

Compass > Discuss > Community Wiki > IdentityNow Articles  
> SailPoint SaaS Change Management and Deployment Best Practices

# SailPoint SaaS Change Management and Deployment Best Practices

- Overview
- SailPoint SaaS Releases
  - Release Cadence, Rollout, and Feature Flagging
  - Impact and Communication
- SailPoint Tenants and Environments
  - Sandbox Tenant
  - Production Tenant
  - Additional Tenants
- Recommended Promotion Process
  - Managing Configurations
    - Always-Managed Objects
    - Conditionally-Managed Objects
    - Non-Managed Objects
  - Promotion Process Between Tenants
    - User Interface
    - REST APIs
    - SailPoint SaaS Configuration Tools

## Overview

This document describes SailPoint's recommended best practices for managing configuration changes across customer SailPoint SaaS tenants. It includes a description of the SailPoint SaaS release process, purpose of SailPoint tenant environments, as well as recommended promotion process for configurations.

Ultimately SailPoint recommends that all organizations follow a controlled configuration, test, and deployment process, though the specific details of that process are left to each organization to define in accordance with their business practices.

Hopefully this guide helps you understand the key aspects in accomplishing that goal, and provides some guidance around best practices and recommendations from the SailPoint side.

## SailPoint SaaS Releases

When discussing how to manage changes across tenants, it is important to also have an understanding of how SailPoint SaaS releases components, so you can understand what they contain, what cadence they are released, how they are communicated, and how they may impact your business in general.

In general SailPoint's SaaS releases are made up of these items:

1. Features
2. Maintenance Fixes
3. Security Improvements
4. Architectural Changes

Releases can contain **features** which are actual net new capabilities, or enhancements to existing capabilities. These functionalities can be impactful – such as is the case with user interface or behavioral changes. They can also be less impactful too – such as is the case with additions to REST APIs, event-triggers, connectors, AI improvements, or even micro-service functionality changes.

Releases can also contain **maintenance fixes**, which are updates to existing feature capabilities to make them more resilient, and potentially fix or address particular issues that have been reported through SailPoint Support channels. These releases are typically less impactful, and welcomed to help ensure that the SailPoint SaaS platform remains a quality, stable service that customers enjoy using.

Releases can contain **security improvements**. SailPoint takes security of the SailPoint SaaS platform very seriously, and is constantly making sure that the platform is resilient to attack, and hardened with multiple layers of security. A lot of these security improvements come from internal SailPoint guidance, as well as results from regular independent penetration tests done on the platform. These improvements can be anytime from standard security patches for technologies we use, to leveraging more up-to-date encryption, or even making a micro-service less vulnerable to a type of attack. These releases are transparent to customers, and is a welcome part of what makes the SailPoint service valuable.

Releases can also contain **architectural changes**. These are changes to how the cloud runs, operates, scales, communicates, or even where and how it stores its data. These releases ensure the cloud runs optimally and can scale to meet its intended designed performance. These releases are transparent to customers, and is a welcome part of what makes the SailPoint service valuable.

## Release Cadence, Rollout, and Feature Flagging

SailPoint SaaS releases are deployed at a high rate. At current writing, our current average is sixty (60) releases a week, spanning a multitude of the SailPoint micro-services that operate the SailPoint SaaS platform. While releases occur at a high rate, most of these are transparent to anyone using the cloud, and impactful changes are communicated and announced ahead of time.

**Further Reading:** *Another great resource on this topic was published to the SailPoint Tech Blog entitled “Our Path to 60 Releases Per Week”. Give it a read!*

While feature releases are deployed, it doesn't actually mean this is enabled for live usage by all tenants at the same time. SailPoint uses a widely adopted SaaS concept called feature flagging to identify which tenants should be using what features. This allows SailPoint to control rollout of certain features in a dynamic way. This has many advantages over existing static, “all or nothing” deployment strategies.

SailPoint can roll changes out incrementally for different populations of tenants based on need. This could mean changes are rolled out to sandbox first, and then enabled in production at a later date. Most feature releases will rollout to sandbox first, and then production second. The amount of time between tenant releases typically depends on how impactful the release is.

