completely supported.
- WSDLs might fail to import if they are not formatted correctly, or if they contain functionality that is not supported by SBM Composer.
- The `<choice>` element is not supported.
- Creating a base type in SBM Composer and then overriding it with any of its `Extension` types is not supported.

  For example, when you create `"SearchRecord"` in the following sample code, you get a child element named `"record"` of type `"sObject"`. If this type is extended by

other types, such as "Employee" or "Hardware", you will not be able to change the type of "record" to "Employee" type on the **Data Mapping** tab of the orchestration workflow Property Editor. You can, however, create a working data element of type "Employee".

```
<complexType name="SearchRecord">
<sequence>
<element name="record" type="ens:sObject"/>
</sequence>
</complexType>
```

- Creating arrays of varying lengths as Web service inputs is not supported. For example, you can create an array on the **Data Mapping** tab of the orchestration workflow Property Editor if you know how many records to create. However, if you are unable to determine the number of records at design time, SBM cannot create such an array.

Debugging for Development and Support

Advanced options for debugging, such as setting the logging level to trace, are available in Application Administrator. Refer to the "Logging" section of *SBM Application Administrator Guide* for more information.

Part 5: Advanced Orchestration Topics

This section includes the following advanced orchestration topics:

Chapter 31: Raising External Events

Important: These instructions are intended for experienced software developers/integrators who are familiar with the SOAP standard and Extensible Markup Language (XML) and who have expert knowledge of the Web Service Description Language (WSDL) and of creating *Web services* [page 701]. A working knowledge of SBM is also required. Pertinent information is provided in Part 4: Working with Orchestrations [page 429] and in the *Serena® Business Manager Web Services Developer's Guide*, which is included with the product.

External events can be used to link a wide range of products to orchestration workflows. This allows you to enable Web service applications, in both on-premise and on-demand environments, to work with Serena Business Manager.

To raise and handle external events requires the creation of a custom *event definition* [page 685] and mapping it to your *orchestration workflow* [page 690].

The following topics describe how to create, import, and map the event definition to generate your external events:

- Events Terminology and Concepts [page 583]
- Accessing the Advanced Orchestration Package [page 585]
- Defining an Event Definition [page 585]
- Creating a Custom Event Definition [page 586]
- Testing Events from an External Source [page 588]
- Creating Event Client using Apache Axis2 [page 589]
- Raising an External Event through E-mail [page 590]
- Configuring Serena Business Manager to Receive E-Mail Events [page 594]
- Upgrading Existing Event Definitions [page 597]

Events Terminology and Concepts

This topic terminology and concepts pertaining to events.

Event

Note: An *event* [page 685] was referred to as a *mashup event* or an *ALF event* in previous releases of Serena Business Manager.

An *event* [page 685] is a *Web services* [page 701] message signaling a meaningful change from an application or tool. For example, an issue defect management application in Serena Business Manager might generate an event every time a user enters a new defect, or another product might raise an external event named `BuildCompleted`. When an event is received by the *Event Manager* [page 685], the SBM Orchestration Engine is called to execute the *orchestration workflow* [page 690] linked to the event.

Events are SOAP messages. Serena Business Manager supports both HTTP and e-mail transports for event SOAP messages. The Event Manager provides RPC/literal messaging and document/literal messaging services to accept event requests. It does not support RPC/encoded messaging.

Important: While events can be raised using either RPC/literal or document/literal messaging format, they are defined using RPC/literal messaging format (see Defining an Event Definition [page 585].)

Extensions that come with particular event types must be defined so an orchestration workflow can access the `extension` data in the event. However, the same event can also be handled by an orchestration workflow that only understands the base event.

To accomplish this, schema restriction-based derived types are defined that allow for tightening of the value range or other restrictions on certain elements in a complex type. In this way, a type that is an `ALFEventType` can be declared that only allows, for example, an `EventType` of `ThingCreated` and an `ObjectType` of either `Thing1` or `Thing2`.

Event Manager

The *Event Manager* is a component in Serena Business Manager that responds to events sent by applications and products. The Event Manager calls orchestration workflows implemented as *BPEL* [page 680] processes.

Event Map

The Event Manager dispatches events according to a configured map, called the *event map*. The event map relates events to the orchestration workflows that should run when an event occurs. Products that can raise events can declare their events by means of a specific WSDL format called an *event definition* [page 685]. (See Defining an Event Definition [page 585].) The event definition can be used by a product, such as Serena Business Manager, to construct an event map and to deploy the event map to the Event Manager. At runtime, the Event Manager receives the event, and if the event appears in a deployed event map, it invokes the associated orchestration workflow or workflows, passing on the event data.

Orchestration Workflow

An orchestration workflow is a sequenced arrangement of Web service calls designed using SBM Composer. Orchestration workflows combine Web services using loops and decision branches and define the way data is mapped between the Web services. The final arrangement is saved as a BPEL process. Orchestration workflows can be linked to application workflows in a process app through actions on both transitions and states. Orchestration workflows can run asynchronously using an event, or they can be called synchronously, where the *action* [page 679] waits for the orchestration workflow to return some data. When designed to be used with an event, orchestration workflows can also be invoked by external systems.

SBM Orchestration Engine

The SBM Orchestration Engine is the component in Serena Business Manager that executes orchestration workflows. Using SBM Composer, Web services can be sequenced and then executed in response to an event or by transitions in applications.

Object

In Serena Business Manager, an *object* is identified in the event structure by the values of `ObjectType` and `ObjectId`. In other words, the `ObjectType` and `ObjectId` are used to refer to the object that originated the event. An event typically originates due to a change

in an object. The `ObjectType/ObjectId` identifies the source of the event within the product and the `EventType/EventId` identifies the type of change that caused the event.

Accessing the Advanced Orchestration Package

The Advanced Orchestration Package contains files that you can use as a basis to create or update your *event definition* [page 685].

Before you create your *application* [page 679]-specific event definition, download the Advanced Orchestration Package `.zip` file. You will be using these files throughout the instructions to create your custom event definitions and events.

The file is available from the Start Page in SBM Composer or from http://www.serena.com/support.

Refer to the `readme.txt` file in the Advanced Orchestration Package `.zip` file for a description of the files in the package.

Defining an Event Definition

An *event definition* [page 685] used in SBM defines events from an *application workflow* [page 680]. You can create your own custom event definition for generating external events.

An event definition is a WSDL file that redeclares the ALFServiceFlow service, making it specific to your event source. The process involves creating a specialized event definition schema derived from the ALF event base schema. The event definition specifies custom, tool-specific types and events. It also defines the *orchestration workflow* [page 690] (declared as a service flow service) that handles these events.

You can create a custom event directly in SBM Composer. This works for many cases. However, for some advanced cases, you might prefer to create the event definition as a separate WSDL file and import it into SBM Composer.

Note: This chapter describes how to create the event definition as a separate WSDL file. For information about how to create the event definition in SBM Composer see Raising External Events [page 556], Creating a New Custom Event Definition [page 475], and Importing an Event Definition File for a New Custom Event Definition [page 476].

Event Definition Messaging and Schema

Event definitions must use RPC/literal messaging and must be created following the definitions in `ALFEventManager?wsdl`.

The ALF event base schema from which you create the derived schema for your event definition is defined in `ALFEventBase_1.xsd`. This schema is common to all event services and is included by reference in your event definition.

Creating a Custom Event Definition

This section provides instructions for creating a custom *event definition* [page 685]. Using the `ExampleEventDefinition.wsdl` file as a *template* [page 699], you provide values that let external products generate events.

Important: This topic provides instructions for manually creating a custom event definition. It is more convenient to create a custom event definition in SBM Composer. For more information, see Creating a New Custom Event Definition [page 475].

To create a custom event definition:

1. Create an empty folder on your local system.
2. Copy the `advanced_orchestration_package.zip` file and paste it in the folder.
3. Extract the `advanced_orchestration_package.zip` file.
   - The `ALFEventManager.wsdl` file is the ALF Event Manager WSDL. Note that the `ExampleEventDefinition.wsdl` imports the `ALFEventManager.wsdl`.
   - The `ALFEventBase_1.xsd` file is the ALF event base schema.
   - `ExampleEventDefinition.wsdl` is the template for your custom event definition. The other files are explained in other topics.

     Note: You can also obtain a copy of the ALF Event Manager WSDL and the `ALFEventBase_1.xsd` files from the `installDir\Composer\Conf` folder.
4. Create a copy of the `ExampleEventDefinition.wsdl` file in an XML editor and rename it as your external product WSDL file, for example, `SCMProduct.wsdl`. If you ever move this file to a different folder, be sure to place a copy of the `ALFEventManager.wsdl` and the `ALFEventBase_1.xsd` files in the folder as well.
5. Edit this new file to modify it for your installation:
   a. Look for following section for event type declaration. Change the value attribute of the enumeration to your event type. Each enumeration value corresponds to different event occurring in your *application* [page 679].

```
<xs:simpleType name="MyEventDefinitionEventType">
<xs:restriction base="EventTypeType">
<xs:enumeration value="MyEventDefinitionEvent"/>
</xs:restriction>
</xs:simpleType>
```

   You can define as many enumeration values as you want, for example:

```
<xs:simpleType name="MyEventDefinitionEventType">
<xs:restriction base="EventTypeType">
<xs:enumeration value="Issue Created"/>
<xs:enumeration value="Issue Deleted"/>
<xs:enumeration value="Waiting for Approval"/>
```

```
</xs:restriction>
</xs:simpleType>
```

b. Look for the following section for object type declaration. You can define as many enumeration values as you want. However, it is recommended that you define only one object type. Change the enumeration value to your external product specific object.

```
<xs:simpleType name="MyEventDefinitionObjectType">
<xs:restriction base="ObjectTypeType">
<xs:enumeration value="MyEventDefinitionObject"/>
</xs:restriction>
</xs:simpleType>
```

c. Look for product value, product instance and product version declarations. *Product value* indicates the external product name. *Product version* is the number you want to give this event declaration. You can use this value to distinguish the events between versions of a product if the event definitions are changed in an incompatible way. If the events are completely compatible, however, you should in general not change this value. *Product instance* is the "logical location," and is determined by the particular installation of the external product in your system. It can default to what is specified in the event definition, but you can also use SBM Composer to indicate the external product from which you are expecting the event.

```
<xs:simpleType name="MyEventDefinitionProductType">
    <xs:restriction base="ProductType">
      <xs:enumeration value="MyEventDefinition"/>
    </xs:restriction>
  </xs:simpleType>

  <!-- Derived ProductVersionType -->
  <xs:simpleType name="MyEventDefinitionProductVersionType">
    <xs:restriction base="ProductVersionType">
      <xs:enumeration value="1.0"/>
    </xs:restriction>
  </xs:simpleType>

  <!-- Derived ProductInstanceType -->
  <xs:simpleType name="MyEventDefinitionProductInstanceType">
    <xs:restriction base="ProductInstanceType">
      <xs:enumeration value="MyEventDefinitionInstance"/>
    </xs:restriction>
  </xs:simpleType>
```

Again in all three cases, replace the enumeration values to match the values for the external product.

Note: The event type, object type, product, product instance, and product version values make a set of events match, and are used to determine which orchestration workflows need to be invoked.

d. Look for the section where extension data is defined. Here you can define different fields you want to send as input to your orchestration workflows. Remember you can send only one set of data using one process app tool.

```
<xs:complexType name="MyEventDefinitionCustomExtension">
    <xs:annotation>
      <xs:documentation>
        Custom Extension
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="MyEventDefinitionData" type="xs:string"/>
    </xs:sequence>
    <xs:anyAttribute/>
  </xs:complexType>
```

You can define any type of schema under sequence either simple type elements as shown above or complex type.

e. Replace all remaining occurrences of "MyEventDefinition" with your product value.

6. At this point, your event definition WSDL is ready to be imported it into SBM Composer.

Testing Events from an External Source

After deploying your process app, you can test the external event by using `sample event xml` provided in `ExampleEventDefinitionSOAPMessage.xml` using tools like SoapUI. This procedure is optional.

If you do not want to perform this test, go directly to Creating Event Client using Apache Axis2 [page 589].

To test the event from an external source, perform the following steps:

1. Create new project in soapUI.
2. Import your *event definition* [page 685] WSDL. For information on creating a custom event definition, see Creating a Custom Event Definition [page 586].
3. SoapUI will generate various request XMLs for you. Choose the one that matches your application's ALFServiceFlowSOAP request. This will be displayed as `<your product value>ALFServiceFlowSOAP`.
4. Open the corresponding request in the SoapUI editor and copy paste the `<Base>` element from `ExampleEventDefinitionSOAPMessage.xml` in it.
5. Replace the event match values—namely EventType, ObjectType, Product, ProductInstance, and ProductVersion—to match your *application* [page 679]-specific values.
6. Fill in your extension data.
7. Make sure that you set the correct *endpoint* [page 684] pointing to the *Event Manager* [page 685]. The URL should like this:

`http://<server>:<port>/eventmanager/services/ALFEventManager`

8. Raise the event by running this request. You can read about the execution in the common logger view in SBM Composer.

Creating Event Client using Apache Axis2

This topic gives an example of using Apache Axis2 to create your services. Apache Axis2 is an enterprise-ready Web service engine that is very user friendly and provides Web service interactions with a dynamic and flexible execution framework. For more information, refer to `http://ws.apache.org/axis2/`.

To create a client for raising events using Doc/Lit binding:

1. Create a new *application* [page 679]-specific WSDL file.

   a. Make a copy of the `ExampleEventDefinitionALFEventManagerDocLit.wsdl` file. This will become your application-specific `.wsdl` file.

   b. Copy both `<xsd:schema>` elements from your application-specific *event definition* [page 685] `.wsdl` file and paste the schema into this new `.wsdl` file where the following comment is:

      ```
      <!-- Paste schema from your event definition wsdl here -->
      ```

      In other words, open the custom event definition file that you created in Creating a Custom Event Definition [page 586] and copy the schema elements into this file. When finished, the `.wsdl` file created from `ExampleEventDefinitionALFEventManagerDocLit.wsdl` will contain three schema elements.

   c. Replace all occurrences of `MyEventDefinition` with your product value.

   Your application-specific `.wsdl` file is ready for consumption by your application.

2. If you are using Axis2 to raise external events, modify the batch file `GenerateStubClassesFromAxis2.bat` and change the name of the `.wsdl` file. Remember to set the Axis2 home.

3. Once the classes are generated, you can use the stub class to raise events.

4. The URL for this *endpoint* [page 684] should in the form `http://<server>:<port>/eventmanager/services/ALFEventManagerDocLit`.

Raising an External Event through E-mail

This section contains general instructions for raising external events through e-mail using the `ExampleEventDefinitionSOAPMessage_forEmail.xml` *template* [page 699], which is included in the Advanced Orchestration Package. For details on creating a sample e-mail *event* [page 685] message, see Creating a Sample E-mail Event SOAP Message [page 591].

To raise an external event through e-mail, perform the following steps:

1. Open the `ExampleEventDefinitionSOAPMessage_forEmail.xml` file in an XML editor. The template file is included in the Advanced Orchestration Package. (The contents of the template are shown in the next section.)
2. Provide values for the following `Body` elements. The values in italics are based on the definitions in your *event definition* [page 685]:
   - `EventId`
   - `Timestamp`
   - *`EventType`*
   - *`ObjectType`*
   - `ObjectId`
   - *`Product`*
   - `ProductVersion`
   - *`ProductInstance`*
3. Define the data elements that you will send with the event in the `Extension` element.
4. Provide your authentication credentials in one of the following ways:
   - Use the `ns:ALFSecurity` element.

     This method is shown in the template.
   - Use a WS-Security header.
5. Specify the recipient's e-mail address in the `wsa:To` element.
6. To send the event XML in the body of the e-mail, copy and paste the contents of the XML file into the body of the e-mail, and then send it to the address in the `wsa:To` element.

   Important: Be sure that your e-mail client sends the e-mail as plain text. If other formats are used, the e-mail will be sent as a multi-part MIME type and the event will not be recognized. If plain text cannot be used, then send the event XML as an e-mail attachment (see the next step).

   Important: Be sure that your e-mail client does not insert extra characters, such as line endings, when you send the e-mail.

7. To send the event XML as an e-mail attachment, attach the XML file to the e-mail, and then send it to the address in the `wsa:To` element.

Important: The attached file must have the `.xml` file extension and contain only the event XML text.

Creating a Sample E-mail Event SOAP Message

This section contains instructions for creating a sample e-mail event SOAP message using the `ExampleEventDefinitionSOAPMessage_forEmail.xml` *template* [page 699].

The contents of the template are included here for your reference.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsa="http://www.w3.org/2005/08/addressing"
xmlns:ns="http://www.eclipse.org/alf/schema/EventBase/1"
xmlns:myev="http://www.example.org/MyEventDefinitionEventExtensions">
  <soapenv:Header>
    <wsa:To>mailto:eventemail@serverName?X-Service-Path=/eventmanager
    → /services/ALFEventManagerOneWayDocLit</wsa:To>
    <wsa:MessageID>urn:uuid:421A2EE66BA3A42A8C1203350641340</wsa:MessageID>
    <wsa:Action>urn:EventNotice</wsa:Action>
  </soapenv:Header>
  <soapenv:Body>
    <ns:ALFEventNoticeDoc version="1.0">
      <ns:Base>
        <ns:EventId>1</ns:EventId>
        <ns:Timestamp>2008-05-12T21:19:20.671Z</ns:Timestamp>
        <ns:EventType>MyEventDefinitionEvent</ns:EventType>
        <ns:ObjectType>MyEventDefinitionObject</ns:ObjectType>
        <ns:ObjectId>1</ns:ObjectId>
        <ns:Source>
          <ns:Product>MyEventDefinition</ns:Product>
          <ns:ProductVersion>1.0</ns:ProductVersion>
          <ns:ProductInstance>MyEventDefinitionInstance</ns:ProductInstance>
        </ns:Source>
        <ns:User>
          <ns:ALFSecurity>
            <ns:UsernameToken>
              <ns:Username>username</ns:Username>
              <ns:Password>password</ns:Password>
            </ns:UsernameToken>
          </ns:ALFSecurity>
        </ns:User>
       </ns:Base>
      <ns:Extension>
        <myev:MyEventDefinitionData>Test Data</myev:MyEventDefinitionData>
      </ns:Extension>
    </ns:ALFEventNoticeDoc>
  </soapenv:Body>
</soapenv:Envelope>
```

Note: The element prefixes (for example, `wsa:`) used in the rest of this section are simply shortcut references to the defining namespace. They are provided to help you identify the values in the template, but the prefixes are not part of the element name.

To create a sample e-mail event SOAP message, you must update all of the bolded values in the template as follows:

1. Set the following element values for the particular system you are using:
   - `<wsa:To>`**`mailto:event@yourdomain.com?X-Service-Path=/eventmanager/services/ALFEventManagerOneWayDocLit`**`</wsa:To>`

The WS-Addressing `<To>` element value must contain a valid `xs:anyURI` in the format shown. The e-mail portion of the URI, in this case `eventemail@yourdomain.com`, should be set to the e-mail address that you have been provided for sending events to the system.

Note: In the on-demand environment, the e-mail address you should use is usually `event@yourdomain.com`. In other words, you should replace `eventemail@serverName` in the template with `event@yourdomain.com`.

- `<ns:Username>`**`username`**`</ns:Username>`
- `<ns:Password>`**`password`**`</ns:Password>`

  The values of the preceding two elements should be set to the user name and password credentials for the system that you use for raising the event.

2. Set the following element values per event type, as defined in your *event definition* [page 685] WSDL:
   - `<ns:EventType>`**`MyEventDefinitionEvent`**`</ns:EventType>`

     Set this element to the `EventType` value for this event.
   - `<ns:ObjectType>`**`MyEventDefinitionObject`**`</ns:ObjectType>`

     Set this element to the `ObjectType` value for this event.
   - `<ns:Product>`**`MyEventDefinition`**`</ns:Product>`

     Set this element to the `Product` value for your event definition.
   - `<ns:ProductVersion>`**`1.0`**`</ns:ProductVersion>`

     Set this element to the `ProductVersion` value for your event definition.
   - `<ns:ProductInstance>`**`MyEventDefinitionInstance`**`</ns:ProductInstance>`

     Set this element to the `ProductInstance` value for your event definition.
3. Set the following element values for each event instance:
   - `<wsa:MessageID>`**`urn:uuid:421A2EE66BA3A42A8C1203350641340`**`</wsa:MessageID>`

     Set this element to a unique ID for each event sent. A UUID is suggested. If you use `ALFEventManagerOneWayDocLit` in the `<wsa:To>` element, this value is not critical since no reply is expected; however, it might be useful in the future.
   - `<ns:EventId>`*`1`*`</ns:EventId>`

     Set this element to a unique ID for each event sent. Although this is not critical, it allows you to identify the particular instance of the event, which might be useful to correlate data.
   - `<ns:ObjectId>`*`1`*`</ns:ObjectId>`

     This element identifies the instance of the object responsible for the event. Although this is not critical, a value recognized by your event definition might be useful for accessing data in your event definition to correlate data.

- `<ns:Timestamp>`**`2008-05-12T21:19:20.671Z`**`</ns:Timestamp>`

  This element records the time the event was sent. It must be in `xsd:dateTime` format (a subset of ISO 8601). It can be set to `nil` as shown, where `xsi:` is the namespace `xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"`.

