

# **Managing Design Changes in Enterprise SBM Installations**

## Summary

This document explains how to organize your SBM maintenance and development process to ease long-term maintenance of your process apps as your business needs change.

## SBM – Designed for Adaptation and Integration

SBM is a process-based work management product that enables businesses to guide, track and monitor the work they do. While it can be used on a purely departmental basis, our most successful customers have discovered that it can be applied throughout their organization to provide structure and visibility to a wide variety of business processes. For example, IT organizations often use SBM to manage their help desk. Users throughout the company submit tickets using Serena Request Center, and then SBM processes guide the ticket through fulfillment processes in the appropriate department. New employee onboarding, benefits and time off requests are routed to HR, IT gets email and infrastructure issues and Expense reports are routed to Finance. Reporting and auditing capabilities built into SBM enable folks whose job it is to oversee these processes to both understand broader trends and drill down to any level to understand the details of any transaction or ticket.

To meet the needs of the dynamic organizations of our customers, we kept a number of basic premises in mind when designing SBM.

### Change is Fundamental

Change is fundamental in business. Market conditions, business goals and business requirements are constantly changing and successful businesses continuously adapt to these changes. Automation of these business processes must both be able to keep up with these changes and continuously improve as optimizations are discovered.

SBM is designed for rapid adaptation to emerging and changing user requirements. Process changes are achieved using familiar drag-and-drop interfaces designed to be simple and accessible to non-developers familiar with the business processes being managed. Our goal is to permit people involved with the day-to-day processes being managed to adjust and adapt those processes as new requirements emerge and inefficiencies are discovered in the way existing requirements are met. SBM makes process changes simple — permitting processes to adapt to changing and discovered business requirements instead of forcing business to adapt to a rigid process.

### Systems Also Participate in Processes

The IT landscape has many diverse tools and systems. Process automation software must work with and not merely alongside of existing systems. Just as people participate in processes, so do other systems of record. A process designer can identify the touch points between systems and build that into the process design at the highest level. SBM is built to enable such integrations. For example, you can use SBM's REST Web service support, SOAP Web service support, Orchestration Workflows, AppScripts, or even our C++ API to bridge the gap between SBM and other systems that comprise the IT environment.

## SBM Process App Development

Process app development is best done iteratively, with a business owner who understands the process being automated closely involved. The business owner knows the people who participate in the process, the roles they play, the data that must be gathered and the reports needed by those who oversee the process. The business owner also understands the goals of the effort and can guide development to ensure the productivity, revenue, quality, risk reduction and visibility goals are met.

If you purchased a solution based on SBM from Serena, your primary goal may be to adapt existing process apps to the unique circumstances of your business environment. This often involves only simple changes to the data model (i.e. changing the information collected during the process) and modifications to the workflow map to reflect differences in the states and flow of the process.

### Developing New Process Apps

Unlike purely custom implementations, where requirements must be fully defined up front, SBM enables you to start with a simple process that guides participants through their work, then, over time, iteratively improve it.

During development, the process designer visually defines:

1. Data that needs to be collected, presented and manipulated by the process.
2. Process workflows that reflect the movement of work in the organization.
3. Roles and responsibilities of the process participants.
4. User interfaces that will be presented to the process participants.
5. Reports that provide visibility to people responsible for monitoring process health.

When the process app has reached a sufficient level of maturity, as validated by the business owner, it can be rolled out to end users.

### **Refining Existing Process Apps after Rollout**

With SBM, the initial rollout isn't the end game, it's the starting point for continuous improvement. Serena recommends that you identify a process owner who gathers feedback from process participants and ensures that the SBM process evolves to address that feedback.

Some customers use a forum that meets periodically with representatives of the process participants to ensure that this feedback is gathered and acted upon. During this phase, the feedback can be used to improve custom forms, automate UI behaviors using form actions, access external data using the REST grid and provide embedded reports for accessing related data. As additional reporting needs emerge, you provide richer reports for consumers of that data. These kinds of changes can be done by people without in-depth technical or programming skills. Familiarity with SBM Composer, together with knowledge of the process being modeled are the only prerequisites.

For example, through user feedback, you discover that users are forced to scroll to the bottom of a transition form to access a required field. So while the SBM implementation may achieve the functional requirements of managing the process, an inefficiency has been identified that can and should be addressed immediately. You can fix this oversight in minutes by dragging the field to the top of the form using SBM Composer.