SailPoint has more control of how how features are used and adopted. Feature flags let customer tenants opt-in to Early Access (EA) programs if they are interested, and preview features ahead of when a feature might be Generally Available (GA) to all customer tenants. This allows SailPoint to get feedback on new features, possibly improve the features further, and hone communication and documentation information for wide-spread rollout.

Feature flagging also allows SailPoint to rollout a new version (as is the case in emergency fixes) if needed, or even rollback to previous versions of micro-services, if that is a better course of action. Taken in with continuous monitoring and alerting, this allows for a more resilient platform with less impactful changes. This can ensure a resilient platform in the best way possible. This is done at the

guidance of the SailPoint Engineering and DevOps personnel and is integral to release planning, and subject to SailPoint Change Approval Board (CAB) approval.

**Further Reading:** *Another great resource on this topic was published to the SailPoint Tech Blog entitled "CI/CD Superhero: Feature Flags". Give it a read!*

## Impact and Communication

SailPoint's mission is to provide the best-in-class features within a robust and stable SaaS service. In order to do that, communication of new features is critical.

All features that are enabled are announced and communicated in several ways.

The SailPoint SaaS feature roadmap is publicly visible at **SailPoint SaaS Product Release Updates** and can be tracked through there.

Releases are also announced and communicated on the **SailPoint Compass Community** via the **Updates** section which includes rollout, timing, and impacts. This community not only serves as a place for announcements, but also serves as a place for roadmap visibility, customer discussion, and relevant product documentation and best practices.

Another way they are announced and communicated is through our in-service messaging, which is visible and located in the lower right corner of the screen. This announces new features, and way even offers walk-throughs and assistance in the service spotlighting changes once features have been released.

Last, SailPoint SaaS status and health can easily be monitored at **status.sailpoint.com** which includes uptime metrics and outages.

## SailPoint Tenants and Environments

A separated tenant environment, with controls in place for testing and migrating changes between environments, is essential to preventing avoidable production outages and minimizing the introduction of misconfigurations in the production environment. SailPoint recommends and provides two distinct tenant environments for this purpose:

1. a sandbox tenant
2. a production tenant

## Sandbox Tenant

The sandbox or test tenant is an environment is used for initial implementation and configuration of the SailPoint service, and serves as a place to test artifact configurations before live usage in a production deployment. Since this environment may act as a precursor to a production tenant, but not actual live usage, it is important to understand this environment is not typically given the same level of resources that are required for production operation.

The purpose of this environment is to to test configurations before production. It allows the implementer or administrator to configure configurations that will be used in the implementation; essentially, all core configuration should be initially done or represented the sandbox tenant. Implementers or administrators can test these items without affecting production behavior. Once testing has been satisfactorily passed, then these same configurations can be promoted to the production environment in a controlled manner. It is also important to note that sandbox environments are always left *in situ* - as they are - and are never refreshed from any other tenant, unless someone configures them differently.

Typically, a sandbox tenant usually only contains a subset of identity and account data, since the implementer only needs the data relevant to the functional tasks they are implementing. Using full, large-scale production data sets are discouraged, because some data processes, such as aggregations and identity refreshes, may take longer, or may impede configuration or testing cycles, which are better operated off smaller data sets for iterative configuration and testing. SailPoint recommends a set of 1,000 identities in this environment. Requests for larger sets of data might be accommodated, with SailPoint approval, assuming the multi-tenant space (pod) has room for that number of identities.

From a source and data quality perspective, ideally, the sandbox tenant data and sources should be as close to a copy of production as possible, within technical and budgetary constraints. Ultimately, final testing for the entire solution is validated in this environment prior to production rollout, so the more similar the environments are, the more valuable the test results will be, and the less risk there is in migration to production. Poorly planned and implemented sandbox tenants are usually the root cause of failed production deployments.

Many configuration objects created in this environment are often exported from the sandbox tenant to be migrated to the subsequent production tenant. Artifacts may optionally be checked in via a version control system, or part of a larger CI/CD process if so desired. This is described in more detail in subsequent sections of this document.