  ```
  <ns:Timestamp xsi:nil="true"/>
  ```

- `<myev:MyEventDefinitionData>`**`Test Data`**`</myev:MyEventDefinitionData>`

  This element is the `Extension` data for this event, as defined by your event definition. The sample event definition only defines one element (`MyEventDefinitionData`) in its `Extension` data. Your event definition can send more complex data.

Configuring Serena Business Manager to Receive E-Mail Events

The Web service–based event mechanism exposed by SBM can accept events either through HTTP or via e-mail. Events through HTTP are enabled by default and require no additional configuration beyond the basic installation of SBM. Events through e-mail require that SBM be configured to access an e-mail mailbox provided by an external e-mail server. The SBM SBM Configurator provides options to configure this feature.

Setting Up a POP3 E-mail Account

The SBM *Event Manager* [page 685] can support receiving events only from a POP3-compliant e-mail server. To use the e-mail event feature, you must set up a dedicated e-mail account on the POP3 server, establishing the e-mail address that you want to use to accept the e-mail event requests. To configure SBM, you need to know the POP3 server host name, the port used by your POP3 server, the e-mail account name, and the e-mail account password.

Important: Be sure that your e-mail client sends the e-mail as plain text instead of a multi-part MIME type.

Options Provided by the SBM Configurator

The SBM Configurator provides the following fields to collect the values necessary to configure the SBM Event Manager to receive e-mail events. If these values are configured correctly during configuration, the e-mail event feature will be enabled.

Host: The network name of the POP3 e-mail server you want to use to provide the e-mail inbox for the e-mailed events. The installer presents the network name of the server on which SBM is being installed as a default. However, it is quite likely that your e-mail server resides on a different machine.

Port: The port used by the POP3 e-mail server. POP3 typically uses port 110.

User name: The e-mail account name for the inbox that your POP3 server provides.

Password: The password to the e-mail account.

Adjusting the Configuration

If you want to adjust the configuration settings for e-mailed events after you have completed the initial mail server set up, you can change the settings in the SBM Configurator at any time.

Important: If you apply changes while the SBM Configurator runs in **utility mode**, browser users may not be able to access the system immediately while the services are restarting. Therefore, consider applying configuration changes at a time when users are not actively using the system.

E-mail Event Messages

To send an event by e-mail, you must construct an XML SOAP document in the correct format and send it to the *event-enabled* [page 685] e-mail account.

Here is an example e-mail event message document:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:ns="http://www.eclipse.org/alf/schema/EventBase/1"
  xmlns:exam="http://www.eclipse.org/ALF/ExampleVocabulary"
  xmlns:myev="http://www.example.org/MyEventExtensions"
  xmlns:wsa="http://www.w3.org/2005/08/addressing"
  xmlns:mym="http://www.example.org/MyEventDefinitionEventExtensions">
  <soapenv:Header>
    <wsa:To>mailto yourEmailUser@yourEmailHost?X-Service-Path=/eventmanager/
→services/ALFEventManagerOneWayDocLit</wsa:To>
    <wsa:MessageID>urn:uuid:421A2EE66BA3A42A8C1203350641340</wsa:MessageID>
    <wsa:Action>urn:EventNotice</wsa:Action>
  </soapenv:Header>
  <soapenv:Body>
    <ns:ALFEventNoticeDoc version="1.0">
      <ns:Base>
        <ns:EventId>421A2EE66BA3A42A8C1203350641340</ns:EventId>
        <ns:Timestamp>2008-10-17T09:59:51.328-08:00</ns:Timestamp>
        <ns:EventType>MyEventDefinitionEvent</ns:EventType>
        <ns:ObjectType>MyEventDefinitionObject</ns:ObjectType>
        <ns:ObjectId>123</ns:ObjectId>
        <ns:Source>
          <ns:Product>MyEventDefinition</ns:Product>
          <ns:ProductVersion>1.0</ns:ProductVersion>
          <ns:ProductInstance>MyEventDefinitionInstance</ns:ProductInstance>
        </ns:Source>
        <ns:User>
          <ns:ALFSecurity>
            <ns:UsernameToken>
              <ns:Username>sbmUser</ns:Username>
              <ns:Password>sbmUserPassword</ns:Password>
            </ns:UsernameToken>
          </ns:ALFSecurity>
        </ns:User>
      </ns:Base>
      <ns:Extension>
        <mym:MyEventDefinitionData>Example Tool Data</mym:MyEventDefinitionData>
      </ns:Extension>
    </ns:ALFEventNoticeDoc>
```

```
  </soapenv:Body>
</soapenv:Envelope>
```

Header Values

The most important different between an e-mail event and an event sent through HTTP is that a Web service addressing header must be used:

```
<soapenv:Header>
  <wsa:To>mailto yourEmailUser@yourEmailHost?X-Service-Path=/eventmanager/
→services/ALFEventManagerOneWayDocLit</wsa:To>
  <wsa:MessageID>urn:uuid:421A2EE66BA3A42A8C1203350641340</wsa:MessageID>
  <wsa:Action>urn:EventNotice</wsa:Action>
</soapenv:Header>
```

The value of element `wsa:To` must be in the following format:

e-mailAddress?X-Service-Path=/eventmanager/services/**eventmanagerService**

where **e-mailAddress** is the dedicated POP3 e-mail account set up to receive events, and **eventmanagerService** is the Event Manager service that you want to receive the event.

The Event Manager service should be either `ALFEventManagerOneWayDocLit` or `ALFEventManagerDocLit`. `ALFEventManagerOneWayDocLit` is generally preferred, because it does not send an e-mail response message.

Event Instance Values

The value of `wsa:MessageID` should typically be a newly allocated UUID in the format shown. It may be useful to set the value of the message EventId to the same UUID value, because that will enable you to correlate the e-mail instance with the event.

You do not have to provide a value for `ObjectId`, although it can be useful to do so. Generally, it should contain a unique identifier for the object instance that is responsible for generating the event.

You must provide a value for `Timestamp` or set it to nil. The expected `Timestamp` value uses the subset of ISO 8601 used by XML Schema as specified for the XML Schema type `datetime`. To set `Timestamp` to nil, you must reference the following namespace:

```
xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
```

and set the nil attribute instead of defining a value:

```
<ns:Timestamp xsi:nil="true"/>
```

Event Match Values

As with any SBM external event, the values of `EventType`, `ObjectType`, `Product`, `ProductVersion`, and `ProductInstance` must be set correctly to match a deployed event mapping. E-mail events are essentially external events, so it is generally necessary to create a custom *event definition* [page 685] WSDL and import it into a process app before you can usefully process an e-mail event.

Authentication

As with any SBM external event, the event will not be accepted unless valid credentials are provided in the event request. Currently, the only practical way to do this is to provide a completed `ALFSecurity` element. You must fill in the values for `Username` and `Password`. At this time, only plain-text passwords are accepted.

Extension Data

If your event definition specifies extension data, you can provide the appropriate XML structure and values inside the `Extension` element in the message.

Event Definitions and Message Style

Event definitions are used to define service flows that use the *RPC literal* message style, whereas the *document literal* message style is generally preferred for event communication. The example given above follows the *document literal* format and the document-literal event is contained within the root document element:

```
<ns:ALFEventNoticeDoc xmlns:ns="http://www.eclipse.org/alf/schema/EventBase/1">
  …Event Base and Extension elements go here…
</ns:ALFEventNoticeDoc>
```

Your message should validate against the WSDL/schema for the ALFEventManagerDocLit or ALFEventManagerOneWayDocLit services:

```
http://localhost:8085/eventmanager/services/ALFEventManagerDocLit?wsdl
```

```
http://localhost:8085/eventmanager/services/ALFEventManagerOneWayDocLit?wsdl
```

If you generate an XML document from an *RPC literal* service definition, the event content is contained within the parameter element `<EventNotice xmlns="">`, which in turn is contained within the operation element `<ns:EventNotice xmlns:ns="http://www.eclipse.org/alf/schema/EventBase/1">`:

```
<ns:EventNotice xmlns:ns="http://www.eclipse.org/alf/schema/EventBase/1">
  <EventNotice xmlns="">
    …Event Base and Extension elements go here…
  </EventNotice>
</ns:EventNotice>
```

To create the correct message for the e-mail event, you must replace the operation and parameter `EventNotice` elements with the document root element, `ALFEventNoticeDoc`.

Upgrading Existing Event Definitions

The following topics describe how to upgrade your event definitions to work with the current version of Serena Business Manager.

- Upgrading from SBM R3.X [page 598]
- Upgrading from SBM 2008 R2.X [page 598]

Upgrading from SBM R3.X

The following points discuss upgrading your external *event definition* [page 685] WSDL files from SBM 2008 R3.X to work with the current version.

- Replace the SBM 2008 R3.x Event Manager `.wsdl` and `.xsd` files with new ones from the Advanced Orchestration Package.
- Change the type of `<your product value>` EventBaseType's EventId to `SourceEventIdType`.

Upgrading from SBM 2008 R2.X

Remember the following points when upgrading your existing *event definition* [page 685] WSDL files from SBM R2.X to the current version.

- Replace the SBM 2008 R2.x Event Manager `.wsdl` and `.xsd` files with new ones from the Advanced Orchestration Package.
- Change the type of `<your product value>` EventBaseType's EventId to `SourceEventIdType`.
- You might need to update your clients to set ALFSecurityType in order to fulfill the authentication requirements if you choose to use *Single Sign-On (SSO)* [page 697].

  Starting in SBM 2008 R3, there is an authentication mechanism for external events.

Chapter 32: Raising Events Using JMS Queues

The Event Manager Queue Adapter (EMQA) raises events by listening to messages (as XML documents) in one or more Java™ Messaging Service (JMS) queues. The EMQA reads the XML documents, packages them as events, and sends them to a configured process app in SBM for processing.

This chapter contains the following sections:

Introduction

This chapter provides information about creating *event* [page 685] sources from XML messages in one or more JMS queues using the EMQA in SBM. Material on creating and handling event messages from these sources is also included.

Audience and Scope

This chapter is for experienced software developers or integrators who are familiar with the SOAP standard and Extensible Markup Language (XML), and who have expert knowledge of the Web Service Description Language (WSDL) and of creating *Web services* [page 701]. In addition, an understanding of Java™ Message Service (JMS) and the XML elements that pass through the JMS queue is needed. A working knowledge of SBM is also required. Pertinent information is provided in Part 4: Working with Orchestrations [page 429] and in the *Serena® Business Manager Web Services Developer's Guide*, which is included with the product.

Prerequisites

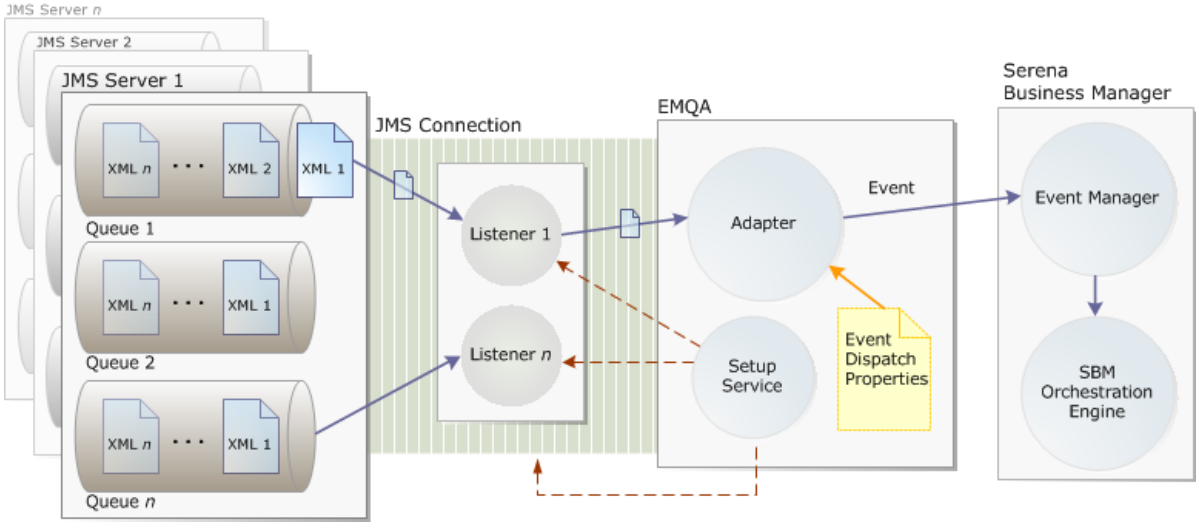
You must have a messaging system that supports the JMS Specification, version 1.1 or later. A previously created connection factory and destination must be available through standard JNDI lookups.

Also, to create the *event definition* [page 685] that enables the *Event Manager* [page 685] to handle the events raised by the EMQA, you must provide the schema associated with the XML documents that you want to send to the JMS queue. (See Creating a Custom Event Definition [page 586] for more information.)

Event Manager Queue Adapter Architecture

This section describes the EMQA architecture. The architecture consists of three main components: 1) the JMS servers and queues, 2) the EMQA, and 3) Serena Business Manager.

Figure 1. The following figure shows the relationship among the three components of the EMQA.



JMS Servers and Queues

The client application produces messages in the form of XML documents and sends them to one or more queues hosted by a JMS server. The JMS server then delivers the XML documents to one of the JMS listeners configured on the EMQA. Any number of JMS server and queue combinations can be configured to run with the EMQA.

Event Manager Queue Adapter

The EMQA is made up of the following three components:

- **EMQA Setup Service**

  Establishes the JMS connections between the JMS servers and the EMQA and adds JMS listeners for specified JMS queues. (See Using the EMQA Setup Service [page 601].)

- **Event Dispatch Properties**

  Provides the product values for the *event* [page 685] SOAP messages. (See Configuring the Event Dispatch Properties [page 606].)

- **Adapter**

Creates event SOAP messages using the XML documents that it reads from the JMS queues, along with the event dispatch properties, and sends them to SBM for processing.

Serena Business Manager

The EMQA sends the process app events that it creates to the SBM *endpoint* [page 684]. The events are first processed by the *Event Manager* [page 685] and then by the SBM Orchestration Engine

.

Deploying the Event Manager Queue Adapter

The EMQA is not deployed by default when you install SBM.

To deploy the EMQA:

1. Back up the `event_dispatch.properties` file in the *`installDir`*`\Common\jboss405\server\default\conf\emqa` folder by adding the `.bak` extension.
2. Copy the following folders from the *`installDir`*`\Orchestration Engine\QueueAdapter` folder into the *`installDir`*`\Common\jboss405\server\default\deploy` folder:
   - `em-queue-adapter-4.1-SNAPSHOT.ear`
   - `emqa-setup-4.1-SNAPSHOT.sar`
3. Restart the JBoss server.

Enabling JMX Console Login

JMX console login is disabled by default when you install SBM. To use the EMQA Setup Service, you must enable JMX console login.

To enable JMX Console login:

1. Open the `jmx-console-users.properties` file located in the *`installDir`*`\Common\jboss405\server\default\conf\props` folder.
2. Delete the number sign (#) next to `admin=admin`.
3. Save and then close the file.

Using the EMQA Setup Service

The EMQA Setup Service lets you establish a JMS connection between the JMS server and the EMQA. This utility is available from the JMX console. You use the EMQA Setup Service to add JMS connections and JMS listeners.

Important: After you add all of the JMS connections and listeners for a particular session, restart the JBoss server. This establishes the connections to the JMS servers and creates the JMS listeners.

Starting the EMQA Setup Service

Follow the instructions in this section to start the EMQA Setup Service.

To start the EMQA Setup Service:

1. Point your Web browser to `http://localhost:port/jmx-console`, where `port` is the port that you specified when you installed Serena Business Manager. (The default port is 8085.)
2. In the **Connect to *Localhost*** dialog box, enter `admin` for **User name** and `admin` for **Password.**
3. Click **OK**.

   The **JMX Agent View *Localhost*** page opens.
4. At the bottom of the page, under **serena.eventmanager**, click **service=EmqaSetupService**.

   The EMQA Setup Service opens on the **JMX MBean View** page.

Adding a JMS Connection

You can add a JMS connection to link the EMQA to a remote JMS server that hosts, or establishes and manages, one or more JMS queues. You can use one of the following methods, which are described in the next two sections.

To determine which method to use:

1. Under **java.util.List listSupportedProviderTypes()**, click **Invoke**.

   A list of preconfigured JMS providers is displayed.
2. If your JMS provider is on the list, use method 1.
3. If your JMS provider is not on the list, use method 2.

Adding a JMS Connection: Method 1

Use this method to add a supported JMS provider type.

Note: You only need to perform these steps once for each JMS provider that you add.

To add a preconfigured JMS provider:

1. Under **java.util.List listSupportedProviderTypes()**, click **Invoke**.

   A list of preconfigured JMS providers is displayed.
2. Copy the name of your JMS provider.
3. Click the **Back to MBean View** link.
4. Under **java.lang.String addJmsServer()**, paste the JMS provider name in the **ParamValue** column of the **providerType** parameter.
5. Enter the name of the JMS server that you want to connect to the EMQA in the **ParamValue** column of the **hostName** parameter.

6. Click **Invoke**.

   The message "The new JMS provider was added successfully" should appear. If you see an error message, go to Troubleshooting [page 613].

7. Click the **Back to MBean View** link.

Adding a JMS Connection: Method 2

Use this method if your JMS provider is not on the list of preconfigured provider types.

Note: You only need to perform these steps once for each JMS provider that you add.

To add a JMS provider:

1. Under **java.lang.String addJmsServer()**, in the **ParamValue** column of the table, enter the required parameter values for the JMS provider you want to add. These values are described in the associated **ParamDescription** column. The value of the **hostName** parameter is the name of the JMS server that you want to connect to the EMQA.

   Note: Contact your IT department if you do not know the values for the parameters, or if you have a custom implementation that requires additional parameters. For assistance in adding parameters, contact Serena Customer Support.

2. Click **Invoke**.

   The message "The new JMS provider was added successfully" should appear. If you see an error message, go to Troubleshooting [page 613].

3. Click the **Back to MBean View** link.

Adding a JMS Listener

Next, you add a JMS listener so the EMQA can read the XML documents that are delivered to a queue by a JMS server. You must add one listener for each server and queue combination.

To add a JMS listener:

1. Under **java.lang.String addListener()**, in the **ParamValue** column, enter the two required parameter values for the JMS listener that you want to create: **hostName** and **queueName**. These values are described in the associated **ParamDescription** column. (Note that the **queueName** parameter value must be in the form "queue/*queueName*".) The **username** and **password** values are optional.