## **Responsibilities and Staffing to Leverage SBM**

To get the most value from SBM, an installation needs to have the right staff for the right purposes. For rapid iterative process improvements, the most important characteristic is familiarity with the business process being automated. Long change cycles undermine the ability for the process to change along with the business. On the other hand, for integrations and more technical aspects of creating and maintaining SBM systems, the requirements should be defined deliberately before development begins, and then implementation can occur in a longer development cycle without close proximity to the business. This section discusses various responsibilities involved with implementing and managing an SBM system.<sup>1</sup>

### **Installation and System Configuration**

System administrators typically perform installation, upgrade and system configuration tasks associated with SBM. These responsibilities often fall to IT staff members who are responsible for maintenance of the physical and software infrastructure of the company.

### **SBM Administration**

In contrast to system administrators, the SBM administrator manages the day-to-day operation of SBM processes. An SBM administrator manages SBM users and groups, privileges, notifications, projects and administrative workflow overrides using SBM Application Administrator. While users can define reports for their own needs, administrators typically define a variety of reports that meet the needs of process participants and observers. An administrator needs to be familiar with the business processes as implemented in SBM and the needs of the SBM user base.

### **SBM Process Design Maintenance**

Closely related to SBM administration are the additional responsibilities for making incremental changes to the design of your processes using SBM Composer. By participating in a solution forum, an SBM administrator/designer can work with users and the process owner, understand the changing requirements and quickly implement the changes that bring the greatest value to the process participants. By having someone with proximity to the process available to make iterative design changes in short time horizons, you can leverage the 'built for change' capabilities of SBM.

---

<sup>1</sup>Depending on its size and complexity, an SBM system could have several people in some roles, or conversely have one person performing taking on more than one role or performing an SBM role along with non-SBM responsibilities.

The following sections discuss types of design work done in SBM Composer and the skill levels needed for each.

### Workflow Design

The workflow defines the path an item takes through an organization as it participates in a process. At the highest level it consists of a set of resting points (states) and transitions that move the item between states. Active states have owners who have current responsibility for the item. Users can transition items between states to pass responsibility to another user. Decision nodes can be defined that use business rules to determine how an item is routed based on information that has been collected. Various properties on the workflow control access to transitions and visibility of data based on the role of the user viewing the item. A process designer may modify a workflow by adding or removing states or transitions, changing the business rules that govern the behavior of decisions, or changing the visibility of data at the workflow, state or transition level. These types of changes are typical as processes evolve over time and can be made with a basic understanding of the business processes and SBM Composer.

### Data Design

Each application in SBM has a *primary table* that defines the information collected in an item over its lifetime in the process. As new requirements emerge, a process designer will typically modify the data model by adding fields to the primary table to collect additional information. For example, if a process designer needed to add an *Urgency* field to the data model, they would drag a Selection field from the Table palette into the primary table, name it "Urgency" and add selection values for "Low", "Medium", "High" and "Critical" to the properties of the field. The new field can be easily added to any custom forms by specifying a related field and a relative position using the *Add to Forms...* feature on the Forms tab of the field. As with workflow changes, data design changes are relatively simple and can be made without special technical or programming skills.

### Visual Design

SBM automatically generates so-called 'quick forms' for states and transitions unless you define a custom form for them. If you're using quick forms, you have limited control the layout of fields on the form by using properties like **Span entire row on forms** and **Privilege section** on the field. You can control the sequence in which the fields appear by changing the field privilege order in workflow, state or transition properties and the order of transition buttons by changing the transitions properties on states.

But while quick forms may be useful for prototyping a system or for infrequently used states or transitions, custom forms provide much greater flexibility over the presentation of data, provide the ability to define active behavior on a form and permit simple integrations using embedded reports, HTML frames and REST web services. With custom forms, you can optimize both view and data entry forms to better serve your end-users.

Designing custom forms is easy. The SBM Composer visual form editor lets you add and move fields using familiar drag and drop techniques. You can start with a custom form based on the quick form or design the form from scratch, adding fields, containers, action buttons, images, links, reports, REST Grids and more. Using custom forms, you have complete control over transition button placement on the form as well as all toolbar and header actions and display.

The next level of visual design involves the addition of behaviors to forms. For example, if you'd like the details tab to be automatically selected based on severity, you can add a form action that reads:

```
When
  this form is loaded
Or Severity field changes value
If
  Severity field in 'Urgent'
Then
  activate Details tab
Else
  activate Standard tab
```

The form actions available to create dynamic form action are numerous and varied, covering most events, conditions and actions you would want to do on a form. For conditions that we haven't provided, you can reference business rules and, as a fallback, you can write a JavaScript snippet to implement more complex conditions. Similarly, if the action you want to perform is not available, you can write a small JavaScript snippet to perform an action when an event occurs and the corresponding condition is met.