**NOTE:** *As required by any standard software release process, testing of even small, incremental changes should be done in this environment before promoting those change to production.*

## Production Tenant

The production tenant is the environment which is designed for actual live usage by users, ongoing operations, and actual integrations with other target source systems. It is the ultimate end-goal for various business, IT, and security processes, and is ultimately where all features will be used.

Because of this pattern of usage, this is the place with the most sensitive information and must be managed with the strictest quality and change controls. Ideally, great care should be taken for any changes to this environment, and everything should be tested and validated appropriately before putting those changes into live production usage.

## Additional Tenants

Some organizations may want or require additional tenants for specific purposes. Some examples include additional testing environments, performance testing environments, or even pre-production tenants which mirror production configuration. Additional tenants can certainly be accommodated with additional subscription cost, subject to the nature of the environment, and licensed according to the SailPoint Software-as-a-Service (SaaS) Subscription Agreement. If you need additional tenants to meet your needs, contact your SailPoint Customer Success Manager (CSM) for details.

## Recommended Promotion Process

Due to the multi-tenant software-as-a-service nature of the SailPoint platform, many factors of releasing, updating, security, and running the platform are handled for you, and was described in detail in the **SailPoint SaaS Releases** section.

Beyond the service itself, are the unique combination of configurations and integrations which make the SailPoint SaaS solution work for your organization. These are features uniquely tailored to meet your needs as a customer, and

SailPoint recommends following a controlled configuration, test, and deployment process to manage these.

There are two important considerations in managing configurations.

1. Which configurations to manage
2. How to manage configurations between tenants

This section discusses how and which configuration to manage, as well as how to integrate these with a cross-tenant promotion process.

## Managing Configurations

In process of managing configurations, it is important to understand which “items” should move from one tenant to another. These “items” are the various objects that make the SailPoint SaaS platform work.

Not all objects in the SailPoint SaaS solution are created equal. Some objects relate to setup and configuration of a tenant, and others may be a byproduct of usage of certain features or processes in the SailPoint SaaS Solution. These generally break these down into three categories:

1. Always-Managed Objects
2. Conditionally-Managed Objects
3. Non-Managed Objects

### Always-Managed Objects

This category of objects are things that pertain to configuration and setup of the SailPoint SaaS setup, and are typically something that SailPoint recommends to manage via some sort of change control process between tenants. Here are common configurations which we recommend, along with a brief explanation of what they are:

- **Sources** - These define the technical details of how to connect to a target system using a SailPoint connector. There may be some minor differences in these configurations across tenants which commonly pertain to connectivity or security information - e.g. the sandbox Active Directory source communicates to a test server, while the production Active Directory source communicates to a production server.
- **Source Schemas** - These are configurations control what account and access data to collect on aggregation (data import) processes. These generally match between sandbox and production, and the only reason

they differ is if data is somewhat different between tenants (i.e. account data is different)

- **Source Correlation** – These are configurations associated with sources which control how a source’s accounts find their owning identities. These generally match between sandbox and production, and the only reason they differ is if data is somewhat different between tenants (i.e. account data is different)
- **Source Account Profiles** – These are configurations which determine provisioning policy around how an account is provisioned – most commonly during account creation.
- **Source Attribute Sync Configuration** – These are configurations which determine which identity attributes should be synced to account attributes.
- **Workflows** – These are configurations which define a sequence of steps for governance related processes.
- **Identity Profiles** – These are configurations which determine the mapping of identity attributes for a set of identities associated with the identity profile’s authoritative source.
- **Transforms** – Configurations which allow for transformation or determination of account or identity attribute data in a “no-code” manner.
- **Rules** – Configurations which allow for calculations in the cloud, or connector-level operations in a “code” manner.
- **Custom Connectors** – Developed and packaged connector artifacts which are uploaded to a tenant for execution in the virtual appliance.
- **Service Desk Integration Modules (SDIM)** – Configurations which map sources’ to a ticketing system, for manual service-desk provisioning fulfillment.
- **Password Policy** – Configurations which define a sources’ password policy for password management, or automatic password generation.
- **Tenant Configuration** – Configurations which control system settings of the tenant.

