

# **SBM Path to Production for Enterprises**

## Introduction to the SBM Development Process

SBM is designed to simplify business process deployment and development. Workflows, forms and other process artifacts can be visually designed in SBM Composer using familiar drag-and-drop interfaces. When a process app is at a point where you're ready to try it out, you can validate it and deploy it to a test environment with the click of a button. You can use the environment's Application Administrator to set up your users, groups and projects in a staging environment. When you're ready to go, Application Repository makes it easy to promote configured processes from staging to your production environment. This whitepaper discusses the standard path to production and an alternative *sandbox* approach.

## The Path to Production Participants

*Path to Production* describes the steps required to deploy a design into a production environment to make it available to your end users. In SBM, it involves the interplay between SBM Composer, SBM Application Repository and various SBM runtime environments. These components are described below:

### SBM Composer

*SBM Composer* is the client application used to design process apps. In Composer, you define the data you will be collecting to represent an item, how that item will move and be updated within your organization, roles within your organization and the forms that users will use for viewing and modifying the item data.

### SBM Application Repository

*SBM Application Repository* serves two purposes. It acts as a source control repository for individual components of the design (like forms or workflows). More importantly for the purposes of this whitepaper, it is the application responsible for deploying process apps to environments and moving process apps between environments.<sup>1</sup> The Application Repository stores each unique version of a process app published to it and maintains a record of all significant interactions with runtime environments. It is the central player in the SBM Path to Production.

### Environment

The other participant in the Path to Production is the *environment*. It refers to a complete set of runtime services for the administrator and end user. These include the Application Engine, which presents the forms designed in Composer to end-users, the Orchestration Engine, which executes automated BPEL processes that call Web services, the Notification Server which dispatches notifications to interested users and the Mail Client, which processes incoming emails as well as other runtime services.

SBM installations typically have multiple environments. A *development environment* is used to test changes without affecting your end users. A *staging environment* is used to duplicate the configurations you'll be using in production and to test a configured process app. A *production environment* is where the process apps and solutions you create provide value to the business users of SBM.

Each environment also contains an *SBM Application Administrator*, which permits administrators to configure and refine the runtime aspects of the process app. Generally speaking, you use Composer to create those parts of a running system that stay the same between environments (the design) while you use Application Administrator to configure aspects of a system that can change from one environment to another. For example, users, groups, projects and notifications are all administrative concepts that are configured in Application Administrator. It is also used to override designs to allow the behavior of the process app to change from one project to another.