Almost all custom forms work can be made by the process designer without any special skills or programming knowledge. You may choose to employ a visual designer to create a better visual experience for your users, but this is rarely necessary.

### **SBM Process Design Creation**

The creation of Process Apps is similar to the maintenance work described above, except that the process designer works with the process owner to envision and design the overall process app rather than thinking in terms of incremental changes to an existing process. After working with SBM in a departmental system, our most successful customers have been emboldened to create new process apps for a wide variety of processes in their organization. They discover a manual process that would benefit from automation, control, audit and reporting. Working with the process owner, they define the process steps and transitions, the data to be collected and the roles of the participants, then use SBM to rapidly create process apps to solve the business problem. Most of the work is in the application editors in SBM Composer which provides familiar metaphors and easy to use interfaces for laying out the process. A process designer can design the data model in the table editor, lay out the workflow using the workflow editor and assign roles to automate their process and have the process up and running in a development environment in short order. They can then iteratively improve the design until it becomes a minimal viable product eligible for roll out to end-users.

The person responsible for process design creation and maintenance might hold a title of Business Analyst and should have familiarity with SBM Composer. A process designer does not need special software engineering or developer training, but should have skills commensurate with using business productivity tools like Microsoft Excel and PowerPoint.

### **Designing and Implementing Integrations**

In contrast to process creation and refinement, integrations with external systems often are time consuming efforts requiring detailed planning, complex implementation and thorough testing. The complexity depends on the type of integration and the third-party products involved. In the simplest case, the integration may be accomplished with a relatively simple REST Grid on a custom form, which requires only knowledge of a third part REST Web service interface and an understanding of how to build custom forms in SBM Composer. If the external system supports SOAP Web services at a business logic granularity, the more complex task of developing SBM Orchestrations may be needed to implement the integration. And in the most complex circumstances, programming may be required to access the data from the external system.

While Serena has tried to make it as simple as possible to create integrations with external systems, this is often the most complex task facing SBM customers. If the system being integrated with has rich business level SOAP and REST Web service interfaces, these integrations can be accomplished without programming using SBM Orchestrations and REST integrations on forms. But even then, the complexity of the work may exceed what a business analyst is comfortable doing. While the skillsets required of an integration developer depend on the tasks at hand, they tend to be closer to those found in a software engineer.

Because integrations are longer term larger efforts involving more detailed planning, their development need not be as closely tied to the ongoing business as the incremental process changes described above. Capabilities of SBM such as the application independent orchestrations can be leveraged to isolate these development efforts and make them deliverable and consumable by the process developer when they are completed.

## **Coordinating Iterative and Long Term Development Efforts**

As described above, there are two very distinct types of work required for development with SBM. Most of the work involves administration and visual design and can be accomplished without deep technical or programming knowledge. An understanding of and proximity to the business process as well as familiarity with SBM's Composer and Application Administrator are all you need. On the other hand, work with orchestrations and integrations usually requires the application of a different level of technical skill. These more complex efforts tend to be longer term efforts and more deliberate than iterative maintenance. How then, do you coordinate this fast moving, iterative and adaptive change to your SBM processes with longer term work? This section describes a number of approaches and corresponding features in SBM designed to answer this question.

### **Iterative Changes**

Generally speaking, iterative changes can continue without detailed coordination with longer terms efforts, so long as that work is well defined and has been encapsulated appropriately as described below. For example, if orchestrations

being developed rely on aspects of an existing application, like the presence of certain fields, states and transitions, these should be left unchanged by iterative changes being made. But while keeping interfaces used by long term efforts stable may simplify the integration of that work when it is completed, keeping the SBM processes fluid and dynamically evolving with the needs of the user and business is primary. If properly encapsulated, bringing those longer term efforts into the process can be done quickly and easily.

The typical process for iterative changes is described in detail the [SBM Path to Production for Enterprises](#) white paper.

### **Long Term Development Efforts**

When undertaking a long-term development effort in SBM, the central goal is to structure your efforts so that they interface with the related process apps using simple, well-defined interfaces. They should encapsulate rather than expose complexity. These development efforts can occur both within and outside of SBM and are typically beyond the skills or comfort level of a typical process designer. The developers involved with these efforts will need a development instance of an SBM system independent of the primary SBM system used for iterative development, staging and production.