2. Click **Invoke**.

   The message "Successfully added listener *listenerName*" should appear. The listener name is in the form "*hostNamequeueName*Mdb", for example, the JMS listener for the server named SE1234 and its hosted queue named Queue1 is `SE1234Queue1Mdb`. If you see an error message, go to Troubleshooting [page 613].

3. Click the **Back to MBean View** link.

Viewing Connection Details

You can view the contents of the `jms-ds.xml` file. This file contains information about the JMS servers that are connected to the EMQA, such as the connection factory and the URL packages class. This file is located in the *installDir*`\Common\jboss405\server\default\deploy\jms` folder.

To view the connection details for a specific JMS server:

1. Under **java.lang.String getConnectionDetails()**, enter the name of the JMS server for which you want to view connection details in the **ParamValue** column of the **hostName** parameter.
2. Click **Invoke.**

   The contents of the `jms-ds.xml` file are displayed.
3. Locate the name of the JMS server in the `<attribute name="ProviderName">`*hostName*`</attribute>` element and view its associated elements.
4. Click the **Back to MBean View** link.

Viewing a List of JMS Listeners

You can view a list of all of the JMS listeners that were created for all of the JMS servers for which you you have established connections to the EMQA.

To view a list of JMS listeners:

1. Under **java.util.List listAllListeners()**, click **Invoke**.

   The list that is displayed contains the JMS listener name and the associated JMS queue name and server name.
2. Click the **Back to MBean View** link.

Deleting JMS Connections and Listeners

You can delete one or more JMS listeners or all of the listeners on a particular JMS server. You can also delete the connection to a JMS server, which will also delete all of the listeners for that server.

To delete JMS connections and listeners:

1. If you want to delete a single JMS listener, perform these steps:
   a. Under **java.lang.String removeListener()**, enter the name of the JMS server in the **ParamValue** column of the **hostName** parameter.
   b. Enter the name of the JMS queue in the **ParamValue** column of the **queueName** parameter. Note that **queueName** must be in the form "`queue/`*queueName*."
   c. Click **Invoke.** The message "Removed listener *hostNamequeueName*Mdb from JBoss.xml and ejb-jar.xml" should appear. If you see an error message, go to Troubleshooting [page 613].
   d. Click the **Back to MBean View** link.

2. If you want to delete all of the JMS listeners for a particular JMS server, perform these steps:
    a. Under **java.lang.String removeListener()**, enter the name of the JMS server in the **ParamValue** column of the **hostName** parameter.
    b. Leave the **ParamValue** column of the **queueName** empty.
    c. Click **Invoke.** The message "Removed [*number of listeners*] listeners" should appear. If you see an error message, go to Troubleshooting [page 613].
    d. Click the **Back to MBean View** link.
3. To delete the connection to a JMS server and all if its associated listeners, perform these steps:
    a. Under **java.lang.String removeJmsServer()**, enter the name of the server in the **ParamValue** column of the **hostName** parameter.
    b. Click **Invoke.** The message "Removed the JMS provider and all listeners for *hostName*" should appear. If you see an error message, go to Troubleshooting [page 613].
    c. Click the **Back to MBean View** link.

Restarting the JBoss Server

After you add all of the JMS providers, servers, and listeners, restart the JBoss server. This establishes the connections to the JMS servers and creates the JMS listeners.

Note: You can also exit the EMQA Setup Service at this time.

Creating the EMQA User

Both the default user name and password for the EMQA user are specified in the `event_dispatch.properties` file as `emqa`. Therefore, you need to create the "emqa" user in SBM System Administrator with the password `emqa`.

To create the EMQA user:

1. Open SBM System Administrator.
2. Click the **User** tab, and then click **Add**.
3. In the **Login ID** field, enter `emqa`.
4. In the **Password** and **Re-enter Password** fields, enter `emqa`.
5. Under **Product Access**, select **API/Script**.

    Note: This user will not have access to the SBM User Workspace.

6. Click the **Privileges** tab.
7. Click the **System** tab, and then click **Set All**.

8. Click the **Folder** tab, be sure that **Public Folders** is selected in the **For Folder** column, and then click **Set All**.
9. Click the **Item** tab, be sure that **Base Project** is selected in the **For Project** column, and then click **Set All**.
10. Repeat step 9 for the **Field**, **Attach**, **Note**, and **Report** tabs.

    Note: It is not necessary to set permissions on the **Table** tab, unless the "emqa" user requires privileges for any listed auxiliary tables.

11. Click **OK**, and then close SBM System Administrator.

    Note: For more information on creating users, see the *SBM System Administrator Guide*.

Configuring the Event Dispatch Properties

You define the `Product`, `ProductVersion`, and `ProductInstance` values that the EMQA uses to create an event in the `event_dispatch.properties` file.

To configure the event dispatch properties:

1. Open the `event_dispatch.properties` file located in the *`installDir`*`\Common\jboss405\server\default\conf\emqa` folder.
2. Enter the appropriate property values for your product.

   The following sample code shows a configured `event_dispatch.properties` file.

```
Product: MyProduct
ProductInstance: Sole
ProductVersion: 1.2
UserPassword:emqa
Username:emqa
DispatchToUrl:http://localhost:8085/eventmanager/services/ALFEventManagerDocLit
ProductCallbackURI:http://www.serena.com/process_app
EventCustomizerClass:
```

   `Product`, `ProductInstance`, and `ProductVersion` configure the values in the event that EMQA sends to the SBM *Event Manager* [page 685]. They should match the values declared in the *event definition* [page 685] that you create for your EMQA application (see Creating an Event Definition to Handle Events from the EMQA [page 611]).

   `DispatchToUrl` configures the location of the local SBM Event Manager for EMQA. The value shown is typical and does not normally need to be modified unless you are using a different port for the JBoss server in your SBM installation.

   Important: Do not change any of the values below `DispatchToUrl`.

3. Save and close the file.

Testing the Event Manager Queue Adapter

This section explains how to test the Event Manager Queue Adapter using the EMQA Test Service.

This section includes the following topics:

Deploying the EMQA Test Service

The EMQA Test Service is not deployed by default when you install Serena Business Manager.

To deploy the EMQA Test Service:

1. Copy the `emqa-testapp-4.1-SNAPSHOP.sar` folder from the *`installDir`*`\Orchestration Engine\QueueAdapter` folder into the *`installDir`*`\Common\jboss405\server\default\deploy` folder.
2. If have not done so already, copy the `em-queue-adapter-4.1-SNAPSHOT.ear` and `emqa-setup-4.1-SNAPSHOT.sar` folders from the *`installDir`*`\Orchestration Engine\QueueAdapter` folder into the *`installDir`*`\Common\jboss405\server\default\deploy` folder.
3. Open the *`installDir`*`\Common\jboss405\server\default\deploy\emqa-testapp-4.1-SNAPSHOP.sar` folder.
4. Copy the `event_dispatch.properties` file to the *`installDir`*`\Common\jboss405\server\default\conf\emqa` folder.
5. Restart the JBoss server.

Deploying the Test Process App

A test process app is included with the EMQA product. You must first deploy this process app before you send a message to the test queue. You can also use the test process app as a reference for creating your own process app.

To deploy the test process app:

1. Open SBM Composer.
2. Click the Composer button, and then select **Import from File** on the menu. The **Import Process App Blueprint** dialog box opens.

3. Locate the sample process app file, `EmqaTestProcessApp.msd`, in the *`installDir`*`\Common\jboss405\server\default\deploy\emqa-testapp-4.1-SNAPSHOT.sar` folder, and then click **Open.**
4. Deploy the process app by clicking the **Deploy open Process App** button on the Quick Access Toolbar.

Establishing a JMS Connection on Your Local Machine

To use the EMQA Test Service, you must have established a connection on your local machine. If you already have a local connection, go to Adding a JMS Listener for the Test Queue [page 608]. If you have not established a local connection, perform the steps in this section.

To establish a JMS connection on your local machine:

1. Point your Web browser to `http://localhost:`*`port`*`/jmx-console`, where `port` is the port that you specified when you installed Serena Business Manager. (The default port is 8085.)
2. In the **Connect to *Localhost*** dialog box, enter `admin` for **User name** and `admin` for **Password.**
3. Click **OK**.

   The **JMX Agent View *Localhost*** page opens.
4. At the bottom of the page, under **serena.eventmanager**, click **service=EmqaSetupService**.

   The EMQA Setup Service opens on the **JMX MBean View** page.
5. Under **java.lang.String addJmsServer()**, enter `JBossMQ` in the **ParamValue** column of the **providerType** parameter.
6. In the **ParamValue** column of the **hostName** parameter, enter the name of your local machine.
7. Click **Invoke**.

   The message "The new JMS provider was added successfully" should appear. If you see an error message, refer to Troubleshooting [page 613].
8. Perform the steps in Adding a JMS Listener for the Test Queue [page 608].

Adding a JMS Listener for the Test Queue

Prerequisites:

You must have established a JMS connection on your local machine.

In this section, you will add a JMS listener for test queue D.

Note: If you are still in the EMQA Setup Utility, skip to step 5.

To add a JMS listener for the test queue:

1. Point your Web browser to `http://localhost:port/jmx-console`, where `port` is the port that you specified when you installed Serena Business Manager. (The default port is 8085.)
2. In the **Connect to *Localhost*** dialog box, enter `admin` for **User name** and `admin` for **Password.**
3. Click **OK**.

   The **JMX Agent View *Localhost*** page opens.
4. At the bottom of the page, under **serena.eventmanager**, click **service=EmqaSetupService**.

   The EMQA Setup Service opens on the **JMX MBean View** page.
5. Under **java.lang.String addListener()**, in the **ParamValue** column of the **hostName** parameter, enter the name of your local machine.
6. In the **ParamValue** column of the **queueName** parameter, enter `queue/D`.

   The **username** and **password** values are optional.
7. Click **Invoke**.

   The message "Successfully added listener *hostName*DMdb" should appear. If you see an error message, go to Troubleshooting [page 613].
8. Click the **Back to MBean View** link.
9. Click the **Back to Agent View** link.
10. Verify that the EMQA user exists. (See Creating the EMQA User [page 605].)
11. Verify that the event dispatch properties are configured. (See Configuring the Event Dispatch Properties [page 606].)
12. Restart the JBoss server.

Sending a Message to the Test Queue

Prerequisites:

You have established a JMS connection on your local machine and created a JMS listener for test queue D.

Follow the step in this section to send a message (the contents of the `emqa_test_msg.xml` file) to test queue D.

Note: If the JMX console is still open and you are on the main page (in Agent View), go to step 4.

To send a message to the test queue:

1. Point your Web browser to `http://localhost:port/jmx-console`, where `port` is the port that you specified when you installed Serena Business Manager. (The default port is 8085.)
2. In the **Connect to *Localhost*** dialog box, enter `admin` for **User name** and `admin` for **Password.**
3. Click **OK**.

   The **JMX Agent View *Localhost*** page opens.
4. At the bottom of the page, under **serena.eventmanager**, click **service=EmqaTestService**.

   The EMQA Test Service opens on the **JMX MBean View** page. Note that the values of the MBean attributes have been provided for you.
5. Under **java.lang.String sendXmlMessage()**, enter the following values:

Parameter	Value
xmlFilename	The path of the test message file. The contents of this file, `emqa_test_msg.xml`, are the JMS message to be sent to the test queue. The file is located in the *installDir*`\Common\jboss405\server\default\deploy\emqa-testapp-4.1-SNAPSHOT.sar` folder. You might want to copy the file to the root of one of your local drives so the path name is shorter, for example, `C:\emqa_test_msg.xml`.
queueName	queue/D
limit	1

6. Click **Invoke**.

   The **JMX MBean Operation Result sendXmlMessage()** page opens, which should contain the contents of the test XML file preceded by `1 Message(s) sent`. If the contents are preceded by an error message or if the page does contain the contents of the file, go to Troubleshooting [page 613].
7. Click the **Back to MBean View** link.

Locating the Item in the SBM User Workspace

Prerequisites:

You have deployed the test process app and sent the test message to the test queue.

Perform the steps in this section to locate the item that was created by the test process app when you sent the test message to the test queue.

To locate the item:

1. Open the SBM User Workspace.
2. Click the **EmqaApplication** tab.

   You might have to click the **More** tab to see it.
3. In the content pane, under **Reports**, click **Show me "Built-In" Reports**.
4. In the content pane, click **Built-In: All Active Items**.
5. Open the highest-numbered item with the title "EMQA Test Message One."

   The **Submit Date** should be at or after the time you sent the message from the EMQA Test Service. If you do not see a current item, or if there is no item, go to Troubleshooting [page 613].

Creating an Event Definition to Handle Events from the EMQA

You must create your own custom *event definition* [page 685] for handling the events generated by the EMQA in SBM. An *event definition* is a WSDL file derived from the ALF Event Manager WSDL and the ALF event base schema. The event definition specifies custom types and events. It also defines the *orchestration workflow* [page 690] (declared as a service flow service) that handles these events. An event definition created by Serena Business Manager is used to generate events from an *application workflow* [page 680]. You can create a custom event definition or import a customized event definition into SBM Composer, which lets SBM Composer create orchestration workflows that can handle the events, and configure the events so that the *Event Manager* [page 685] will expect them.

Events – A Quick Introduction

An *event* [page 685] is a SOAP message that corresponds to the Serena Business Manager event schema. The message includes five elements that are used by the Event Manager for event matching:

```
<EventType>
<ObjectType>
<Product>
<ProductVersion>
<ProductInstance>
```

The values of these elements determine which orchestration workflow or workflows run when the event is received. The values are declared in the event definition, which is imported into SBM Composer and used to create mappings between the event and the orchestrations that are implemented to process the event. The event definition can also declare event-specific extended data that is sent with the event, enabling the *orchestration* [page 690] to extract the additional data.

How the EMQA Creates the Event SOAP Messages

The adapter reads the XML document messages from the JMS queues specified in the EMQA Setup Service and constructs an event message using the following information:

- The root element of the document

  Because the root element determines the document type, the type of event that is sent varies dynamically depending on the type of document that is read from the queue. Specifically, the values for the `<EventType>` and `<ObjectType>` elements are determined from the name of the root element of the XML document that is read by the EMQA from the JMS queue.

  For example, if an XML document has the root element `<MyDocument>`, as in this example:

  ```
  <?xml version="1.0" encoding="UTF-8"?>
  <exmpl:MyDocument xmlns:exmpl="urn:MyDocumentSchema">
    <exmpl:One>data1</exmpl:One>
    <exmpl:Two>data2</exmpl:Two>
  </exmpl:MyDocument>
  ```

  then the value of both the `<EventType>` and the `<ObjectType>` element will be set to `MyDocument`.

- The values specified in the `event_dispatch.properties` file

  The values of the `<Product>`, `<ProductVersion>`, and `<ProductInstance>` elements are set according to the values in the `event_dispatch.properties` file.

- The contents of the XML document

  The contents of the XML document, including the root element, are inserted into the event message structure as the value of the `<Extension>` element of the event. Using the `<MyDocument>` example from above, the `<Extension>` element will look something like this:

  ```
  …
  <ns1:Extension>
    <exmpl:MyDocument xmlns:exmpl="urn:MyDocumentSchema">
      <exmpl:One>data1</exmpl:One>
      <exmpl:Two>data2</exmpl:Two>
    </exmpl:MyDocument>
  </ns1:Extension>
  …
  ```

  Note: If the document being sent in the queue has a pre-existing schema, you might be able to construct the event definition by including the document schema in the event definition and declaring an appropriately typed element as the content of the `Extension` element. Make sure that the element has the same namespace as the root element of your document. Typically, you must use an element reference that points to the root element of the document.

Troubleshooting

The troubleshooting tips in this section cover the most common problems that you could encounter when installing and testing the EMQA. If you cannot resolve your problems after trying the solutions suggested here, contact Serena Customer Support.

I cannot establish a connection to the EMQA.

Explanation: You could get errors in a number of places indicating that the connection to the EMQA failed. In addition, your process app might not work as expected.

Solution: To correct this problem, try the following:

- In the EMQA Setup Service, verify that the host name is correct.
- Be sure that JMS is installed on the server and is accessible from port 1099.
- Open the `jms-ds.xml` file located in the *`installDir`*`\Common\jboss405\server\default\deploy\jms` folder, find the attributes for your JMS server (the server name is part of the value for the `ProviderName` attribute), and verify that the information matches the properties defined for your server.

I got a `ClassNotFoundException`.

Explanation: This error means that the JMS server is not available, or you did not establish a connection between the server and the EMQA. Your server will continue to try to connect to the EMQA until this problem is resolved.

Solution: To correct this problem, try the following:

- Follow the steps under I cannot establish a connection to the EMQA. [page 613]
- If you still cannot establish a connection, delete the JMS provider for the server in the EMQA Setup Service, and then establish the connection again. (See Deleting JMS Connections and Listeners [page 604] and Adding a JMS Connection [page 602].

An item was not created in Serena Business Manager.

Explanation: In the SBM User Workspace, you could not find the item that should have been created.

Solution: To correct this problem, try the following:

- Perform the steps under I cannot establish a connection to the EMQA. [page 613]
- Be sure that you have created an "emqa" user and that this user has the same authentication credentials specified in the `event_dispatch.properties` file. Also verify that the "emqa" user has the proper privileges. (See Configuring the Event Dispatch Properties [page 606] and Creating the EMQA User [page 605].

Part 6: Dialog Boxes

This section contains the following topics:

Chapter 33: Dialog Boxes

The topics in this section describe SBM Composer dialog boxes. A help topic for the dialog box opens if you press F1 or click **?** when the dialog box is open.

About SBM Composer Dialog Box

Use this dialog box if you are asked by Serena Support to provide information about your software version and computer system configuration. Click the Composer button, and then click **SBM Composer Options**. In the **SBM Composer Options** dialog box, select **Resources**, and then click **About**.

Action Wizard Dialog Box

Use this dialog box to define the *action* [page 679] you want to associate with the selected state or transition. You can set up some action types to be executed before or after a transition, upon entry to a state, or upon exit from a state.

For a state or transition, available actions include invocation of a synchronous *orchestration workflow* [page 690], transition of a linked item, execution of a script, and execution of a Web service method. For a transition, asynchronous orchestration workflows and triggers are also available.

On each page of the wizard, refine the action definition.

For an asynchronous orchestration workflow, a trigger, and a transition, the definition includes which item will be affected, the conditions under which the action will occur, and the orchestration workflow to be invoked or the transition or trigger to be executed.

For asynchronous orchestration, a script, or a Web service, the definition includes which item will be affected, when the execution occurs (before or after a transition, for example), and the actual orchestration workflow, script, or Web service method to be executed.

Some aspects of the definition (such as orchestration workflow type, field names, values, and relationships) must be specified. Click links in the description to make your selections.

Click **Next** and **Back** to modify your choices until you are satisfied with the action definition. Click Finish to save the definition.

The **Action Wizard** dialog box lets you perform the following tasks:

- Selecting the Action Type [page 375]
- Selecting the Affected Item [page 375]
- Selecting the Timing [page 376]
- Selecting the Condition [page 377]
- Selecting the Action [page 379]

Add Application Reference Dialog Box

This dialog box opens when you right-click the **References** heading (or the name of an existing reference) in App Explorer and select **Add Application Reference**. Use it to select an *application* [page 679] that contains the table, field, or other *design element* [page 683] with which you need to make an association.

Element	Description
Look in	Select whether the application is in a process app that is stored in the *Local Cache* [page 688] or in a process app that was checked in to the SBM Application Repository from SBM Composer. Click the column headings to sort the process apps by name, application, category, updater, and date updated. A **+** next to a process app name means that it contains applications. (You cannot define references based on orchestrations.) Click **+** (or double-click the process app name) to list its applications. Then select the application that contains the design element you want to use and click **Add** (or double-click the application name).

Add Existing Design Element from the Head (Tip) Version Dialog Box

This dialog box opens when you right-click a *design element* [page 683] in a patch version and select **Add Existing**. Use this dialog box to copy a design element from the head (tip) version. This is the only way to add some design elements to a *patch context* [page 691].