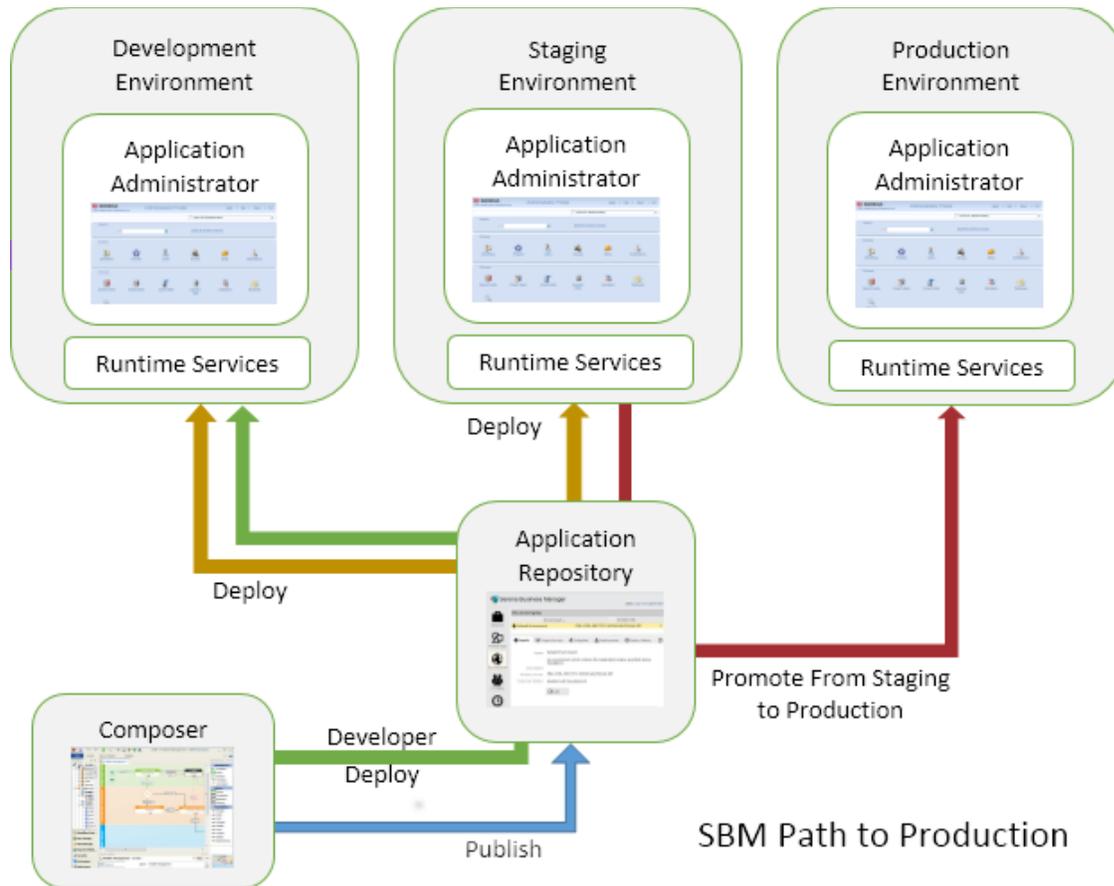
## Standard Installation Topology

SBM can be installed with various topologies to serve the varying needs of our customers. A standard installation has three environments, one each for development, staging and production, and will use a single Application Repository for mediating the deployment of process apps to environments and the promotion of process apps between environments.

---

<sup>1</sup> Although a developer can deploy a process app to an environment directly from SBM Composer, internally, this first publishes the process app to Application Repository, then initiates a deployment to the target environment.

This diagram illustrates the relationships between Composer, the Application Repository and the runtime environments in this type of installation.



### Path to Production Lifecycle

The following steps describe a typical path to production lifecycle with a standard installation.

A business developer begins by creating or modifying a process app using Composer. During development, he or she refines the process app, deploys it to the development environment and tests, repeating until the design is ready for staging. To facilitate development, the administrator may choose to turn off versioning for these development deployments.<sup>2</sup>

When the design is done, the developer publishes the process app to the Application Repository where an administrator will deploy it into a staging environment.

In the staging environment, the process app is configured using Application Administrator and the reporting interface to provide the runtime and administrative features required for production. For example, users are defined and granted appropriate privileges and roles. Projects may be defined and design overrides for the projects put in place. Once the process app is fully configured, acceptance testing occurs in this environment to ensure it is ready for users to begin using it in the production environment.

<sup>2</sup> The environment in Application Repository has a Composer property that can be set to "Enable Deployment" to permit direct deployment from Composer. If that property is set to "Enable Development Deployment", the user in Composer can choose to deploy without checking in changes or creating versions of the deployed blueprint in Application Repository.

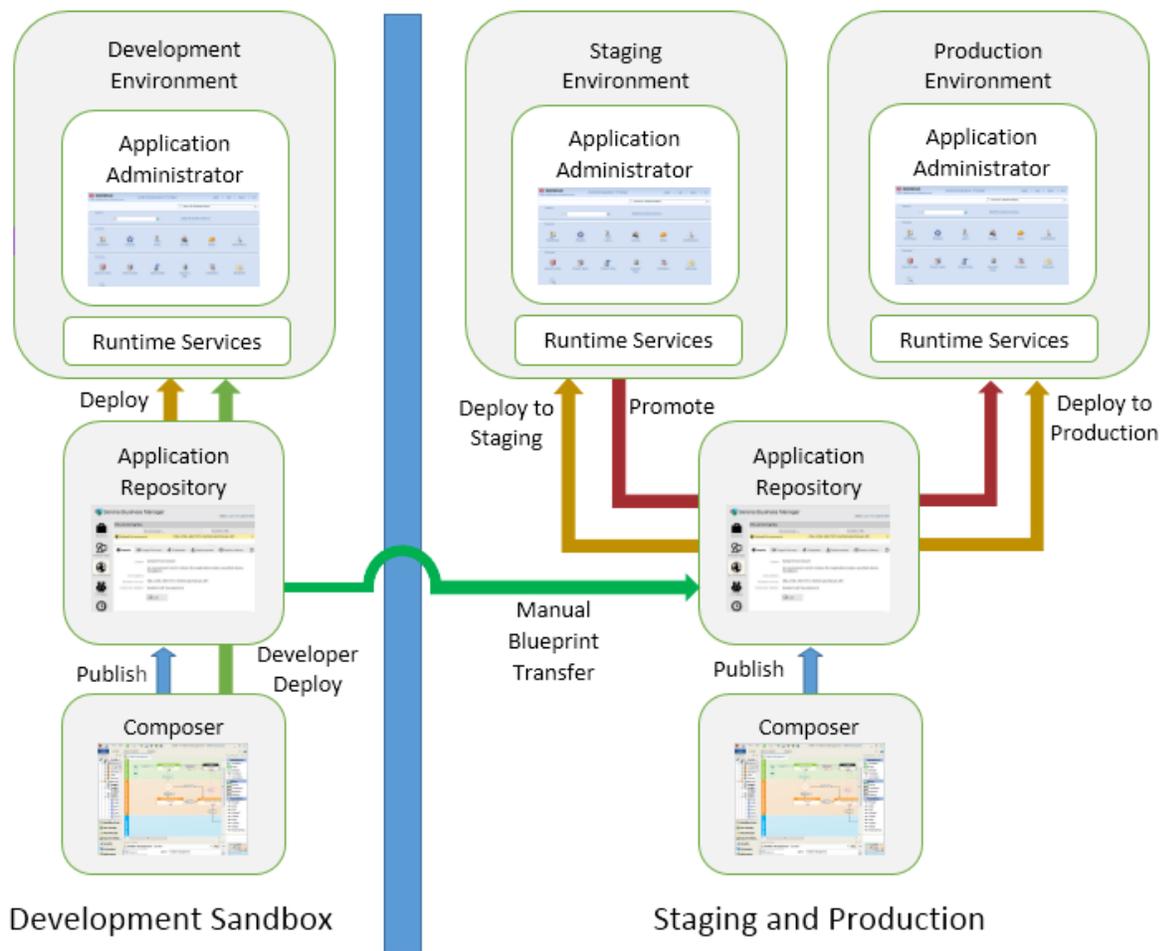
Finally, when the process app is accepted for production, it is promoted from the staging environment into the production environment using the Application Repository's *Promote* feature. Promotion differs from deployment, because while design transfers only the design created in SBM Composer, promotion transfers the runtime and administrative settings to the target environment in addition to the underlying design.