#### **Development outside of SBM**

An example of this type of development is creation of a REST service that exposes data from third-party line-of-business software in a way that can be displayed on an SBM custom form REST Grid. Depending on circumstances, this could involve writing a code module that interfaces with an API on the external system and hosting of the service. Similarly, it could involve the creation and hosting of SOAP web services that expose the functionality of an external system at a business granularity, suitable for use in an SBM Orchestration or as a Web service action on a transition. A development SBM system is needed to ensure that the interfaces they are developing are compatible with SBM requirements. In the examples above, this would mean using SBM to test the REST and SOAP web services they are developing. By its nature, this type of development work is encapsulated because it presents a high-level, well-defined interface to SBM.

#### **Development within SBM**

This category of development involves the use of various more technical integration technologies that SBM provides. An example is development of orchestrations with appropriate fault handling and compensation. Development of complex scripts or software that calls the SBM C API would also fall into this category. These efforts often occur in conjunction with development occurring outside of SBM. For example, an external system may expose only a very low level API. This API needs to be wrapped with business level Web services, which can then be consumed by SBM orchestrations.

#### **Techniques for Encapsulating SBM Development**

Actions on transitions in the application workflow provide most of the interface points for interactions with encapsulated development efforts. For example, synchronous workflows, asynchronous workflows, AppScripts and Web services are all initiated from the Actions on a transition. The other major integration point is 'on-the-glass' just-in-time integrations on the forms end users see. The sections below describe various types of integrations and discuss development practices that create the appropriate isolation and encapsulation to facilitate subsequent integration with the primary SBM system.

##### *General Considerations – Internal Names*

Many of the integration techniques discussed below use SBM Web services, AppScript or the C API to interact with workflows in SBM applications. Since field names can change, all web services that interact with the fields should use the field database name, which cannot be changed after initial deployment. Any references to workflows, states and transitions should use their fully qualified internal names, since the name properties of those items may be ambiguous, or may change as the application is modified to adapt to changing needs.

##### *Orchestrations*

There are four types of orchestrations to consider in this section. If you are developing an orchestration in a long term development effort, when possible, you should develop the orchestration in a separate process app to facilitate later integration. The various types of orchestrations are discussed below.

##### *Synchronous orchestration workflows*

These orchestration workflows are called from an SBM Application by a transition action. Item data is mapped to orchestration workflow inputs and updated by the orchestration workflow outputs. A synchronous orchestration must

exist in the same process app as the application which calls it. Because of this, after such an orchestration has been developed, it must be merged into the process app on the primary SBM system using the Compare/Merge feature of SBM Composer. The specification of a synchronous orchestration workflow should define the inputs and outputs of the synchronous orchestration and these fields should remain unchanged during the development effort.

#### Application Dependent Asynchronous Orchestration Workflows

This is the most common orchestration initiated from an SBM application. They are executed by raising an event using a transition action. The event data corresponds to the data present in the primary table of that application. These orchestrations may call web services in both external systems and the SBM application. As always, you should take care to keep the interfaces between the application and this orchestration unchanged during development of the orchestration. You should ensure that the *Event without Reply* orchestration link in the application has the "Lock definition" checkbox enabled to prevent the interface with the orchestration from changing when the fields in the primary table are renamed. This mechanism is available in SBM 2009 R4.02 and beyond.

These orchestrations do not need to be part of the same process app as the calling application. To create a compatible orchestration in a separate process app, export the Orchestration Link named *Event without Reply* from the initiating application, then, in a separate empty process app, add an orchestration. Under the Application Links item in the application explorer, right click and choose the *Add New Event Definition...* menu item. In the dialog that appears, choose the *Create from event definition* file option, navigate to the event definition you exported and you can proceed with development. As always, you'll want to have an SBM system that raises the event to test with.

#### Application Independent Asynchronous Orchestration Workflows

These are asynchronous orchestration workflows that wrap external systems and can be reused between multiple SBM applications. The event interface for these workflows is application independent and is defined by the orchestration developer to provide sufficient instructions to the external system. These workflows can be thought of as defining a common reusable interface to an external system with each workflow representing a command. They are called from an SBM application using an action on a transition. They differ from application dependent asynchronous orchestration workflows because the application's item data can be mapped to the event interface, while the application dependent event interface is automatically defined by the application's item data.

To create an application independent asynchronous orchestration workflow, create a new process app and add an orchestration by right clicking on the process app in the application explorer and selecting the *Add new > Orchestration* menu option. Under the Application Links item in the application explorer, right click and choose the *Add New Event Definition...* menu item. In the dialog that appears, choose the *Create new custom event definition* option, then define the Object types and Event types. These values are used by the event manager to determine what orchestrations to run when an event is raised. Now define the custom data that will be passed in the event. These values will be the inputs to orchestration workflows run using this event.