For more information, see Working in a Patch Context [page 88].

Important: The design element you want to add must already exist in the head version.

Add Transition Dialog Box

This dialog box opens when you click **Add** on the **Behavior** tab of the Property Editor for a **Button**, **HyperLink**, or **Image** control on a custom state form. Use this dialog box to specify the transition that should be executed when the user clicks the control in the SBM User Workspace. For more information, see Behavior Tab of the Control Property Editor [page 208].

Element	Description
Application workflow	If there are multiple application workflows in the process app, select the workflow whose default custom state form contains the control.
State	Select the state with the outgoing transition that should be executed when the user clicks the control.
Transition	Select the outgoing transition that should be executed when the user clicks the control.

Note: One of the following conditions must be met for a state to be in the **State** list:

- The custom state form is specified on the **Form** tab of the state Property Editor.
- The custom state form is specified on the **Forms** tab of the workflow Property Editor, and **[Inherit from workflow]** is specified on the **Form** tab of the state Property Editor.
- The system Update and Delete transitions are present in every application workflow. If you want to map these transitions, select **[All application workflows]** in the **Application workflow** element.

Application Configuration Dialog Box

This dialog box opens when you create a new *application* [page 679]. Use it to provide the required configuration information.

The following table describes the configuration settings for a new application.

Element	Description
Application: Application name	The name you entered in the **New Application** dialog box.

Element	Description
Application: Internal name	The uppercase name under which the application is stored in the database. This name must be unique, and cannot be changed after the process app containing the application is published.
Application: Tab name	The label (up to 16 characters) that appears on the tab for this application in the SBM User Workspace. The name on this label must be unique.
Table: Table name	The name for the *primary table* [page 692] (such as `Issues`).  **Tip:** The name that you type in this field automatically populates the **Database table name** field, which must be unique within the SBM database and is limited to 24 Unicode characters. You should limit the logical name to 24 characters, as additional characters will be left off of the **Database table name**.
Table: Singular item name	The name of a single item in the table (such as `Issue`).  **Note:** This field is automatically filled as you type the **Table name**, so you probably need to edit this field to use it as intended.
Table: Database table name	The uppercase name of the table in the database. This name must be unique. SBM Composer automatically fills this field as you type the **Table name**, ignoring characters after the first 24.  **Note:** The database name can contain only ASCII alphanumeric characters (A–Z, a–z, 0–9). Any other characters are ignored as you type them.

Change Reference Dialog Box

This dialog box opens when you right-click the **References** heading (or the name of an existing reference) in App Explorer—**References** or App Explorer—**All Items** and select **Change**. Use it to change the *application* [page 679] that contains the table, field, or other *design element* [page 683] with which you need to make an association.

Element	Description
Look in	Select whether the *application* [page 679] is in a *process app* [page 692] that is stored in the *Local Cache* [page 688] or in a process app that was checked in to the SBM Application Repository from SBM Composer. Click the column headings to sort the process apps by name, application, category, updater, and date updated. A **+** next to a process app name means that it contains applications. (You cannot define references based on orchestrations.) Applications that can be resolved automatically are displayed in bold. Click **+** (or double-click the process app name) to list its applications. Then select the application that contains the design element that you want to use, and click **Add** (or double-click the application name).

Check In Design Elements Dialog Box

Use this dialog box to indicate which elements you want to check in to the repository. The process app must be checked in before you can *publish* [page 692] it. To open this dialog box, select **Process App Repository** and then **Check In All** from the Composer menu. Click **OK** to start the check-in process.

Element	Description
Process app tree	Select the check boxes to check in individual elements, entire groups of elements, entire applications or orchestrations, or the entire process app. Click **+** and **–** to expand and collapse levels of the hierarchy.
Comment	Enter text that to appear in the *version history* [page 700] for the selected design elements.

If the selected process app or application *design element* [page 683] already exists in the SBM Application Repository, you cannot check it in, even if it has another name. Instead, SBM Composer offers you these choices:

- Check in new versions of the selected design element and all its comprised design elements created from the local versions.

- Get the latest versions of the selected design element and all its comprised design elements from the repository. To the extent possible, their SBM Application Repository status (that is, whether they are checked in or checked out) is retained. This is the same as the **Get Latest Version** command.

  CAUTION: As with the **Get Latest Version** command, any changes you made locally to the selected design element and its comprised design elements are lost.

- Cancel the check in, export the selected process app (or the process app containing the selected application), and then restart the check-in process. The exported *blueprint file* [page 681] can be sent to another *designer* [page 683] or loaded directly into Application Administrator.

Check Out Design Elements Dialog Box

Use this dialog box to indicate which elements you want to check out of the SBM Application Repository. Elements must be checked out before you can edit them. To open this dialog box, from the Composer menu, point to **Process App Repository** and then select **Check Out All**.

Element	Description
Process app tree	Select the check boxes to check out individual elements, entire groups of elements, entire applications or orchestrations, or the entire process tree. Click **+** and **–** to expand and collapse levels of the hierarchy.

Compare Process Apps Dialog Box

This dialog box opens after you select **Compare** from the Composer menu and then select either a process app that was published to the SBM Application Repository or a *blueprint file* [page 681] that is stored in the file system of your computer.

Use the comparison to determine the modifications that would be required to change from the currently open process app to the process app you selected for comparison.

Element	Description
Legend (below the title bar)	Identifies the blueprint file or published version with which you chose to compare the currently open process app.
Export	Saves the detailed descriptions of the differences as an HTML page. Use the **Browse For Folder** dialog box to select a storage location for the report, and all of the images and other files that support it.

Element	Description
Tree (left column)	Shows the combined content of the two process apps, highlighting design elements that differ and showing the process app in which they exist. An element that differs exists in both process apps. A heading differs if its name or description has changed, or if any of its elements exist in one process app but not in the other. An element or heading is named in italics if it or its content differs in any way between the open process app and the compared process app. Click any element name in the tree to scroll to the corresponding design element in the **Differs** column. Click the **+** and **–** boxes to expand and collapse levels of the hierarchy, so you can see all the elements.  **Note:** Click anywhere in this column to view a menu of standard browser commands, from which you might choose to print the report. You can also export the report and print it from your browser.
Differs (right column)	Provides detailed information about the specific differences between the currently open process app and the process app selected for comparison. Red indicates elements that exist only in the currently open process app, green indicates elements that exist only in the process app selected for comparison, blue indicates elements that exist (with differences) in both process apps, and pale grey indicates elements that are identical in the two process apps. Scroll the right column manually to see its content. Click links in the right column to view images, forms, workflow images, SBM AppScripts, and JavaScripts.  **Note:** Click anywhere in this column to view a menu of standard browser commands, from which you might choose to print the report. You can also export the report and print it from your browser.

Configure Process App Dialog Box

Use this dialog box to configure a process app that you are creating based on a *template* [page 699] under **Available templates** in the **Create New Process App** dialog box. For more information, see Create New Process App Dialog Box [page 626].

The **Configure Process App** dialog box forces you to change *process app* [page 692], *application* [page 679], database table, and workflow names. This is necessary because validation fails if any application, table, or workflow in the new process app has the same database name as it had in the template. Applications, tables, and workflows within a given deployment environment must have unique database names.

Note: The process app name or application name of the template is included in the label for each group of elements. This helps you avoid using the same name in your new process app.

Note: The **Application**, **Table**, and **Workflow** elements are not in this dialog box if you are creating an empty process app or an orchestration process app.

Element	Description
Process app: Process app name	Type a name for the process app. You cannot use the name of the process app template.
Process app: Category	Type or select a process app category. Categories let you sort process apps in your *Local Cache* [page 688] or the SBM Application Repository. The categories in the list are the AppCentral™ categories. Categories that you previously typed in this field are not in the list.
Application: Application name	Type a name for the process app application. Applications within a deployment environment must be uniquely named, even if they are used in different process apps. Applications within a process app must also be uniquely named. Applications created independently in different process apps can have the same name. In this case, changes to one do not affect the other.
Application: Internal name	Type the uppercase name under which the application is stored in the database. This name must be unique, and cannot be changed after the process app containing the application is published. **Note:** The internal name can contain only ASCII alphanumeric characters (A-Z, a-z, 0-9). Any other characters are ignored as you type them.
Application: Tab name	Type the label (up to 16 characters) that appears on the tab for this application in the SBM User Workspace. The name on this label must be unique.

Element	Description
Table: Table name	Type the name for the *primary table* [page 692] (such as `Issues`).  **Tip:** The name that you type in this field automatically populates the **Database table name** field, which must be unique within the SBM database and is limited to 24 Unicode characters. You should limit the logical name to 24 characters, as additional characters will be left off of the **Database table name.**
Table: Singular item name	Type the name of a single item in the table (such as `Issue`). You cannot use the name that is used in the process app template.  **Note:** This field is automatically filled as you type in the **Table name** field, so you probably need to edit this field to use it as intended.
Table: Database table name	Type the uppercase name of the table in the database. This name must be unique. SBM Composer automatically fills this field as you type the **Table name**, ignoring characters after the first 24.  **Note:** The database table name can contain only ASCII alphanumeric characters (A-Z, a-z, 0-9). Any other characters are ignored as you type them.
Workflow: Workflow name	Type the name (32 characters or less) of the *application workflow* [page 680] to be used by the process app.

Create New Process App Dialog Box

This dialog box opens when you click select **New** from the Composer menu or press Ctrl+N from anywhere in SBM Composer.

In this dialog box, you can create a process app based on the following types of process apps:

- Process apps stored in AppCentral™. AppCentral™ is a channel that lets designers share process apps. It helps designers build process apps quickly and easily by providing a tight integration with SBM Composer. The process apps in AppCentral™ are organized by category. When you select a process app from AppCentral™, another **Create New Process App** dialog box opens. This dialog box requires you to fill in information about yourself and your company, and agree to an End-User License Agreement (EULA). For more information, see Create New Process App Dialog Box (AppCentral™) [page 629].
- A process app with the basic elements of an *application* [page 679] or an orchestration, or an empty process app to which you can add applications and orchestrations.

- Process apps that are stored on your computer. These are blueprint (`.msd`) files and *template* [page 699] (`.mst`) files.

  Note: A *blueprint* contains process app design elements such as workflows, orchestrations, roles, fields, and custom forms. A *template* is a type of process app that is a starting point for creating other process apps. Templates cannot be published or deployed, and cannot be referenced by other process apps. For more information about templates, see About Templates [page 77].

Element	Description
Process App Categories	Click **Templates** to show the process app templates in the **Available Templates** pane.
AppCentral	Select a category of process apps that are stored in AppCentral™ to show them in the **Available Process Apps** pane. Select **Latest** to see only those process apps that are compatible with your version of SBM Composer, or **All** to include older versions of the process apps.
Available Templates	This pane is displayed when you click **Templates** under **Process App Categories**. It contains icons that represent a basic application process app, a basic orchestration process app, and blueprints and templates that you opened from your computer. When you select an icon for a blueprint or template, a brief description is displayed in the pane on the right. The file name is included in the description. Files with an `.msd` extension are process app blueprints and files with an `.mst` file extension are process app templates. Select a template icon and click **Create** to open the **Configure Process App** dialog box and create the new process app. For more information, see Configure Process App Dialog Box [page 625]. When you click the **Browse** button, a standard **Open** dialog box opens that lets you bring in a template file or a *blueprint file* [page 681] that is stored on your computer.
Available Process Apps	Shows images that represent the process apps in the category you clicked under **AppCentral**. Click the "page" links at the bottom of this pane to see more process apps in the selected category. When you select one of these images, a description of the process app is displayed in the pane on the right. Select an image and click **Create** to open another **Create New Process App** dialog box and agree to the license agreement. For more information, see Create New Process App Dialog Box (AppCentral™) [page 629].

Note: If you try to create a process app based on a template that is not compatible with the on-demand SBM Application Repository to which you are connected, you receive a warning message that gives you the opportunity to switch to on-premise mode. However, if you switch, you cannot export or *publish* [page 692] the new process app. This situation does not occur if you are connected to an on-premise SBM Application Repository, because an on-demand process app is always compatible with an on-premise SBM Application Repository. This is because on-demand functionality is a subset of on-premise functionality.

Create New Process App Dialog Box (AppCentral™)

This dialog box opens when you click a pre-built process app in the **Create New Process App** dialog box to use as a *template* [page 699] for a new process app. It requires you to provide information about yourself and your company and requires you to agree to an End-User License Agreement (EULA) for process apps that are stored in AppCentral™.

Element	Description
Personal	Type your first name, last name, e-mail address, and telephone number.
Company	1. Type your company name and address. 2. Select a country from the **Country** drop-down list. 3. Type your company's country, city or town, state or province, and zip or postal code.
Terms of use	1. Click the link to read the terms of use. 2. If you agree to the terms of use, select the check box. 3. Click **Create** to create the new process app from the blueprint.

Create Patch Context Dialog Box

Use this dialog box to create a new label, called a *patch context* [page 691], for the changed version of a published process app. Appending a unique identifier (such as "Patch 1") to the original label helps relate the new version to the original and helps keep subsequent versions straight.

If you do not accept the option to create a patch context, SBM Composer opens the selected process app in read-only mode. This means that you cannot check out, check in, or undo the checkout of the process app.

Delete Item Dialog Box

This dialog box opens when you try to delete a *design element* [page 683], but the design element cannot be deleted because it is being used by the process app.

For example:

- You add a form to an *application* [page 679], use the form on a state or transition, and try to delete the form. The form cannot be deleted because the state or transition uses it.
- You create a report definition for an *auxiliary table* [page 680], and then try to delete the auxiliary table. The auxiliary table cannot be deleted because the report definition uses it.

To go to the part of the process app that contains the design element that is being used, either double-click the element in the dialog box, or select the design element in the dialog box and then click **Go to location**.

Note: If you want to delete a design element that is being used by the process app, you must remove the connections between that design element and all of the design elements that use it.

Deploy Process App Dialog Box

Use this dialog box to deploy a process app. If the process app has not been validated, checked in, or published, these functions are performed transparently, unless there is an error. To open this dialog box, from the Composer menu, point to **Deploy** and then select **Deploy**.

Element	Description
Environment	Select the *environment* [page 684] to which you want to deploy the process app. For environments to be seen in this list, in the **Edit Environment** dialog box in Application Administrator, **Enable Deployment** must be selected from the **SBM Composer** list.
Label	Type the label or version name for the process app, if you want to modify the text that identifies this version of the process app.
Visibility	Select the **Allow others to deploy this process app** check box or clear it, depending on whether you want others to be able to deploy the process app.
Optional Comment	*(Optional)* Type information about the deployment.

Note: Applications and tables in your deployment environment must be uniquely named. If the deployment fails because an application or table in the environment already has the same name as one of the applications or tables in the process app you are deploying, rename the application or table in the process app you are deploying, and try again.

Deploy Options Dialog Box

This dialog box opens when you click **Options** on the **Deploy Process App** dialog box. Use this dialog box to specify options when you deploy a process app.

Element	Description
Stop deploy	Specify whether the deployment should stop automatically if any warnings are raised during the deployment process. (By default, deployment always stops if there are errors.)

Element	Description
Schedule	Select this check box to delay the deployment. Use the controls to set the time and date.
Email notification, Email address	Use these controls to specify what deployment related *notifications* [page 689] that you want to receive and to what address they should be sent.  **Important:** This only works if your administrator enabled e-mail notifications. If this was not done, notifications will not be sent, even if you complete the fields in this dialog box.
Endpoint mappings	For each *endpoint* [page 684] in your process app, click to the right of the external endpoint, and either select the defined endpoint to which it should be mapped, or select **New endpoint** and select **New endpoint** to open the New Endpoint Dialog Box [page 637].

Embedded Report Configuration Dialog Box

 **Important:** When you create a report in the SBM User Workspace, you must specify a report reference name so the report will work even after the process app is promoted to another environment. For instructions, see Referencing a Report [page 250].

This dialog box opens when you click **Configure Report** on the **General** tab of the Property Editor for the Embedded Report Widget [page 229]. You copy the report URL from the User Workspace and then paste the URL into this dialog box. After you close the dialog box, the **Report name** field on the **General** tab of the widget Property Editor is populated with the report name that you specified in the User Workspace.

The following describes the configuration settings for a new embedded report.

Element	Description

Report address	Type or paste the URL of the report. Input parameters are immediately added to the **Inputs** section.
	An example URL is:
	`tmtrack.dll?ReportPage&template=reports%2Flist&ReportRef=`
	`→UBG_ISSUES.NewIncomingIssues&HasRuntimeParams=`
	`→1&F_TS_ACTIVEINACTIVE=&embedded`

Inputs	Shows the input parameters for the report. The inputs are read-only, because they come from the report in the User Workspace and cannot be modified through SBM Composer.

Find Dialog Box

Use this dialog box to locate specific design elements in the process app using the keyboard. To open this dialog box, press Ctrl+F, type the text you want to find, and then

press Enter. The first design element is highlighted in App Explorer, and its editor opens. To continue searching with the dialog box open, click **Find** or Enter. To continue searching with the dialog box closed, press F3.

Note: To access the **Find** dialog box with the mouse, select **Find** from the context menu.

This dialog box is used to find items in a specific area of SBM Composer. To extend your search to a broader area of SBM Composer, use the Find Items Dialog Box [page 632].

Element	Description
Find what	Type the name of the element you want to find, or use the drop-down list to reuse a previous search string.
Find options	**Match case.** Select this check box to restrict the search to items that match the case of the word or words in the **Find what** box.

Find Items Dialog Box

Use this dialog box to locate specific design elements in the process app context that you specify. To open this dialog box, click the **Find Items** button on the Ribbon or quick access toolbar, press Alt and then F, or press Ctrl+Shift+F.

Element	Description
Find what	Type the name of the element you want to find, or use the drop-down list to reuse a previous search string.
Look in	Use the drop-down list to limit your search to the current editor, the current *application* [page 679], or the entire process app.
Which types	The choices in this section depend on the scope of the search. Select the check box for the design element or design elements you want to search, or select the **Select all** check box. To start over, click the **Unselect all** check box.
Find options	Select any of the options to further restrict this search.

Note: You can also use the Find Dialog Box [page 631] to locate design elements. It has less options than the **Find Items** dialog box.

Find and Replace Dialog Box

Use this dialog box to locate specific text in the AppScript or JavaScript editor, or to locate the text and replace it with something else.
To open this dialog box, right-click in the content area of the editor, and do one of the following:

- Select **Find...** to find text. Alternatively, press Ctrl+F.

- Select **Replace** to find and replace text. Alternatively, press Ctrl+H.

Element	Description
Find what	Type the text you want to find or replace with something else.
Replace with	Select the check box and then type the text that should replace the text in the **Find what** box.
Options	Select options to limit the search.
Find next	Click this button to find the next instance of the search text.
Replace all (***Replace*** *command only*)	Click this button to replace all instances of the search text with the text you specified.
Replace (***Replace*** *command only*)	Click this button to replace the selected instance of the search text with the text you specified, and find the next instance.

Find Results Pane

This pane shows the results of a search from the **Find Items** dialog box.

This pane occupies the same space as the Property Editor, below the editor pane. When multiple panes are open in this area, arrange them the same way as you arrange editor tabs in the editor pane.

If you double-click an item in this pane:

- The applicable editor opens, and the item is selected in the editor.
- The Property Editor for the item opens.
- The item is displayed in blue to indicate that it has been "visited."

If you select an item in this pane and then right-click, the context menu contains the following commands:

- **Go to Location** is the same as double-clicking the item (see above).
- **Remove** deletes the item from the pane.
- **Clear** removes all items from the pane.

Form Configuration Dialog Box

This dialog box opens when you create a new custom form. Use it to provide basic parameters for the new form.

Note: For more information, see Creating Custom Forms [page 198] and Quick Forms and Custom Forms [page 196].

The following table describes the configuration settings for a new custom form.