## Using a Sandbox Development System

Because development procedures vary from company to company, the SBM configurations you use need to adapt accordingly. In this section, we describe a path to production appropriate for a setting in which physical separation of a sandbox development system from the staging and production system is a requirement. Advantages of this type of configuration include:

- Simplified management of user and privilege models between development and production
- Physical separation permits physical security around the production environment
- Support for disconnected development efforts

The following diagram illustrates the relationships between Composer, Application Repository and the runtime environments in a sandbox system:



When you have a separate development sandbox system, users of that system have no access to either the staging or production environments. A developer can be granted full administrative privileges to do whatever he or she needs to do in the Application Repository and the development environment without risk of unwanted changes to the production system. In fact, the developer need not even be a user in the staging/production system.

With this topology, the process app developer works entirely in the development sandbox. When the design is done and ready for staging, the process app is exported from either Composer or the Application Repository as a blueprint

(.msd) file and then imported into the Application Repository on the staging and production system. This is the manual equivalent of publishing the process app from the sandbox system to the Application Repository in staging and production system.

You can also make administrative changes to the process app in the sandbox and transfer these to the staging system using the manual promotion capability of the Application Repository. Obtain a snapshot from the development environment after you've made the administrative changes there, export it from the sandbox Application Repository as a snapshot (.mss) file, import it to the Application Repository in the staging and production system and then complete the promotion from there.

An administrator then deploys the process app to the staging environment and staging proceeds as before using Application Administrator and the reporting interface.

Finally when the process app has been configured and is ready for production, the administrator uses Application Repository to promote the configured process app to the production system.

## Patching Process Apps in Production

In the processes outlined above, the process app design is never modified in the staging and production system. This means that the "source of truth" for the design is always the development system and the Application Repository to which it is connected. While this makes the development process simple, in practice it is not always feasible. In considering alternative approaches consider the following questions.

- Are you modifying the version of the process app that is currently in production?
- Are you applying only the changes that you intend to the production environment?
- Have you applied the same changes to your ongoing development version of the process app?
- For ID'ed items that you are adding to production, have you ensured that the identical item (with the same ID) has been added to the development version of the process app?

This section discusses the challenges in making changes directly to the staging or production system and describes approaches for addressing those challenges.

### Requirements for Implementing a Patch in Production

To make a change directly to a production process app, you need to obtain access to the version of the process app that was deployed to production, make changes to that design and redeploy to production. If you were in the middle of an incomplete development effort, you need to make sure that none of those changes accidentally get to production – only the patch change you want. You also need to make sure that the change you made to production is incorporated back into your ongoing development so it is not lost when you subsequently deploy changes currently under development.

A potential issue with patching a system arises when you make a change to production that introduces an artifact like a state, which is identified in the system with a unique identifier rather than a name. If you independently add a state with that same name in your development system it will not have the same identity as the one you previously added to production. This means that when you subsequently deploy that process app to production, there will be two states with the same name, one of which is no longer participating in the workflow. The techniques for avoiding this problem differ between a standard installation and an installation with a sandbox development system.