To raise an event defined by this application link, first export it by clicking the *Export event definition...* button and saving it to an .mtd file. Now, in the initiating application, right click on the **External Events** item under **Orchestration Links** in the application explorer and select the *Add New External Event...* menu item, navigate to the event file and import it into the application. Finally, to use this, create an action on a transition, select orchestration workflow, and, in step 2 of the first wizard page, click on [the local event](#) and change it to [an external event](#). Finish the wizard, selecting the external event you just imported, then click on data mapping to map the item data in the application to the inputs in the event. This mechanism is available only in SBM 10.1 and beyond.

#### Externally Initiated Asynchronous Orchestration Workflows

These are asynchronous orchestration workflows initiated by an event raised from an external system. The event interface for these orchestration workflows is defined according to the requirements of the external system. These orchestration workflows typically call SBM web services for manipulating items in a specific application.

To create an orchestration that can be initiated from an external system, follow the same steps described above for creating an orchestration and defining a new custom event definition. Instead of exporting the event definition, click the *Export external event WSDL* button to create a WSDL file that specifies a web service suitable for raising the event. More details and explanation is available in the documentation.

#### Other Advanced Development Techniques

Other types of advanced development include AppScript development and C API development. While there are many SBM installations that leverage these interfaces, if you can accomplish what you need using Form Actions, Web services and Orchestrations, you will end up with a more maintainable system.

The main concern in all of these types of development, when done as a development effort separate from the primary SBM system is that the interface points are well understood and stable. The Internal Name feature introduced in SBM 10.1 has made this substantially easier than before. Previously, any references to the SBM workflow model, including workflows, states and transitions, was done by name. Since state and transition names are visible on forms, ordinary maintenance and evolution of workflows could change the names and break integrations. Now, references to these items using AppScript, the Web service APIs or the CAPI can use the internal name, which remains unchanged after the initial publication of the process app.

### **Merging Long-term Changes into Development, Staging and Production Systems**

Once development of a long term project has completed, you will need to integrate that effort back into the primary SBM system.

#### **Orchestrations**

For orchestrations developed in separate process apps, this process is fairly simple. If the orchestration is initiated by an external system, you just deploy the process app containing the orchestration and it will execute when the external system fires the event. For other asynchronous orchestration types, you just need to modify the calling application to raise the event that calls the orchestration. If the orchestration is defined using an external event, you will need to map the data from the calling application into the inputs of the orchestration. If it was defined based on the applications data model (application dependent orchestrations), just raise the event in an action on a transition.

For synchronous orchestrations or orchestrations you have kept in the same process app as the related application, you will need to use the Composer compare and merge tool to copy the orchestration and its Orchestration Link from the development process app blueprint into the process app in the primary SBM system, then call the orchestration from the transitions where you would like the work to be done.

#### **Merging AppScripts**

AppScripts can be copied between process apps as files. So long as the interface between the app script and the application it is interacting with has remained stable (for example, through the use of internal names), it will continue to work after it is copied to the primary system.

#### **Merging Other Changes**

Finally, any development you have done in a separate system can be merged into a process app in the primary SBM system using Composer's compare and merge feature. Using that feature, you can copy any changes you have made a divergent version of a process app back into the primary system process app. You can use this for virtually all types of design elements, including orchestrations, forms, workflows, AppScripts, rules, roles, fields and auxiliary tables. If you have changes within a design element, the comparison will give you a report that outlines the difference and will highlight the differences visually in the familiar SBM Composer environment.

While this technique will permit you to bring any separate development efforts into your primary system, you should work to encapsulate any long terms projects in separate design elements, so they can be moved as a unit into the primary system without the difficulty of sorting through diverging properties of existing objects.

## **Take Home Messages**

Get value from SBM by enabling processes to adapt to changing business rather than forcing business to adapt to a rigid process.

Most changes needed to adapt and evolve processes over time are easy and intuitive with SBM. SBM is designed to permit iterative process changes by business savvy non-technical personnel. Keep those people close to the process.

Task a process owner to be responsible for periodically convening a forum consisting of representatives of process participants - submitters, fulfillers and process observers, along with the process owner. This forum should meet to discuss how the people use the solution and learn what can be improved. Iteratively improve the process based on the feedback that team provides.

Some efforts, such as integrations, are best done by developers as a longer term project. Encapsulate these efforts in ways that make independent development feasible and incorporation into the primary SBM system painless.

Do not bundle small projects that can bring an immediate benefit to your organization with larger development efforts. Doing so diminishes SBM's value proposition of being built for change.