Element	Description
Name	The name of the new custom form. Each form must have a unique name. If you type the name of an existing form, a number is appended to the name (for example, "State Form 2").
Type	Select **State form** or **Transition form**.  **Note:** You cannot select **State form** if you are creating a form for a transition, and you cannot select **Transition form** if you are creating a form for a state or for an *auxiliary table* [page 680].
Style	Select one of the following: **Based on "quick" form for**: Creates a form based on the *quick form* [page 693] for the selected *application workflow* [page 680], *primary table* [page 692], or auxiliary table in the open process app. **Based on another form**: Creates a form based on the selected custom form in the open process app. In most cases, the form types do not have to match, so you can create the form from either a state form or a *transition form* [page 700]. For exceptions, see the **Type** element description. **Blank form for**: Creates a blank form.
Preview	Displays a read-only version of the blank form or the form from which you are basing the new form.
Actual size	If this check box is selected, the size of the form in the **Preview** element is the same as the size of the form in the form editor. If this check box is cleared, the form is sized to fit in its entirety in the **Preview** element.

Form Preview Dialog Box

This dialog box shows how the selected form would look to someone using your process app. This dialog box is accessible from the **Forms** tab of the Property Editor for the workflow, state, or transition, and from the **Design** tab of the Ribbon.

Element	Description
Context	Select a state or transition that uses or inherits the form.
Role	Select from the roles authorized to view the form.

Element	Description
Enable JavaScripts	Select this check box if you want JavaScripts that are included in the form to be executed during form preview. Otherwise, JavaScripts are suppressed during form preview.  **Note:** By default, this check box is not selected, because some scripts are too complex to show during form preview.

Get Latest Design Elements Dialog Box

Use this dialog box to get the latest version of the selected process app or elements from the repository, overwriting any changes you made since checking in the selected process app or elements. This dialog box does not check out the process app or elements. To open this dialog box, from the Composer menu, point to **Process App Repository** and then select **Get Latest of All**.

Element	Description
Process app tree	Click the check boxes to get individual elements, entire groups of elements, entire applications or orchestrations, or the entire process app. Click **+** and **–** to expand and collapse levels of the hierarchy.

Import Process App Blueprint Dialog Box

This dialog box opens when you point to **Import and Export** and then select **Import from File** from the Composer menu.

Find the *blueprint file* [page 681] previously exported from SBM Composer or saved from Application Administrator, and then click **Open**. You can also find a *template* [page 699] file previously exported from SBM Composer. (For information about templates, see About Templates [page 77].)

After you import a blueprint or template, use the **Open** command to continue working on it.

CAUTION: If you try to import a process app that you already imported on the same computer, SBM Composer warns you that the version in the *Local Cache* [page 688] is overwritten with the imported process app. If you import the process app anyway, any work you did on that process app because the previous import is lost.

If the process app was also checked in to the SBM Application Repository, SBM Composer warns you that, on checking it in, you have to decide whether to check in new versions of its design elements created from the local process app, or overwrite the local process app with the latest versions of its design elements from the SBM Application Repository.

To prevent loss of any work you did on the local copy:

1. Cancel the import.
2. Export the local process app to a file.
3. Restart the import process.

You can send the exported blueprint file to another *designer* [page 683] or load it directly into Application Administrator.

Insert Dialog Box

Use this dialog box to set up the number of rows and the number and width of columns in the expander, group box, panel, or tab you are dragging to the form. This dialog box opens when you drag one of these elements to the form.

Note: This provides a starting point. In the Property Editor for the form, and from the context menu that opens when you right-click a row or column, you can add and delete rows and columns, or change the width of columns and the height of rows on an existing form.

New Application Dialog Box

Use this dialog box to add a new *application* [page 679] to the *process app* [page 692].

Element	Description
Name	Provide a name for the new application. Applications within a deployment environment must be uniquely named, even if they are used in different process apps. Applications within a process app must also be uniquely named. Applications created independently in different process apps can have the same name. In this case, changes to one application do not affect the other application.
Category	Select an existing category, or type the name of a new category.

New Endpoint Dialog Box

This dialog box opens from the **Deploy Process App** dialog box when you click the arrow to the right of the external *endpoint* [page 684], and then select **New Endpoint**.

Element	Description
Name *(required)*	Name by which the new endpoint is identified in the **Endpoint mappings** section of the **Deploy Process App** dialog box.
Description	An optional description of the *environment* [page 684]. This text is displayed below the list of available environments in the **Deploy Process App** dialog box.

URL *(required)*	Enter (or select) the URL for the external endpoint.
	Examples of valid URLs follow:
	`http://serverName:port/eventmanager/services/ALFEventManager`
	`http://serverName:port/orchestrationName-→orchestrationWorkflowNameInLowercaseservlet`

Authentication	Select the authentication method to be used for the endpoint: **none**, **Basic**, or **Single Sign-On (SSO)**. This information is typically available from the same source as the endpoint URL. If you selected **Basic**, type the user name and password used to perform the authentication.

Open Labeled Version Dialog Box

To open an earlier labeled version of a process app, select it in the **Open Process App** dialog box and click **Open Labeled Version**. In the **Open labeled version** dialog box that opens, select the version by the label assigned to it on publication. If you use labels to denote patches to your process apps (as in MyProcessApp, MyProcessApp 1, MyProcessApp Patch 1.1, and so on), they are easier to find in this dialog box.

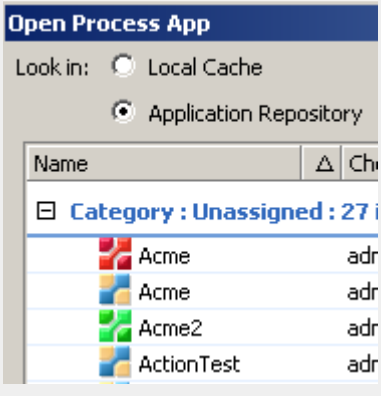
You have the option to create a *patch context* [page 691] in which to make any changes, if one does not already exist. If you do not accept the option to create a patch context, SBM Composer opens the selected process app in read-only mode.

For detailed information about patch contexts, see Working in a Patch Context [page 88].

Open Process App Dialog Box

This dialog box opens when you select **Open** from the Composer menu or press Ctrl+O from anywhere in SBM Composer. Use it to select the latest version of a process app that was checked in to the SBM Application Repository or a process app that you saved to your computer (the *Local Cache* [page 688]).

Note: When you open SBM Composer for the first time, the list of recently opened process apps is empty. Any new process apps are added to the list after you close them.

Element	Description
Look in	Choose whether to select a process app stored on your computer (Local Cache) or a process app that was checked in to the SBM Application Repository. If a process app in the SBM Application Repository is listed twice, it means someone loaded the process app directly into the SBM Application Repository (from an exported *blueprint file* [page 681] or from the SBM Application Engine) since the last time the process app was checked in or published from SBM Composer. The process app with the red icon is the one that was loaded directly into the SBM Application Repository. The process app with the yellow and blue icon is the one that was checked in or published from SBM Composer. A process app with a green icon is a *template* [page 699]. For more information about templates, see About Templates [page 77].  Use the icon, the time stamp, or the description that is displayed below the list to determine which one you want to open. **CAUTION:** If you select a process app from the repository that was opened on your computer (that is, a process app that already exists in the Local Cache), SBM Composer warns you that the local process app will be overwritten. You can open the local process app or overwrite it with the process app in the repository, losing any changes you made since the process app was checked in.
Open labeled version	Opens the **Open Labeled Version** dialog box, in which you can open an earlier labeled version of the selected process app. You have the option to create a *patch context* [page 691] in which to make any changes, if one does not already exist. For more information, see Open Labeled Version Dialog Box [page 637] and Working in a Patch Context [page 88].

Publish Process App Dialog Box

This dialog box opens when you right-click the process app name in App Explorer and select **Publish**, or select **Publish** from the Composer menu.

Use this dialog box to *publish* [page 692] the process app so it can be deployed. If the process app was not validated or checked in, those tasks are performed first.

Element	Description
Label	In the **Version name** field, type the label or version name for the process app.
Visibility	If you do not want others to deploy this process app, clear this check box.
Optional Comment	Type an explanation of or comment about the *deployment* [page 683], if needed.

Tip: If multiple mashers are working on the design elements in your process app before you publish the process app, right-click the process app *design element* [page 683] (the top-level item) in App Explorer, and select **Get Latest** (unless the process app design element is already checked out to you when you publish).

Tip: Applications and tables within the SBM Application Repository must be uniquely named. If the publication fails because an *application* [page 679] or table in the SBM Application Repository already has the same name as one of the applications or tables you are checking in, rename the local design elements, and try again. See Check In Design Elements Dialog Box [page 622] for related information.

If the process app was published, a different **Publish Process App** dialog box opens when you right-click the process app name in App Explorer and select **Change Published Scope**.

Element	Description
Name	Displays the name of the process app.
Comment	Type text that will appear in the publication history for the process app.
Scope	**Public**: Anyone with the appropriate privileges can deploy the process app. **Private**: Only you can deploy the process app (typically used for process apps under development).
Baseline Label	Type a baseline label. This label is displayed in Application Administrator.  **Note:** You cannot change this field when you are changing the scope of publication.

Referenced Applications Dialog Box

This dialog box opens when you execute the **Deploy** or **Quick Deploy** command on a process app that contains external references. This dialog box lists any applications to which the process app that you are deploying has references.

You receive warning messages when you try to deploy a process app with a reference to a missing application. You can still deploy the process app, but references are lost if they are not resolved before a process app is deployed.

Note: Applications that are unresolved references are not listed in this dialog box.

Resolve Reference to Application Dialog Box

This dialog box opens when you right-click an unresolved reference under the **References** heading in App Explorer or click the message that appears at the top of the application editor. Use it to locate the *application* [page 679] that is likely to resolve the unresolved reference.

CAUTION: Save any changes you make to the process app before you resolve an application reference, because the only way you can "undo" the application reference resolution is to close the process app without saving changes.

Note: There are situations in which you do not need to resolve references. For more information, see Resolving References [page 368].

Element	Description
Look In	Select whether the application is in a process app that is stored in the *Local Cache* [page 688] or in a process app that was checked in to the SBM Application Repository from SBM Composer. Click the column headings to sort the process apps by name, application, category, updater, and date updated. A **+** next to a process app name means that it contains applications. (You cannot define references based on orchestrations.) Click **+** (or double-click the process app name) to list its applications. Then select the application that contains the *design element* [page 683] that you want to use, and click **Resolve** (or double-click the application name). Compatible applications (that is, applications that have the same design number) are shown in bold. For information about design numbers, see About Design Numbers [page 362].

REST Service Configuration Dialog Box

This dialog box opens when you click the **Configure URL** button on the **General** tab of the Property Editor for the REST Grid *widget* [page 701]. It lets you add or change widget parameters.

The following table describes the configuration settings for a new REST service.

Element	Description
Address	Type or paste the URL of the REST service. Input parameters are immediately added to the **Inputs** section. Click **Update outputs** to populate the **Outputs** section and test the URL. An example URL is: `http://answers.yahooapis.com/AnswersService/V1/questionSearch?appid=YahooDemo&query=cars&output=json` When you type a URL to a REST service that requires authentication, a popup window is presented, in which you type a user name and password.
Add input	Adds add a new input parameter. When you add a new parameter, the URL is immediately updated.
Remove input	Removes the selected input parameter.
Update outputs	Populates the **Outputs** section and tests the URL.
Clear outputs	Clears the parameters in the **Outputs** section.
Inputs	Displays input parameters that are sent to a REST service. The parameters are populated when you provide a URL. You can set new default values for input parameters that override ones set in a provided URL. These parameter values can be changed later on the **Query** tab. The URL is updated after you change a parameter name or value, and then click somewhere else or press the Enter key.
Outputs	Displays output parameters received from a REST service. If you change the input parameters, click **Update outputs** to refresh the output parameters list based on changes to the input parameters (for example, adding, deleting, renaming, or changing a value).
Ignore XML prefixes	Specify whether XML namespace prefixes should be ignored. Namespace prefixes are arbitrary, can be automatically generated, and can change. The REST Grid widget requires a constant namespace prefix, so if the prefix changes, the widget does not work. If you select this check box, the namespace prefix that is returned in the data from the REST service is not used, so the widget works consistently. This check box is present only if you specified a REST service address that returns XML (an XML REST Web service, not a JSON REST Web service).
Data for rows	Defines the array in the data set to be used for rows in the grid.

Select Published Process App Dialog Box

This dialog box opens when you point to **Compare** and then select **With Published Process App** from the Composer menu. Select the process app blueprint you want to compare to the currently open process app, and then click **Open**.

Element	Description
Look in (Process App Repository)	Lists the latest published version of the process apps in the SBM Application Repository.  **Note:** If the Look in element shows *Local Cache* [page 688], SBM Composer is disconnected from the SBM Application Repository If this is the case, then perform the following steps: 1. Cancel the compare operation. 2. In the lower right corner of the SBM Composer window, click the **Offline** status indicator. 3. Click **Work Offline** to disable that option (that is, to connect to the SBM Application Repository), and make sure that SBM Composer reports its status as **Connected**. 4. From the Composer menu, point to **Compare** and then select **With Published Process App** again. The **Look in** element should now show SBM Application Repository.

Service Mappings Dialog Box

This dialog box opens when you select a Web service in the **Action Wizard** and click the **data mapping** link. Use it to set up the exchange of data between the Web service (referred to by a state or transition *action* [page 679]) and your *application* [page 679].

Element	Description
Service data	This column lists the Web service operations and their parameters.
Application data	Use this column to map a field from the application's *primary table* [page 692] to the needed parameter for the Web service. To select a field from the primary table, click in a cell to display a drop-down list of values. For example, you might map the *Owner* field from the primary table to the *UserID* field required by the Web service operation.  **Note:** If you change the name of a primary table field after mapping it here, be sure to review this mapping to determine whether it needs to be reestablished using the new name.

Element	Description
Constant value *(Inputs tab)*	Use this column to "hard code" a value as the input to the corresponding parameter, rather than mapping it to an application data field.  **Tip:** Specifying a constant value clears the corresponding **Application data** selection, and selecting an application data field clears the corresponding **Constant value** entry.

Sort By Dialog Box

Use this dialog box to sort the rows in the displayed table by the values in up to three of the columns in the table.

For each column, select the column name from the drop-down list, and select the sort order (A–Z or Z–A).

Undo Check Out Design Elements Dialog Box

Use this dialog box to discard any changes you made to the selected design elements since you checked them out, and leave the elements checked in and unchanged in the SBM Application Repository. To open this dialog box, point to SBM Application Repository and then select **Undo Check Out All** from the Composer menu.

Element	Description
Process app tree	Click the check boxes to undo the check out of individual elements, entire groups of elements, entire applications or orchestrations, or the entire process app. Click the **+** and **–** boxes to expand and collapse levels of the hierarchy.

Update Status of Design Elements Dialog Box

Use this dialog box to get the SBM Application Repository status of the selected design elements. The SBM Application Repository status could be checked out by you, checked out by someone else, and so on.

Element	Description
Process app tree	Click the check boxes to update the SBM Application Repository status of individual elements, entire groups of elements, entire applications or orchestrations, or the entire process app. Click the **+** and **–** boxes to expand and collapse levels of the hierarchy.

Version History Dialog Box

Use this dialog box to see the history of versions of the process app or selected element that is checked in to the SBM Application Repository. Sort by any of the column headings. Switch to the **As text** tab for a format you can copy and use somewhere else.

To open this dialog box, click the Composer button, point to **Process App Repository**, and then select **Process App Version History**. Alternatively, right-click the process app name in App Explorer, and then select **Version History**.

To see check-in comments on the **As list** tab, either select the **Show comments** check box, or right-click a version on the tab and then select the **Show comments** check box.

Web Service Configuration Dialog Box

This dialog box opens when you create a new Web service. Use it to provide the required configuration information.

The following table describes the configuration settings for a new Web service.

Element	Description
WSDL	The name of the Web Service Description Language (WSDL) file that defines the Web service. Browse for a file with a `.wsdl` extension (or enter a URL), and then press the Tab key or click in another field to read the file.
Service	Lists the services defined in the WSDL file. If only one service is defined, this field is read-only.
Port	Lists the unique names of ports for the selected service, as defined in the WSDL file. If only one port is defined, this field is read-only.
Documentation	Optional information about the services, as provided by the creator of the WSDL file. This field is read-only.
Operations	Lists the individual operations available for the selected service, as defined in the WSDL file.

Where Used Dialog Box

This dialog box lets you see where any *design element* [page 683] is used in a process app. For example, you could have a large process app that has many transitions and states. It would be useful if you could find out which transitions and states use a particular form.

There are two ways you can open this dialog box:

- Select the design element (for example, a form) in App Explorer and then click **Where Used** in the **Find** area on the **Home** tab of the Ribbon.
- Select the design element in App Explorer, right-click, and then select **Where Used**.

To go to the part of the process app that contains the design element being used, either double-click the design element in the dialog box, or select the design element in the dialog box and then click **Go to location**.

Element	Description
Element Name	The name of the design element (for example, **Assign**).
Element Type	The type of design element (for example, **Regular Transition**).
Design Element	The design element that contains the design element being used (for example, **ChangeRequestAppWorkflow**).
Application/ Orchestration	The name of the *application* [page 679] or *orchestration* [page 690] that contains the design element (for example, **ChangeRequestApp**).

Appendixes

This section contains appendixes to the *SBM Composer Guide*.

Appendix 1: SBM JavaScript Library

The following topics provide a complete reference to the SBM JavaScript Library.

Overview

The SBM JavaScript library is automatically included with every custom form. You do not need to do anything to access this library.

JavaScript can be added to a custom form in two ways:

- Importing or editing a JavaScript file, adding it to the process app, and then adding it to a custom form. The advantage of using this method is that you can reuse the file in several forms.
- Adding an HTML/JavaScript *widget* [page 701] with the content of `<script>[your content]</script>` to a custom form. The advantage of using this method is that you can quickly edit and test the widget on a custom form, and later move its content to a JavaScript file to reuse on other custom forms.

SBM Composer uses the methods described in this document to support the rich interface process apps that you create using form widgets. You can also use these methods to create your own RIM behavior.

After you create a JavaScript file, you can import it by adding it to your custom form from the **JavaScripts** tab in the Property Editor for a form in SBM Composer.

Tips for Working with Fields

- Field names can be either the **Name** or **Database field name** specified in the Property Editor for the field. They can also be the name of a button or widget or other non-field control. The names must match the names in the Property Editor (for example, a **Database field name** must must be uppercase and cannot have spaces).
- When working in an environment where the field display names change often, it is best to use the **Database field name**.
- Forms must have fields visible and editable before you can use methods such as `ShowField`, `HideField`, `EnableField`, and `DisableField`. The JavaScript does not override what is set at the field or form level; it filters the form as it is.

Reference

The methods described in this section are divided into the following categories:

- Event Methods [page 650]
- Field Methods [page 655]

Event Methods

- AddLoadCallback [page 650]
- AddDelayCallback [page 651]
- AddClickCallback [page 652]
- AddChangeCallback [page 653]
- AddRadioCallback [page 654]
- AddSubmitCallback [page 654]

AddLoadCallback

Adds a callback that is invoked after the page loads.

Parameters

Name	Type	Description
`Callback`	Function	The function to be invoked when the page loads.

Return Value

Result	Value
(none)	

Examples

This example registers a named function to be called when the page loads:

```
function MyFunction()
{
    // do stuff on load
}

AddLoadCallback(MyFunction);
```

This example registers an anonymous function to be called when the page loads:

```
AddLoadCallback(
    function() {
```

```
        // do stuff on load
    }
);
```

Comments

Callbacks are invoked in the order in which they are added. This delays scripting until all of the page objects load. If this method is called after the page loads, it has no effect.

AddDelayCallback

Adds a callback that is invoked after the page loads and all of the load callbacks are invoked.

Parameters

Name	Type	Description
`Callback`	Function	The function to be invoked when the page loads and after the load callbacks are invoked.