### Approach for Patches in Standard Installations

When you're using a single Application Repository (i.e. you're not using a sandbox system), you can make changes to a design that's been deployed to production using Composer's *Patch Context* feature. When you open a process app in Composer in a patch context, you're working with a labeled version of the process app. You can check design elements out, modify them, check them back in, then publish and deploy the modified version of the process app. You can do this with minimal disruption of ongoing new development of the process app.

There are some limitations on what you can change in a patch context to prevent subsequent ID collisions. For example, you can't directly add a new state or field; however, you can copy a state or field from the development tip. This prevents divergence between the production system and the development tip for these items. So if you want to add a field to a patch production, you need to first add that field in your development system, then check the table out in the patch context and use the table's **Add Existing Field...** context menu item to copy it into the patch context. Once you have updated the patch context with the needed changes, you can publish the patched version of the process app and deploy it to staging or production.

## Opening a Patch Context

Bring up the **Open Process App** dialog by choosing **Open...** on the **File** menu. Select the "Look in: Repository" radio button and select the process app you are patching. Click the "Open labeled version..." button. A dialog appears with a list of published versions. Choose the version that is in production and click OK. If you have not previously opened this version as a patch context, a dialog appears offering to create a patch context based on the selected version label. Click **Yes** to create the patch context. When you're using Composer in a patch context, "[Patch]" appears in the title bar after the process app name and the version label.

## Approach for Patching in Development Sandbox Installations

When you are patching a production environment where the development system is isolated, the same concerns outlined above apply. However, because the systems are isolated from each other, SBM cannot prevent you from creating ID conflicts, so you need to take steps to ensure all changes to production will be consistent with subsequent deployment of work under development. There are several ways you can approach the problem.

### Using a Patch Context

If you have the version of the process app that is currently deployed to production in the sandbox Application Repository, you can create a patch context in the sandbox Composer, based on that label, make changes as described above and then export the design as a blueprint. At this point, the blueprint can be imported into Application Repository in the production system and deployed to the staging or production environment. As before, you'll have to update the development tip with any changes made directly in the patch context.

### Using the Composer Compare Merge Feature

There are two ways to use Composer's merge feature to ensure changes made in one system are accurately reflected in the other. You can copy changes from the sandbox system into the production system, or vice versa. The approach you take depends on whether the change you want to make in production has already been made in the sandbox system.

#### *Merging Changes from Sandbox to Production*

If the production change has already been made on the sandbox system, you'll want to merge that change into the production system. To do this, first obtain a blueprint (.msd) file of your process app from the sandbox system by using the **File->Export...** menu item. Then open a Composer instance that is connected to the production Application Repository and open the version of the process app that is in production. If you originally deployed it by importing it into production Composer, published then deployed it, it will be in the Open Process App dialog (Repository option) with a blue and gold icon. If you originally deployed it by importing it into Application Repository and deploying it from there, it will appear in the dialog with a red icon.

Once you have the production version of the process app open in Composer, you can use the **File->Compare->With Local File...** menu command to compare the blueprint containing the sandbox version of the process app with the production version. You can then use the **Copy to Open Process App** context menu item in the compared app (the blueprint from the sandbox system) for any items you want to duplicate in the production environment. This method preserves the unique IDs that represent the design elements so you won't need to worry about conflicts later on.

#### *Merging Changes from Production to Sandbox*

If you'd like to make the changes first in the production system, you can do so and then merge these changes back into the sandbox system by using the technique outlined above in reverse. In this case, you'd obtain a blueprint file for the process app from the production system by exporting from Composer, open the sandbox version of the process app in Composer, then merge into the development system.

### Making Minor Changes in Parallel

For changes that do not create an artifact with a unique ID at runtime, you can make the changes in parallel between the systems. For example, if you are adding a transition between two states, you can add it on both the production system and, independently, on the sandbox system. However, you must be sure not to add any of the following design elements independently to the systems:

- Tables
- Fields
- Selections
- Application Workflows

- States
- Report Definitions
- Roles
- Applications
- Orchestrations

Doing so will cause a conflict when the version of the process app in the sandbox system is ultimately deployed to production. You may choose to use the Comparison feature described above to make sure these minor changes were accurately made in both systems.

## Conclusion

When using a standard installation with one Application Repository instance shared between all environments, SBM provides a straightforward and controlled path to production capability. If you are using a sandboxed system for development, the possibility of divergence between the designs is increased. In this case, you need to be aware of the issues that may arise and techniques for merging changes described in this document.<sup>3</sup>

---

<sup>3</sup> See also [SBM Development Best Practices](#) for information about minimizing the difficulty of merging changes.