Return Value

Result	Value
(none)	

Examples

This example registers a named function to be called when the page loads:

```
function MyFunction()
{
    // do stuff on delayed load
}

AddDelayCallback(MyFunction);
```

This example registers an anonymous function to be called when the page loads:

```
AddChangeCallback("Unit Price",
    function() {
        // do stuff on delayed load
    }
);
```

Comments

Callbacks are invoked in the order in which they are added. If this method is called after the page loads, it has no effect. This method is used to register the `AddClickCallback` and `AddChangeCallback` methods after the page loads.

AddClickCallback

Adds a callback that is invoked when the field or control is clicked.

Parameters

Name	Type	Description
`fieldName`	String	The name of the field to find.
`Callback`	Function	The function to be invoked when the object is clicked.

Return Value

Result	Value
(none)	

Examples

This example registers a named function to be called on click:

```
function MyFunction()
{
    // do stuff on click
}

AddClickCallback("MyButton", MyFunction);
```

This example registers an anonymous function to be called on click:

```
AddClickCallback("MyButton",
    function() {
        // do stuff on click
    }
);
```

Comments

Callbacks are invoked in the order in which they are added. When this method is called before the page loads, it is queued in the `AddDelayCallbacks` method and registered after the page loads.

AddChangeCallback

Adds a callback that is invoked when the field or control changes.

Restriction:

This method is not invoked for some field types when a user simply double-clicks a value or uses the right arrow to move a value from the left-hand box on a **ListBox** control to the right-hand box. For the method to be invoked, the user must also click the values in the right-hand box to select them. This restriction applies to the *Multi-Group*, *Multi-Relational*, *Multi-Selection*, and *Multi-User* field types.

Before this method can be invoked to populate a searchable *Multi-Group*, *Multi-Relational*, *Multi-Selection*, or *Multi-User* field, values must be present in the left-hand box on the **ListBox** control. Values are added to this box after the user clicks the **Find** button on the control.

Parameters

Name	Type	Description
`fieldName`	String	The name of the field to find.
`Callback`	Function	The function to be invoked when the field or control changes.

Return Value

Result	Value
(none)	

Examples

This example registers a named function to be called on change:

```
function MyFunction()
{
    // do stuff on change
}

AddChangeCallback("Unit Price", MyFunction);
```

This example registers an anonymous function to be called on change:

```
AddChangeCallback("Unit Price",
    function() {
        // do stuff on change
    }
);
```

Comments

Callbacks are invoked in the order in which they are added. When this method is called before the page loads, it is queued in the `AddDelayCallbacks` method and registered after the page loads.

AddRadioCallback

Adds a callback that is invoked when a radio button is clicked.

Parameters

Name	Type	Description
`objname`	String	The name of the field to find.
`Callback`	Function	The function to be invoked when the radio button is clicked.

Return Value

Result	Value
(none)	

Examples

This example registers a named function to be called on click:

```
function MyFunction()
{
    // do stuff on click
}

AddRadioCallback("Defect", MyFunction);
```

This example registers an anonymous function to be called on click of a radio button:

```
AddRadioCallback("Enhancement",
    function() {
        // do stuff on click
    }
);
```

Comments

Callbacks are invoked in the order in which they are added. When this method is called before the page loads, it is queued in the `AddDelayCallbacks` method and registered after the page loads.

AddSubmitCallback

Adds a callback that is invoked before the page is submitted.

Parameters

Name	Type	Description
`Callback`	Function	The function to be invoked before the page is submitted.

Return Value

Result	Value
True	Continue processing other submit callbacks. If a function does not return a value, it is assumed to be "true".
False	Stop processing other submit callbacks and cancel the submit operation.

Examples

This example registers a named function to be called before the page is submitted:

```
function MyFunction()
{
    // do stuff on submit
}

AddSubmitCallback(MyFunction);
```

This example registers an anonymous function to be called before the page is submitted:

```
AddSubmitCallback(function() {
    function() {
        // do stuff on submit
    }
);
```

Comments

Callbacks are invoked in the order in which they are added. This callback can be used to perform validation or other data verification. If you want to display a field validation error, call `MakeFieldInvalid` and return true from this callback so that any required or invalid field errors are automatically processed.

Field Methods

- MakeFieldInvalid [page 656]
- MakeFieldValid [page 657]
- GetFieldByName [page 657]
- GetFieldWidgetByName [page 658]
- GetLabelByName [page 659]

- IsFieldChecked [page 659]
- GetFieldValue [page 660]
- SetFieldValue [page 661]
- GetFieldValues [page 662]
- SetFieldValues [page 663]
- GetMultiListValues [page 663]
- SetMultiListValues [page 665]
- MakeFieldRequired [page 666]
- MakeFieldOptional [page 666]
- DisableField [page 667]
- EnableField [page 668]
- HideField [page 669]
- ShowField [page 669]
- HideSection [page 670]
- ShowSection [page 670]
- ExpandSection [page 671]
- CollapseSection [page 672]
- RefreshWidget [page 672]
- SetLabelText [page 673]
- GetLabelText [page 673]

MakeFieldInvalid

Marks the specified field as invalid, if it is not already marked as such. It does not update the form and the label. It displays an error on submit that prevents the form from being submitted.

Parameters

Name	Type	Description
`fieldName`	String	The name of the field.
`msg`	String	The message to display for this invalid field. The message is displayed when the user clicks **OK** in a *transition form* [page 700].

Return Value

Result	Value
(none)	

Examples

This example marks the *Unit Price* field as invalid:

```
MakeFieldInvalid("Unit Price", "The value in the 'Unit Price' field "
  + "must be greater than $5 and less than $500.")
```

Comments

(none)

MakeFieldValid

Marks the specified field as valid (that is, not invalid), if it is not already marked as such. It does not update the field or the label.

Parameters

Name	Type	Description
`fieldName`	String	The name of the field.

Return Value

Result	Value
(none)	

Examples

This example marks the Unit Price field as valid:

```
MakeFieldValid("Unit Price");
```

Comments

(none)

GetFieldByName

Finds an HTML DOM object that matches the specified name. Use this command to find the field to use in other JavaScript calls.

Note: Use the GetFieldWidgetByName [page 658] method for complex *field types* [page 686] (such as *Single Selection*, *Multi-Selection*, *Single Relational*, and *Multi-Relational*).

Parameters

Name	Type	Description
`fieldName`	String	The name of the field to find.

Return Value

Result	Value
Success	Field object corresponding to a field. This can be an `<input>`, `<select>`, `<textarea>`, or `<span>` element.
Failure	Null

Examples

This example returns the field object:

```
GetFieldByName("State");
```

Comments

(none)

GetFieldWidgetByName

Gets the wrapper element for the specified complex field type (such as *Single Selection*, *Multi-Selection*, *Single Relational*, and *Multi-Relational* fields) or form *widget* [page 701] (such as the PDF widget and REST Grid widget).

Parameters

Name	Type	Description
`objName`	String	The name of the complex field or form widget control.

Return Value

Result	Value
Success	The wrapper element for the complex field type or widget.
Failure	Null

Examples

This example returns the wrapper element for the "Testers" *Multi-Selection* field:

```
GetFieldWidgetByName("Testers");
```

Comments

(none)

GetLabelByName

Finds an HTML DOM object that matches the label for the specified name. Use this command to find the label to use in other JavaScript calls.

Parameters

Name	Type	Description
`objName`	String	The name of the field label to find.

Return Value

Result	Value
Success	Label object corresponding to a field. This is a <label> element.
Failure	Null

Examples

This example returns the label object:

```
GetLabelByName("Unit Price");
```

Comments

(none)

IsFieldChecked

Determines whether the check box is selected for the specified *Binary/Trinary* field with two possible values.

Parameters

Name	Type	Description
`fieldName`	String	The name of the field.

Return Value

Result	Value
Success	Returns `true` if the check box is selected, or `false` if the check box is cleared.
Failure	Returns `false` if the field is not found or if the field is not a *Binary/Trinary* field with two possible values.

Example

This example returns `true` if the check box is selected, or `false` if the check box is cleared.

```
IsFieldChecked("Regression");
```

Comments

(none)

GetFieldValue

Gets the value of the specified field.

Note: To determine whether the check box is selected for a *Binary/Trinary* field with two possible values, it is recommended that you use the IsFieldChecked [page 659] method, not this method.

Note: The SBM AppScript also has a `GetFieldValue` method.

Parameters

Name	Type	Description
`fieldName`	String	The name of the field.
`defaultValue` (optional)	String	The default value to return if the field does not exist on the form, if it is not found, or if its value is empty. If this parameter is not provided, "null" is returned.

Return Value

Result	Value

Success	String, date, or number matching the field type.  **Important:** A string of comma-separated values is returned for *Multi-Group*, *Multi-Relational*, *Multi-Selection*, and *Multi-User* fields. If the values have embedded commas, you must use the GetFieldValues [page 662] method instead.
Failure	Null or the default value

Examples

This example returns the string value of the field:

```
GetFieldValue("Unit Price");
```

Comments

(none)

SetFieldValue

Sets the value of the specified field.

 **Note:** *Sub-Relational* fields are not supported. These fields are driven by associated *Single Relational* or *Multi-Relational* fields, which can be set using the `SetFieldValue` method.

 **Note:** The SBM AppScript also has a `SetFieldValue` method.

Parameters

Name	Type	Description
`fieldName`	String	The name of the field.
`Value`	String	The value to set on the field. If the field is a Combo Box or List Box, the value must match the value of an item already in the list, and is case-sensitive. If the field is a *Multi-Group*, *Multi-Relational*, *Multi-Selection*, or *Multi-User* field, the value is a string of comma-separated values. If the values have embedded commas, you must use the SetFieldValues [page 663] method instead.
`fireEvent`	String	If this parameter is set to "true," then setting the values causes the change event to fire. If this parameter is set to "false," then the change event does not fire.

Return Value

Result	Value
(none)	

Examples

This example sets the string value of the field:

```
SetFieldValue("UNITPRICE", "1.45", "true");
```

Comments

(none)

GetFieldValues

Gets an array of values from the specified field.

Parameters

Name	Type	Description
`fieldName`	String	The name of the field.
`defaultValue` (optional)	String	The default value or array of values to return if the field does not exist on the form, if it is not found, or if its value is empty. If this parameter is not provided, "null" is returned.  **Note:** If the `defaultValue` is a single string value, it is converted to a single-element array.

Return Value

Result	Value
Success	An array of strings, dates, or numbers matching the field type.
Failure	Null or the default value

Examples

This example returns the array value of the field:

```
GetFieldValues("Reviewers");
```

This example returns a default array of values:

```
GetFieldValues("Reviewers", ["Tom Smith", "Susan Jones"]);
```

Comments

`GetFieldValues` is typically used with *Multi-Group*, *Multi-Relational*, *Multi-Selection*, and *Multi-User* fields. If this method is used with other *field types* [page 686], it returns an array with a single value.

Before this method can be invoked to populate a searchable *Multi-Group*, *Multi-Relational*, *Multi-Selection*, or *Multi-User* field, values must be present in the left-hand box on the **ListBox** control. Values are added to this box after the user clicks the **Find** button on the control.

SetFieldValues

Sets an array of values for the specified field.

Parameters

Name	Type	Description
`fieldName`	String	The name of the field.
`value`	Array	The array of values to set on the field. If the field is a **Combo Box** or **List Box**, the value must match the value of an item already in the list, and is case-sensitive.

Return Value

Result	Value
(none)	

Examples

This example sets the array of string values for the field:

```
SetFieldValues("Numbers", ["one", "two", "three", "four", "five"]);
```

Comments

`SetFieldValues` is typically used with *Multi-Group*, *Multi-Relational*, *Multi-Selection*, and *Multi-User* fields. If this method is used with other *field types* [page 686], it sets the field to the first value in the array.

GetMultiListValues

Gets an array of values from the specified **List Box** control.

Note: This method supports only *Multi-Group*, *Multi-Relational*, *Multi-Selection*, and *Multi-User* fields.

Parameters

Name	Type	Description
`objName`	String	The name of the **List Box** control.
`defaultValue` (optional)	String	The default value or array of values to return if the user did not select at least one list box value, or if the control does not exist on the form. If this parameter is not provided, "null "is returned.  **Note:** If the `defaultValue` is a single string value, it is converted to a single-element array.
`asValue` (optional)	String	If this parameter is set to "true," then the actual array of values as set in the HTML element are returned instead of display values. If this parameter is set to "false" or is not used, then an array of display values is returned. In the `<option...value="NEW_YORK">New York</option>` example, New York is the display value, and NEW_YORK is the actual value.

Return Value

Result	Value
Success	An array of strings representing the display value or actual value of each selected item in the **List Box** control.
Failure	Null or the default value.

Examples

This example returns an array of actual values based on items the user selected in the **Sites** list box.

```
GetMultiListValues("Sites", "New York", true);
```

Comments

Before this method can be invoked to populate a searchable *Multi-Group*, *Multi-Relational*, *Multi-Selection*, or *Multi-User* field, values must be present in the left-hand box on the **ListBox** control. Values are added to this box after the user clicks the **Find** button on the control.

SetMultiListValues

Sets an array of values in the specified **List Box** control.

Note: This method supports only *Multi-Group*, *Multi-Relational*, *Multi-Selection*, and *Multi-User* fields.

Parameters

Name	Type	Description
`objName`	String	The name of the **List Box** control.
`values`	Array	An array of values to set in the **List Box** control.
`fireEvent`	String	If this parameter is set to "true," then setting the values causes the change event to fire. If this parameter is set to "false," then the change event does not fire.
`asValue` (optional)	String	If this parameter is set to "true," then the actual array of values as set in the HTML element are returned instead of display values. If this parameter is set to "false" or is not used, then an array of display values is returned. In the `<option...value="NEW_YORK">New York</option>` example, New York is the display value, and NEW_YORK is the actual value.

Return Value

Result	Value
(none)	

Examples

This example sets an array of display values based in the **Sites** list box.

```
SetMultiListValues("Sites", ["New York", "San Francisco", "Chicago",
→"Los Angeles", "Seattle"], true, false);
```

Comments

(none)

MakeFieldRequired

Marks the specified field as required, if it is not already marked as such. It updates the field and the label appropriately.

Note: This method can be used only on a field that is set in SBM Composer or SBM System Administrator to be not required, not read-only, and in a field section that is visible in the SBM User Workspace. It is used with the `MakeFieldOptional` method to dynamically change the required or optional state of a non-required field on a form.

Parameters

Name	Type	Description
`fieldName`	String	The name of the field.
`bShowIfHidden` (optional)	Boolean	Show the field if it was hidden by the `HideField` method. The default value is true.

Return Value

Result	Value
(none)	

Examples

This example marks the *Unit Price* field as required:

```
MakeFieldRequired("Unit Price");
```

This example marks the *Unit Price* field as required but does not show it if it is hidden:

```
MakeFieldRequired("Unit Price", false);
```

Comments

(none)

MakeFieldOptional

Marks the specified field as optional (that is, not required), if it is not already marked as such. It updates the field and the label appropriately.

Note: This method can be used only on a field that is set in SBM Composer or SBM System Administrator to be not required, not read-only, and in a field section that is visible in the SBM User Workspace. It is used with the `MakeFieldRequired` method to dynamically change the optional or required state of a non-required field on a form. This method does not apply to fields originally set as required on a workflow or transition level. It applies only to fields set as required with the `MakeFieldRequired` method.

Parameters

Name	Type	Description
`fieldName`	String	The name of the field.
`bShowIfHidden` (optional)	Boolean	Show the field if it is currently hidden. The default value is true.

Return Value

Result	Value
(none)	

Examples

This example marks the *Unit Price* field as required:

```
MakeFieldOptional("Unit Price");
```

This example marks the *Unit Price* field as required but does not show it if it is hidden:

```
MakeFieldOptional("Unit Price", false);
```

Comments

(none)

DisableField

Marks the specified field as disabled/read-only, if it is not already marked as such. It updates the field only, not the label.

Note: This method is used with the `EnableField` method to dynamically change the disabled or enabled state of a non-required field on a form. This method does not apply to fields originally set as enabled on a workflow or transition level. It applies only to fields set as enabled with the `EnableField` method.

Parameters

Name	Type	Description
`fieldName`	String	The name of the field.
`bShowIfHidden` (optional)	Boolean	Show the field and the label if they are currently hidden. The default value is true.

Return Value

Result	Value
(none)	

Examples

This example marks the *Unit Price* field as disabled/read-only:

```
DisableField("Unit Price");
```

Comments

(none)

EnableField

Marks the specified field as enabled (that is, not disabled), if it is not already marked as such. It updates the field only, not the label.

Note: This method is used with the `DisableField` method to dynamically change the disabled or enabled state of a non-required field on a form. This method does not apply to fields originally set as disabled on a workflow or transition level. It applies only to fields set as disabled with the `DisableField` method.

Parameters

Name	Type	Description
`fieldName`	String	The name of the field.
`bShowIfHidden` (optional)	Boolean	Show the field and the label if they are currently hidden. The default value is true.

Return Value

Result	Value
(none)	

Examples

This example marks the *Unit Price* field as enabled:

```
EnableField("Unit Price");
```

Comments

(none)

HideField

Marks the specified field as hidden, if it is not already marked as such. It updates the field and the label appropriately.

Parameters

Name	Type	Description
`fieldName`	String	The name of the field.
`bHideFieldOnly` (optional)	Boolean	Hide the field only. The label remains visible. The default value is false.

Return Value

Result	Value
(none)	

Examples

This example marks the *Unit Price* field and its label as hidden:

```
HideField("Unit Price");
```

This example marks the *Unit Price* field as hidden, but the label remains visible:

```
HideField("Unit Price", true);
```

Comments

(none)

ShowField

Marks the specified field as visible, if it is not already marked as such. It updates the field and the label appropriately.

Note: This method does not apply to hidden fields (fields that belong in the Hidden section of a form).

Parameters

Name	Type	Description
`fieldName`	String	The name of the field.
`bShowFieldOnly` (optional)	Boolean	Show the field only. The label remains hidden. The default value is false.

Return Value

Result	Value
(none)	

Examples

This example marks the *Unit Price* field and its label as visible:

```
ShowField("Unit Price");
```

This example marks the *Unit Price* field as visible; however, the field's label remains hidden.

```
ShowField("Unit Price", true);
```

Comments

(none)

HideSection

Marks the specified section or tab as hidden, if it is not already marked as such.

Parameters

Name	Type	Description
`sectionName`	String	The name of the section or tab.

Return Value

Result	Value
(none)	

Examples

This example marks the **Notes** section as hidden:

```
HideSection("Notes");
```

Comments

This currently works only with sections and tabs.

ShowSection

Marks the specified section or tab as visible, if it is not already marked as such.

Parameters

Name	Type	Description
`sectionName`	String	The name of the section or tab.

Return Value

Result	Value
(none)	

Examples

This example marks the **Notes** section as visible:

```
ShowSection("Notes");
```

Comments

This currently works only with sections and tabs.

ExpandSection

Expands the specified section, if it is not already expanded.

Parameters

Name	Type	Description
`sectionName`	String	The name of the section.

Return Value

Result	Value
(none)	

Examples

This example expands the **Notes** section:

```
ExpandSection("Notes");
```

Comments

This currently works only with sections.

CollapseSection

Collapses the specified section, if it is not already collapsed.

Parameters

Name	Type	Description
`sectionName`	String	The name of the section.

Return Value

Result	Value
(none)	

Examples

This example collapses the **Notes** section:

```
CollapseSection("Notes");
```

Comments

This currently works only with sections.

RefreshWidget

Refreshes the specified *widget* [page 701]. This causes the widget to be refreshed or reloaded. New values are obtained and passed to the widget to display the results.

Parameters

Name	Type	Description
`widgetName`	String	The name of the widget.

Return Value

Result	Value
(none)	

Examples

This example refreshes the widget named PDF:

```
RefreshWidget("PDF");
```

Comments

(none)

SetLabelText

Sets the current text of the specified label.

Parameters

Name	Type	Description
`fieldName`	String	The name of the field.
`Text`	String	The text to set on the label. This method automatically adds a colon (`:`) and an asterisk `*` (for "required") as appropriate.

Return Value

Result	Value
(none)	

Examples

This example sets the text of the label:

```
SetLabelText("Unit Price", "Price/Unit");
```

Comments

(none)

GetLabelText

Gets the current text of the specified label.

Parameters

Name	Type	Description
`fieldName`	String	The name of the field.

Return Value

Result	Value
Success	The original value for the label on the form, or the value last set with the `SetLabelText` method.

Failure	Null

Examples

This example gets the text of the label:

```
GetLabelText("Unit Price");
```

Comments

(none)

JavaScript Examples

- Setting Field Properties Based on Field Values [page 674]
- Changing Field Properties Based on Date Change [page 676]
- Changing Field Properties Based on Field Value Length [page 677]
- Marking a Field as Optional or Required [page 677]

Setting Field Properties Based on Field Values

To recreate these examples, your *application* [page 679] should contain the following fields and they should be added to your custom form:

Field type	Database name	Display name	Values
`Single Selection`	REQUEST_TYPE	Request Type	Billable Nonbillable
`User`	ACCOUNT_MANAGER	Account Manager	Users assigned to roles specified for this field or added as values for the field in SBM System Administrator.
`Text`	SOW_NUMBER	SOW Number	Specified by users in the SBM User Workspace.
`Binary`	SALES_APPROVAL	Sales Approval	Yes No
`User`	SALES_MANAGER	Sales Manager	Users assigned to roles specified for this field or added as values for the field in SBM System Administrator .

Example 1

In this example, you can set the *Account Manager* and *SOW Number* fields as required when users select **Billable** from the *Request Type* field. If another value is selected from

the *Request Type* field, the *Account Manager* and *SOW Number* fields are removed from the form, and are made optional.

```
function RequestTypeChanged()
{
    var text = GetFieldValue("REQUEST_TYPE");
    if (text == "Billable") {
        MakeFieldRequired("ACCOUNT_MANAGER", true);
        MakeFieldRequired("SOW_NUMBER", true);
    }
    else {
        HideField("ACCOUNT_MANAGER");
        HideField("SOW_NUMBER");
        MakeFieldOptional("ACCOUNT_MANAGER");
        MakeFieldOptional("SOW_NUMBER");
    }
}

AddChangeCallback("REQUEST_TYPE", RequestTypeChanged);
```

Example 2

In this example, you can set the *Account Manager* and *SOW Number* fields as required and display the *Sales Approval* field when users select **Billable** from the *Request Type* field. If another value is selected from the *Request Type* field, the *Account Manager*, *SOW Number*, and *Sales Approval* fields are removed from the form, and *Account Manager* and *SOW Number* fields are made optional. In addition, when users select **Yes** in the *Sales Approval* field, the *Sales Manager* field appears on the form. If another value is selected, the *Sales Manager* field is removed from the form. **Note:** The *Sales Approval* field must be a drop-down list. It cannot be a radio button or a check box.

```
function RequestTypeChanged()
{
    var text = GetFieldValue("REQUEST_TYPE");
    if (text == "Billable") {
        MakeFieldRequired("ACCOUNT_MANAGER", true);
        MakeFieldRequired("SOW_NUMBER", true);
        ShowField("SALES_APPROVAL");
    }
    else {
        HideField("ACCOUNT_MANAGER");
        HideField("SOW_NUMBER");
        HideField("SALES_APPROVAL");
        MakeFieldOptional("ACCOUNT_MANAGER");
        MakeFieldOptional("SOW_NUMBER");
    }
}

AddChangeCallback("REQUEST_TYPE", RequestTypeChanged);

function SalesApprovalChanged()
{
    var text = GetFieldValue("SALES_APPROVAL");
    if (text == "Yes") {
```

```
        ShowField("SALES_MANAGER");
    }
    else {
        HideField("SALES_MANAGER");
    }
}

AddChangeCallback("SALES_APPROVAL", SalesApprovalChanged);
```

Changing Field Properties Based on Date Change

To recreate these examples, your *application* [page 679] should contain the following fields and they should be added to your custom form:

Field type	Database name	Display name	Values
Date/ Time	DUE_DATE	Due Date	On the **Options** tab, select **Date Only**.
User	EMRG_APPROVER	Emergency Approver	Users assigned to roles specified for this field or added as values for the field in SBM System Administrator.
Text	EMRG_REASON	SOW Number	Specified by users in the SBM User Workspace.

Example

In this example, when users specify a value in the *Due Date* field that is fewer than seven days from the current date, the *Emergency Approver* and *Emergency Reason* fields are set as required. If the *Due Date* field value is more than seven days past the current date, the *Emergency Approver* and *Emergency Reason* fields are removed from the form, and users are informed that the due date is too far in the future.

```
function DueDateChanged()
{
    var text = GetFieldValue("DUE_DATE");
    var DueDate = new Date();
    DueDate.setTime(Date.parse(text));

    var NextWeek = new Date();
    NextWeek.setDate(NextWeek.getDate()+7);

    if (DueDate < NextWeek) {
        MakeFieldRequired("EMRG_APPROVER", true);
        MakeFieldRequired("EMRG_REASON", true);
    }
    else {
        HideField("EMRG_APPROVER");
        HideField("EMRG_REASON");
    }
```

```
    var FixedDate = new Date();
    FixedDate.setFullYear(2015,1,1);
    if (DueDate > FixedDate) {
        alert('Date too far in the future!');
    }
}

AddChangeCallback("DUE_DATE", DueDateChanged);
```

Changing Field Properties Based on Field Value Length

To use the following example, verify that the system *Description* field has been added to your *application* [page 679] and is available on your custom form. This example also uses the system *Title* field, which is automatically added to all applications.

Example

In this example, the *Description* field is set as required if the *Title* field contains fewer than 20 characters.

```
function TitleChanged()
{
    var text = GetFieldValue("TITLE");
    if (text.length < 20) {
        MakeFieldRequired("DESCRIPTION");
    }
    else {
        MakeFieldOptional("DESCRIPTION");
    }
}

AddChangeCallback("TITLE", TitleChanged);
```

Marking a Field as Optional or Required

In this example, the *Description* field is optional if the *Request_Amount* field is less than $1000.00. This example assumes that the *Request_Amount* field is numeric. If it is a string, it must be converted into a number before the comparison is made.

Example

For this example, the *Description* field must not be set as required in SBM Composer or overridden in SBM System Administrator.

```
function RequestAmountChanged()
{
    var amount = GetFieldValue("REQUEST_AMOUNT");
    if (amount < 1000.00) {
        MakeFieldOptional("DESCRIPTION");
    }
    else {
        MakeFieldRequired("DESCRIPTION");
    }
```

```
}

AddChangeCallback("REQUEST_AMOUNT", RequestAmountChanged);
```

Glossary

Action

A predefined activity tied to a transition or state. When the transition occurs, so does the associated action; or when the state is entered or exited, the associated action occurs. An action can cause the transition of a related item, such as a child or subtask.

Advanced SQL Query

SQL, or Structured Query Language, is an ANSI-standard language for selecting records from a database. The Advanced SQL reports in the SBM User Workspace provide two ways for users to implement or control SQL: SBM Processed SQL and Pass-Through SQL.

Anonymous Submit

Allows users who do not have SBM user accounts to submit items into selected projects.

"Any" State

A special state provided by the system that enables designers to create transitions from every state in the workflow.

AppCentral™

A channel containing process app templates that enable process app users to share process apps. It helps users build process apps quickly and easily by providing a tight integration with SBM Composer. The templates in the AppCentral™ are organized by category.

Application

A collection of elements that work together in an interactive business process to solve a business requirement, such as managing a team's work tasks or tracking customer support calls. Applications typically contain workflows, fields, forms, roles, projects, reports, and notifications. Applications can include *orchestrations* that emit events that execute activities in other tools or applications. Orchestrations can be used to combine applications and create *process apps*.

Application Report

Listing reports that are defined in SBM Composer along with other process app artifacts. (Listing reports return textual lists of primary or auxiliary items based on the display, sorting, and search options that are selected.) Application reports are a starting point for users to pre-configure regular reports. In the SBM User Workspace, users can generate any number of regular reports from a single application report.

Application Table

This generally refers to a primary table associated with a given application or applications.

Application Tabs

Application tabs filter the buttons on the **Application** toolbar for each application in the SBM User Workspace.

Application Workflow

A collection of states, transitions, and fields that define an interactive business process. All primary items follow a workflow process.

Archive Wizard

Enables administrators to archive inactive primary items and change history information for active and inactive items. Archived items are stored in special archive tables in the database. The archive feature allows archived items to be restored and purged.

Audit Log

A historical record of *deployments*, *promotions*, and *application* changes shown in *Application Administrator*.

Auxiliary Item

Refers to items that are stored in auxiliary tables. Auxiliary items support, but do not follow, an *application workflow* process. Auxiliary table records are useful because they store auxiliary information that can be referred to repeatedly by one or more *applications*. Contact and company records are examples of auxiliary items.

Auxiliary Relational Field

A Relational field that refers to an auxiliary table. Users can select values from the auxiliary table for fields contained in other primary or auxiliary tables.

Auxiliary Table

Auxiliary tables store information such as contacts, companies, problems, and resolutions. Auxiliary tables do not follow a workflow process, but are ideal for storing information needed repeatedly.

BPEL

See *Business Process Execution Language.*

BPEL Engine

See *BPEL Server*.

BPEL Server

A software component that parses BPEL code, creates specific instances of the processes described by that code, and makes the processes available as Web service *endpoints*. Also manages any needed execution details, such as state information.

Base Project

A header project located at the root level of the project hierarchy. The base project cannot be assigned a workflow or contain primary items.

Base Workflow

A workflow that always exists at the root level of the workflow hierarchy in SBM System Administrator. The base workflow serves as a placeholder for all workflows in the system and it cannot be edited, deleted, or used by any project. In SBM Composer, the base workflow is simply called a workflow, which distinguishes it from its *sub-workflows*.

Basic Condition

A report query that allows users to define the report criteria by making selections from the list of fields.

Blueprint File

A file that is created by SBM Composer and that contains process app design elements. It contains roles, scripts, icons, tables, workflows, and orchestrations. This file is deployable to any other database.

Broker Service

See *Serena Broker Service*.

Business Process Execution Language (BPEL)

An Oasis standard XML-based language that is used to describe business processes that are executed across the Web through the coordination of Web services.

Calculated Field

Provides a way to measure time intervals between transitions or perform numeric calculations. This feature is available with Date/Time and Numeric fields, but is implemented for transitions.

Change History

The history of changes that occur as primary items move through the workflow or as auxiliary items are updated. By default, the Change History section appears at the bottom of the Item Details pane.

Check In

Commits changes made to a process app or other design element to the SBM Application Repository, and makes it available for *check-out* by other designers.

Check Out

Creates a new revision of a process app or other design element and copies the current version from the SBM Application Repository to a designer's local work area. The current version in the SBM Application Repository is locked so that no one else can check it out and change it.

Child Project

A project derived from an existing project. Child projects are also referred to as sub-projects.

Child Workflow

See *Sub-workflow*.

SBM Common Services

The component that enables external data from disparate domains to be loaded and manipulated in REST Grid Widgets. Also powers the PDF Widget, which generates PDF documents from data in the SBM User Workspace.

Conditional Routing

Enables designers in SBM Composer to create application workflows that automatically route an item to a particular state based on visually designed rules. The rules evaluate item data and determine which transition will be executed. The rules are mapped to one or more **Decision** steps in the workflow.

Copy Transition Type

Enables users to make a copy of one primary item and place it in another project.

Create Subtask Transition

Allows users to create a new primary item in a specified project. Designers can specify that the new item be linked to the original, or principal, item and that subtasks are transitioned according to values in a specific Binary/Trinary field.

Cross-Database Posting

Enables users to post primary items to another SBM database.

Custom Fields

A set of field types designers can use to create their own custom fields for data collection. Custom fields can be added to primary and auxiliary tables.

Database Lock

A feature that enables you to protect the integrity of a database so that changes cannot be made to application elements in SBM System Administrator. This prevents these changes from being overwritten during promotion to this database from another environment.

Default Fields

Each workflow and project contains a list of default fields. For workflows, the default fields list is determined by the primary table on which the workflow is based. For projects, the default fields list is determined by the workflow to which the project is assigned.

Default Weight

Give numeric value to selections for Single Selection fields. Numeric values can be used with Summation fields to sum the values contained in the field.

Dependent Fields

Autopopulated fields that tailor the selection lists for dependent Single Selection, Multi-Selection, Single Relational, Multi-Relational, Multi-Group, Multi-User, and User fields. When users select a value from one field, the values available in a dependent field are limited to those specified in the dependency.

Deployment

The process by which a developer or an administrator makes a *process app* defined in *SBM Composer* available on a host *environment*. The deployment can be performed from SBM Composer or Application Administrator.

Design Element

A component of a *process app*, such as a workflow, field, form, table, action, or an *orchestration*. Design elements created in *SBM Composer* can be saved locally or checked in to the *SBM Application Repository*.

Designer

The person who uses SBM Composer to design workflows, data tables and fields, roles, forms, and other design elements. Designers develop applications and orchestrations, linking the different processes together within a process app. Designers are also referred to as "application developers."

Development Deployment

Enabling *development deployment* means that an SBM Composer user can deploy development versions of a process app to the environment for testing purposes without generating new versions of the process app.

Dynamic Column Sorting

Enables users to quickly sort columns that appear in Listing reports. Columns can be sorted in ascending or descending order.

E-mail Recorder

Automatically attaches e-mail messages sent from external e-mail clients as notes to the primary and auxiliary items from which messages were sent. The E-mail Recorder also attaches replies to these messages to items. In addition, replies to e-mail notifications can be attached to the items to which they pertain.

"E-mail" State

A state used as a starting point in a workflow for e-mail submissions or external posting of items. The "E-mail" state is available when designers set up transitions.

E-mail Submission

Allows users to submit primary items via e-mail.

E-mail Template Tags

Used to access pieces of information administrators may want to place in e-mail messages sent by the system. When the e-mail message is sent, each tag is replaced by information it represents.

E-mail Template

Used to standardize and customize messages sent for e-mail notifications, self-registration confirmation and password change notices, and e-mail submissions. E-mail templates used to generate messages sent from the SBM User Workspace can also be customized.

Elapsed Time

A function found within a Date/Time field that stores a value representing elapsed time in days, hours, minutes, and seconds.

Endpoint

An entity, processor, or resource that can be addressed with a Web service message. For example, an endpoint might be a URL that specifies a Web service or a supported *BPEL engine*. Endpoints have a type, a protocol, and possibly credentials necessary to connect to the service. Endpoints defined for a *process app* must be resolved before the *process app* can be deployed to an environment. Once the process app is deployed, SBM uses these endpoints to communicate with the SBM Orchestration Engine and external services. The orchestration event manager uses the endpoints to call the proper BPEL engine.

Environment

A group of runtime hosts to which *process apps* are deployed or promoted. An environment consists of at least one SBM host, and possibly one or more orchestration servers and Web service *endpoints*.

Environment Set

A collection of related environments to which you want to deploy process apps and among which those process apps can be promoted. For example, an environment set might include Development, Testing, and Production environments for a single SBM system; your company might have multiple environment sets. Note that the Global Process App (and any process app containing the Global Application) typically should be deployed *only* to environments in the same environment set from which it originated.

Escalation

The ability of one notification to generate another notification when specific conditions occur or when a specified time interval elapses. The notification that is sent is referred to as an *escalation notification*.

Escalation Notification

A notification that is sent by another notification when specific conditions occur or when a specified time interval elapses.

Event

A Web service message signaling a meaningful change from an application or external product. For example, a defect tracking application might emit an event every time a user enters a new defect. A external build product that is event enabled could emit a "Build_Completed" event. When a process app event is received by the Event Manager, the SBM Orchestration Engine is called to execute the workflow associated with the event.

Event definition

A file with a specific format that lets SBM applications and external products declare the events they can raise. A event definition also lets receiving applications understand the events so they can respond to them. Event definitions that define events so that other process apps can respond to them are known as *event definitions* and have a `.mtd` file extension. Event definitions that raise external events from an external product are known as *custom event definitions* and have a `.wsdl` file extension.

Event-Enabled

A Serena Business Manager application or a third-party software tool that is capable of emitting events.

Event Manager

A component that responds to events sent by event-enabled tools. The Event Manager calls orchestration workflows implemented as BPEL processes.

Event Manager Log

A record of events raised by the *Event Manager* and *SBM* as shown in *Application Administrator*.

Export

The process in *SBM Composer* by which a designer stores a *process app* on the file system. The resulting *blueprint file* can be reimported into SBM Composer for continued development or imported into *Application Administrator* for deployment. See also *Publish*.

External User Access

Provides minimal set of privileges to users, such as customers or contacts.

External Users

Users who have a minimal set of privileges that allows them to submit and view certain items.

Favorite Folders

Enable users to add links to frequently used features, forms, items, and reports in folders that they create or that are provided by the system. Favorites provide a personal view of items in the system; users can view only their own favorites, not other users' favorites.

Field Sections

Provide a way to organize fields on state and transition forms in the SBM User Workspace. Organizing fields in sections provides a way to group fields for display, as well as to limit user accessibility to certain fields. When quick forms are used, field sections are used to control security and field organization. When custom forms are used, field sections control user access to fields.

Field Types

The types of fields that can be added to primary and auxiliary tables. Binary/Trinary, Multi-Selection, Single-Selection, or Summation fields are examples of the types of fields that can be added.

Field Value Searching

Enables users to locate values for selection fields, such as User fields or Multi-Relational fields. This is useful for finding the correct value for a field that may have many selections. Two types of Field Value Search options are available: *Value Find* and *Relational Field Value Lookup*.

File Associations

File associations are similar to version control actions, which display the history of source-code control operations associated with primary items. File associations differ, however, because users can add, modify, and delete them from the SBM User Workspace.

Forms

The pages in the SBM User Workspace in which users submit, transition, and update items. Designers can use *quick forms* or create custom forms for states and transitions.

From State Property

Designates the state from which a transition originates.

Get Latest

Retrieves a read-only copy of the latest revision of the selected *process app*, *application*, *orchestration*, or other *design elements* from the *SBM Application Repository* to a designer's *Local Cache*; does not create a new revision of the files. Compare to *Check Out*.

Global Administrator

Global administrators have full access to and are responsible for managing the entire SBM system.

Global Application

A special *application*, contained in the *Global Process App*, that initially includes the seven system auxiliary tables and their associated default icons. In a system upgrade, the Global Application initially also contains any other existing auxiliary tables that are not associated with a single specific application and any existing scripts and triggers used in transition actions. You can also add to the Global Application any other auxiliary tables (and associated design elements—images, Javascripts, roles, styles, and table forms) you create that you want to make available for use in multiple applications. The Global Application is also where you create scripts and triggers to be used in transition actions.

Global Process App

A special *process app*, included automatically in every SBM database, that contains the *Global Application*. Creating a new *environment* or upgrading an existing environment typically includes getting the Global Process App into Application Administrator, so the contents of the Global Application are available for use in other process apps. You gain access to those contents by defining a reference to the Global Process App in any process app in which you want to use them.

Global Search

Allows users to search for items in multiple primary and auxiliary tables at once. Global searches can be saved as Multi-Table reports.

Group-Specific Transition

Enables administrators to limit the groups whose members can execute a particular transition.

Hierarchy Structure

A graphical representation of parent and child workflows, projects, and folders. Elements of workflows and projects can be inherited throughout the hierarchy structure.

ID Search

Enables users to search for primary items by Item ID in the selected application.

Inheritance Rules

Provide the ability to create and customize elements of the tracking system that are inherited throughout a hierarchical tree structure.

Item

A generic term for primary and auxiliary items. See also *Primary Item* and *Auxiliary Item*.

Item Notifications

Provides a way for users to subscribe to e-mail notifications pertaining to an individual primary or auxiliary item.

Item Type

Describes a primary item, such as a defect or change request, tracked in a workflow. The list of item types is populated from the selections list for the **Item Type** field.

Journal Field

Journal fields automatically stamp a Text field with the date/time and user ID. Designers can specify that users must append the text in a new Journal entry to an existing entry. This prevents users from modifying existing Journal entries. Designers can also allow users to edit existing Journal entries.

Knowledge Base

A collection of information stored in the Problems and Resolutions tables. Links to problem and resolution records are contained in folders, which can be accessed for internal and public viewing by SBM users and anonymous users.

Launch Page

The default SBM User Workspace home page for users. It provides links to frequently used information and features. From the Launch page, users can view items they own, perform common tasks, and view their favorites.

License Server

The Serena License Manager allows administrators to centralize license management across multiple Serena software tools. The License Server helps administrators keep track of active licenses and versions of software used.

Local Cache

The special area on a designer's computer file system where SBM Composer stores process apps and other design elements whenever the designer works on them, both before they are checked in to the SBM Application Repository and any time they're checked out. Each Windows login ID has a separate area allocated automatically (by Windows) for its own Local Cache. Compare to *SBM Application Repository*.

Locks

Three types of locks are provided: record, administrative, and database locks. Record locks block access to an auxiliary or primary item while a user is updating or transitioning it. Administrative locks ensure only one administrator is editing a specific portion of the system at any given time. Database locks enable you to manually lock a database to prevent changes from being overwritten during promotion activities.

Mail Client

The Mail Client controls the e-mail submission of primary items, posting of items between SBM databases, and the E-mail Recorder.

Managed Administrator

An administrator granted restricted access to certain administrative features. For example, managed administrators can edit projects and workflows; add, delete, or edit user and group accounts; manage notifications, etc.

Managed Global Administrator

A managed administrator who manages other managed administrators.

Mass Update

Mass updates allow users to transition, update, and delete multiple primary items at the same time. Mass updates can also be used to update and delete multiple auxiliary items.

Named License

Permits the use of one user license on a single workstation at one time.

Navigation Pane

Located in the left pane of the SBM User Workspace, the navigation pane provides easy access to active items in selected projects, reports, favorites, public, and Knowledge Base views.

"None" State

A special state used to define a Submit transition.

Notification Rules

Conditions that cause notifications to be executed.

Notification Server

The server used to generate notifications when certain events or conditions occur in the system. The Notification Server is also used to send all out-going e-mail, such as e-mail sent by users from the SBM User Workspace, e-mail notifications, mail client responses, and to automatically call Web service functions or execute scripts.

Notifications

E-mail messages sent to users when certain events or conditions occur in the system. Notifications can also be used to automatically add and remove items from folders and to execute scripts.

Occasional User Access

Provides the same set of privileges as the External User access type, but also enables users to own items they submit, update items they own, transition items they own, and update their user profile. Occasional users can also be granted privileges to view all field sections for items they submitted. This access type is only available with the use of seat-based licenses or with SBM On-demand.

ODBC

ODBC (Open Database Connectivity) data source contains connection information and other database specific parameters necessary to connect to a database.

Orchestration

A container to collect related *orchestration workflows* as part of a *process app*.

Orchestration Workflow

A sequenced arrangement of Web service calls designed using *SBM Composer*. Orchestration workflows combine Web services using loops and decision branches and define the way data is mapped between the Web services. The final arrangement is saved as a BPEL process. Orchestration workflows can be linked to *application workflows* in a *process app* through *actions* on both *transitions* and *states*. Orchestration workflows can run asynchronously using an *event*, or they can be called synchronously, where the *action* waits for the orchestration workflow to return some data. When designed to be used with an *event*, orchestration workflows can also be invoked by external systems.

Overrides

A means of configuring elements of workflows and projects. Overrides can be applied at the workflow or project level to the inherited order and properties of Default fields, Transition fields, and State fields. Overrides can also be applied to transition properties.

Ownership

Ownership provides accountability for an item during the workflow process. A primary owner assigned to the item is responsible for the item while it resides in a particular state; secondary owners can have secondary responsibility for items as well.

Parent Project

Projects that contain child projects or sub-projects are referred to as parent projects.

Parent Workflow

Workflows that contain sub-work.flow (child workflows).

Parent-Child Actions

Actions that transition items that are selected as values in a Primary Relational field or in which a Primary Relational field is the selected value. The transition can be executed unconditionally or can be based on the values for a Binary/Trinary field in either the original item or the item in the Relational table.

Pass-Through SQL

Used with Advanced SQL reports, pass-through SQL allows users to implement sub-queries to system tables or to other user tables. In addition, the full power of SQL (functions, aggregates, etc.) is available.

Patch Context

A patch context allows parallel and concurrent development to occur on the same *process app*, so that patches can be applied to running process apps as ongoing development continues. When performing maintenance work, a designer can create a patch context from a "snapshot" of the process app. The chosen snapshot serves as the baseline for the maintenance work. When a designer is working in a patch context, any changes apply only to the patch context and do not affect ongoing development of the "baseline" version of the process app. Any number of designers can do concurrent development in a patch context.

Post Item Transition Type

Enables users to post an item from an existing primary item to another project or to another primary or auxiliary table.

Preferences

The SBM User Workspace options set for users. Preferences are set on the Preferences tab page in the SBM System Administrator or Web Administrator. With the correct privilege, users can modify their user profile in the SBM User Workspace.

Preferred Application

Determines which application tab is selected when users launch the SBM User Workspace.

Preferred Projects

Enable users to limit the list that appears in project lists in the SBM User Workspace. This enables users to quickly submit into, search, and create reports against projects that they most frequently use. When users specify preferred projects, they can easily switch between the full list of projects they can work with and their preferred project list.

Primary Item

Items, such as issues and change requests, that are submitted into projects in the SBM User Workspace and that follow an *application workflow* process.

Primary Owner

To provide accountability for primary items, a primary owner is designated for each state in the workflow. Users can only select a single user as a primary owner for each state the item resides in. This ensures that primary items always have one user who is primarily responsible for them while they resides in each state.

Primary Relational Field

A Relational field in any application (except the Global Application) that refers to a primary table. Users can select values from the primary table for fields contained in other primary or auxiliary tables.

Primary Table

A table that stores information about primary items, which follow an *application workflow* process.

Principal Item

A primary item that has subtasks associated with it. Subtasks are created using the Create Subtask transition type or are manually linked together by users in the SBM User Workspace.

Privileges

Define the information and features users, members of groups, and users assigned to roles can access. The types of privileges that can be granted depend on the type of product access the user has been assigned.

Process app

A process app comprises one or more *applications* and *orchestrations*.

Processed SQL

Used with Advanced SQL reports, SBM attempts to process conditional expressions included in SQL statements.

Project

A collection of primary items that follow a workflow process. Projects are displayed in a hierarchy, with each level of the hierarchy representing a different project. A project is a means for organizing items.

Project Hierarchy Structure

A graphical representation of parent projects and child or sub-projects. The Base Project exists at the root level of the hierarchy.

Promotion

The process by which administrators can replicate a *process app* from one host *environment* to another or move a fully configured process app into a new environment.

Public Folders

Contain links to items such as reports, which can be accessed by users who have been granted appropriate privileges. Notifications can also be created to automatically add items to or remove items from public folders.

Publish

The process in *SBM Composer* by which a designer stores a complete *process app* in the SBM Application Repository, making it available for deployment in *Application Administrator* or SBM Composer. See also *Export*.

Publish Problem Transition Type

Allows users to publish problems and resolutions pertaining to a primary item to the Knowledge Base.

Query

A search question that determines the data retrieved from the database. SBM provides two types of report query mechanisms: basic conditions and Advanced SQL conditions.

Quick Administrator

A feature that accelerates the setup process by preconfiguring each process app with certain runtime elements after the process app is deployed for the first time. These elements include a project for each workflow and sub-workflow in a process app and a default set of notification and notification rules.

Quick Deploy

An SBM Composer feature that lets a designer deploy a process app without being prompted to provide information that a regular deployment requires. Quick Deploy can also be used to deploy a process app without creating versions in Application Administrator, and without checking in design elements after the deployment finishes.

Quick Form

The non-custom form for a state or transition. Quick form is the name for the forms created automatically. With quick forms, fields are displayed in sections according to the permissions assigned to them. (Contrast with *custom forms*, in which permissions control only the accessibility and visibility of fields.) With quick forms, designers can use overrides to control the order in which fields appear within the sections.

Quick Links

Enables users to save and return to any page that is open in the content pane. For example, a quick link could open a specific report, the **Submit** form for a specific project, the three-pane view for a specific table, or a Web page.

Quick Transition

Quick transitions bypass the transition form in the SBM User Workspace, allowing users to transition an item with a single click. The form opens for users if there are required fields specified for the transition, however.

Record Lock

Record locks block access to an auxiliary or primary item while a user is updating or transitioning it. Administrators can determine the amount of time users are blocked from an item while it is being updated or transitioned. Locks can be manually removed by an administrator.

Reference

Lets designers make an association in one process app to a table, field, or other design element in an application contained in another process app.

Regular User Access

This access type grants full product access to users. Regular User access is controlled by privileges.

Relational Field Value Lookup

Provides an advanced searching mechanism that enables user to find values for Single Relational and Multi-Relational fields.

Relational Fields

Relational fields allow designers to establish relationships between primary and auxiliary tables. Single Relational, Multi-Relational, and Sub-Relational field types are available.

Remote Administration

Allows administrators to work in the SBM System Administrator via a remote connection.

Repository

See *SBM Application Repository*.

Required Attachments

Designers can require that users attach a note, URL, file, or item link to a primary item before transitioning it.

Restricted Attachments

Attachments that are restricted by users' Note and Attachment privileges.

Revision

A specific instance of a *process app* or *design element*. When designers modify an item and check it in, a new revision is stored in the *SBM Application Repository*.

Rich Editable Grid

Enables users to make a variety of changes to information in multiple items at the same time from a grid view. For example, users can view a list of items in the grid and provide a different value for the Severity field for each item as it applies.

Rich Graphical Reports

Offer dynamic charting and animation capabilities for Distribution, Trend, and Duration reports. Adobe Flash Player is required for users to run Rich Graphical Reports.

Role

A collection of application-related privileges. Roles provide one way to assign multiple SBM privileges to users and groups. Users and groups may have different roles in different projects, and a user or group can be assigned to multiple roles. Examples of the types of permissions associated with a role are the ability to read and update fields; the ability to perform specified actions on items, attachments, notes, and reports; and the ability to specify access to, or restriction from, certain transitions. Designers create roles in *SBM Composer* as part of a process app, which can comprise multiple applications. The roles span the applications within the process app.

SBM API (Application Programming Interface)

A C++ API (UNIX and Windows) that offers an object-oriented, high-level interface for manipulating SBM items, and a low-level interface for access to all tables and for handling of HTTP requests and browser connectivity. The SBM API is suitable for batch jobs, simple custom integrations, and custom imports and exports.

SBM AppScript

SBM AppScript is a subset of VBScript 4.0, with SBM extensions. Programmers can create and edit scripts that allow them to modify the system.

SBM Application Administrator

A Web-based component that is used to administer applications. Administrators with appropriate privileges can deploy and promote *process apps* to runtime environments. Application Administrator also hosts the SBM Application Repository.

SBM Application Engine

The component in Serena Business Manager that executes *applications*.

SBM Application Repository

An *Application Administrator* component that provides versioning capabilities, including *check in*, *check out*, and labeling for *design elements* and *process apps* created in *SBM Composer*. Version management capabilities allow application designers to collaborate on the creation of design elements. See also *Undo Check Out*, *Get Latest*, and *Publish*.

SBM Composer

A Windows client application that designers can use to create and edit *process apps*.

SBM Configurator

The component that you use to configure the settings and layout of your SBM installation. You use the SBM Configurator after installation to define your component distribution, enter database connection information, and manage advanced authentication and performance settings.

SBM Orchestration Engine

The component in Serena Business Manager that executes *system workflows* defined in *orchestrations*. Using *SBM Composer*, the designer can include Web services in an orchestration, which can then be executed in response to an *event* or by transitions in *applications*.

SBM Server

The collection of engines required for runtime execution of process apps (for example, the SBM Application Engine, the SBM Orchestration Engine, the event manager, and Single Sign-On).

SBM System Administrator

The Windows client application used to configure the SBM Application Engine. SBM System Administrator is used to manage projects, notifications, roles, users, and groups. Also , use this application to configure system settings and run database utilities, such as the Data Import Wizard.

SBM User Workspace

Refers to the end-user interface, which is accessed from a Web browser. The SBM User Workspace is used to enter and track primary items, report on these items, manage auxiliary table items, and perform limited administrative tasks.

SSO

See *Single Sign-On*.

Sample Database

An evaluation database that can be optionally installed by the SBM setup program. The Sample database can be used to explore the product.

Search Features

SBM offers several methods of searching for items in the SBM User Workspace: ID Search, Basic Search, Advanced Search, Global Search, Advanced Lookup Tool, and Field Value Searching.

Secondary Owner

Designers can set up a workflow so that one or more users are secondarily responsible for items while they reside in a particular state. A User, Multi-User, or Multi-Group field can be used to populate the secondary ownership property.

Self-Registration

Allows customers to automatically register themselves as external users. Automatic and Manual registration are available.

Serena Broker Service

Enables administrators to configure the SBM Mail Client and Notification Server. The Broker Service can also be used to stop and start these services.

Serena Business Manager

SBM is a set of components that allow users to easily create workflows, applications and process apps. Without requiring technical IT resources, SBM allows non-technical business users to build process apps that coordinate execution across applications, platforms, and teams.

Serena License Manager

The Serena License Manager (SLM) is a separate component that allows administrators to centralize license management across multiple Serena software tools. The License Manager helps administrators keep track of active licenses.

Single Sign-On (SSO)

Refers to a software component that enables a user to log in to a Web-based component of SBM and be recognized on subsequent accesses to that component or other Web-based components of SBM. This software also provides the ability for security tokens to be used in an orchestration, allowing Web services to be called without requiring the user to provide credentials at inconvenient times.

Snapshot

A version of a process app that Application Administrator creates that contains specific database information. It contains everything that a *blueprint file* contains, plus user data such as folder, projects, and users.

Solution

A bundle of process apps created by Serena Software, Inc.

SourceBridge

An extension that integrates source code control systems with SBM. SourceBridge can be used with version control applications and integrated development environments that comply with Microsoft Source Code Control Interface (MSSCCI) standards, such as Microsoft Visual SourceSafe. SourceBridge is used within the integrated development environment (IDE) or within the Serena Version Manager Windows or Web clients.

Standard Notification

Used to send e-mail messages to users when certain events or conditions occur. Standard notifications are also used to add items to and remove items from folders, call Web service functions, and execute scripts.

State Change History

Displays in the SBM User Workspace the history of the states a primary item has resided in during the workflow process.

State Fields

A set of fields associated with a particular state. State fields are derived from the default fields list and can be customized in SBM Composer so that users view appropriate information for a primary item while it resides in a particular state.

State Form

A form used only for viewing primary items while they reside in a particular state.

State

A key element of application workflows, states are positions that a primary item resides in while moving through the workflow process. States can also be considered a stopping point along a workflow's path.

Stopwatch

A function found within Date/Time fields that allows users to record elapsed time in the SBM User Workspace.

String Builder Tool

A feature in SBM Composer that lets designers insert references to table fields and form controls, and insert references to grid data in grid-style widgets. This creates dynamic forms and widgets that are driven by application and grid widget data and the actions of the user.

Sub-project

A project derived from an existing project. Sub-projects are also referred to as child projects.

Submit

The process used to add a new item to the system. Primary items are submitted into projects and tracked through their life cycle or workflow process. Auxiliary items, such as problems, are submitted into auxiliary tables.

Submit Form

A form page in the SBM User Workspace that is used to add new items to the system.

Subtask

A primary item that is associated with a principal task. Subtasks are created using the Create Subtask transition type or by users who link items together in the SBM User Workspace.

Subtask Action

Action that executes on subtask items that are created using Create Subtask transitions or that are manually linked by users in the SBM User Workspace.

Subtask Status

The value in a specified Binary/Trinary field that determines the status of subtasks residing in a specific state.

Sub-workflow

A workflow derived from an existing workflow. Subworkflows are also referred to as child workflows.

System Event Manager

See *Event Manager*.

System Favorite Folders

Four folders provided by SBM that are located in the Navigation pane of the SBM User Workspace and accessed by users. The Inbox, Submitted Items, Transitioned Items, and Updated Items folders are System favorite folders.

System Field

A fields provided by SBM that is required for the system's operation. System fields can be edited but cannot be deleted.

System Reports

Offer information about administrative aspects of the system, such as fields, user privileges, group membership, item locks, user activity, and license usage.

Target Server

A server in an environment that handles a specific type of process, such as an *event* or a BPEL process. Examples of target servers are the System Event Manager and the *BPEL server*. (The SBM Application Engine is also a target server, but it is listed separately, under the General tab of *Application Administrator*.) Process apps are deployed to target servers, which run the processes; *endpoints* are references to Web services or other resources that are used by the process app while it runs.

Template

(1) A type of process app that is a starting point for creating other process apps. For example, if a designer wants to include certain database fields in every process app, he or she could create a template that includes those fields.

(2) A PDF document that a designer specifies while configuring the PDF Widget. The document contains input fields that are filled in when a process app runs.

Termination Rule

Determines the condition that stops notification escalation from occurring. The system stops sending escalation notifications once a Termination Rule becomes true.

To State Property

Designates the state to which a transition is moving.

Transition Field

A field that is displayed to a user when a primary item is transitioned from one state to another. Transition fields can be customized so that users provide appropriate information for primary items as they move through the workflow.

Transition Form

A form page in the SBM User Workspace that contains fields for recording information about a primary item before it moves to another state in the workflow.

Transition Triggers

Used to automatically transition linked primary items. When one linked item is transitioned to another state, the item linked to it is transitioned as well.

Transition

A key element of application workflows, transitions activate a primary item's movement from state to state in the workflow process.

Undo Check Out

Discards any changes that designers have made to a process app or design element since it was *checked out*, leaving the item checked in and unchanged in the *SBM Application Repository*. The revision number that was created on check-out is released.

Unrestricted Attachment

Attachments that can be viewed by users who have privileges to view the item to which the attachment has been added.

Value Find

Represented by a **Find** button next to fields in the SBM User Workspace, Value Find capability for selection-type fields allows users to enter search criteria, such as an entire word, a few letters, or an asterisk, and then click the **Find** button to perform the search. Results appear in a drop-down list, allowing users to select a value for the field.

Version Control History

In the SBM User Workspace, displays the history version control actions associated with a primary item. File association information is also considered version control history.

Version History

In SBM Composer, an ordered list of the revisions to a process app or design element *checked in* to the *SBM Application Repository*. The version history shows such information as who checked in the item and when.

Web Architecture

SBM allows local and remote sites fully functional access to the system using a Web browser. This architecture ensures accuracy, cuts down on the duplication of effort in tracking systems, eliminates system downtime, and allows for future upgrades of the tracking system to proceed smoothly.

Web Server

The application server used to operate SBM.

Web Services

Applications that are accessible using standard Internet protocols and formats such as Extensible Markup Language (XML), Hypertext Transfer Protocol (HTTP), and Simple Object Access Protocol (SOAP). Developers can implement applications that interact with Web services on any platform in any programming language, as long as the language can create and respond to messages that are sent using SOAP over HTTP.

Web Services API

The SBM Web Services API enables developers to create integrations with SBM that create, read, update and delete items in SBM.

When Rule

Determines the condition or conditions that activate a notification.

Widget

Small applications that are part of the SBM User Workspace and provide a special function. The form palette contains several specialized widget types.

Workflow

A collection of states, transitions, and fields that define an organization's tracking process.

Workflow Hierarchy Structure

A graphical representation of parent workflows and child workflows or subworkflows.

XML E-mail Submission

The e-mail submission feature includes XML parsing of the body of the e-mail message. Fields in the submitted item are populated with data contained in the message body.