## Conditionally-Managed Objects

This category of objects fall into a bit of a “grey area”, where some customers choose to migrate them across tenants while others prefer to recreate them independently in each environment. The decisions on whether or not to migrate these generally falls to the customer’s choice and their business and change control processes.

- **Roles** – These are configurations which determine assignment of access to identities. While role assignment criteria is generally ports across tenant, often times access profiles and entitlements may be somewhat different



between sandbox and production tenants. For this reason, these objects are generally not managed between tenants.

- **Access Profiles** – These are configurations which determine a bundle of entitlements. Access profiles are based on one or more entitlements, which may be somewhat different between sandbox and production tenants. For this reason, these objects are generally not managed between tenants.
- **Separation of Duties Policies** – These are configurations which define a separation of duties policy and are generally data dependent – especially around access models, which may be somewhat different between sandbox and production tenants. For this reason, these objects are generally not managed between tenants.
- **Event Trigger Subscriptions** – These are configurations which define a subscribed target to send event triggers to. While there is no reason these can't be migrated between tenants, they generally aren't managed as the event trigger subscriptions tend to be environmentally specific.
- **Branding** – These are configuration which define the colors and logos to brand the SailPoint SaaS interface with. Each customer handles branding slightly differently; some customers share similar branding across tenants, while others differentiate their sandbox and production tenants via branding schemes (e.g. production is red; sandbox is blue).
- **Email Templates** – Configuration templates which define the email messages that the SailPoint SaaS services uses in email-based communications. Each customer handles email templates slightly differently; some customers share similar email templates across tenants, while others differentiate their sandbox and production tenants' email communications schemes (e.g. production has a different email footer than sandbox).

## Non-Managed Objects

This category of objects are objects that are not designed for import and export across tenant. Some are data objects created during the normal course of usage (e.g. audit events); they may be loaded by an account aggregation task (e.g. identities, accounts or entitlements), or generated based on some feature function (e.g. certification campaigns, tasks).

- **Virtual Appliances** – Every tenant has different virtual appliance clusters. While these can be named similarly cross-tenants, this is usually part of one-time tenant setup and not something you should migrate between environments.
- **Identities** – Identities are built in accordance with the identity profile mappings, and based on aggregated account and authoritative source information. Alternately, these may come from Non-Employee Lifecycle

Manager (NELM) processes, which are a by-product of actual tenant usage. Identities are not something that can be migrated from one tenant to another.

- **Accounts** - Accounts are aggregated from source account aggregations and are not something that can be migrated from one tenant to another.
- **Entitlements** - Entitlements are aggregated from source entitlement (group) aggregations and are not something that can be migrated from one tenant to another.
- **Audit Events** - These objects are generated as a by-product of actual tenant usage, and because they relate to auditing, they are actually read-only. They cannot be migrated from one tenant to another.
- **Account Activities** - These objects are system generated to describe in-flight or recently completed access request or provisioning activities. These are generated as a by-product of actual tenant usage, and they cannot be migrated from one tenant to another.
- **Certification Campaigns** - These objects are created by generating a certification campaign. These are a result of usage of the system, and based on identity and account data, so they cannot be migrated from one tenant to another. Similar certification campaigns can be configured in each tenant, they just can't be migrated.
- **Tags** - Tags are created as a by-product of usage of the search features, and cannot be migrated cross-tenant.
- **Tasks** - These objects are a result of manual provisioning or approval processes and are a by-product of actual tenant usage. They cannot be migrated cross-tenant.
- **Saved Searches** - These saved searches are created as a by-product of usage of the search features, and cannot be migrated cross-tenant.

*Don't see something? Feel free to ask on our Compass Community!*

## Promotion Process Between Tenants

There are several ways that people move configurations between SailPoint SaaS tenants. Here are some of the common ways that are used, and how they are applied.

Each approach has different tradeoffs, depending on level of technical ability, or desired level of automation to meet your needs. These generally break these down into three categories:

1. User Interface
2. REST APIs
3. SailPoint SaaS Configuration Tools

## User Interface

The user interface is the most obvious choice in configuring the SailPoint SaaS Service, as it merely involves an administrator configuring things the same way in one tenant as they are in another tenant. This is very approachable, and doesn't require any special knowledge or technical tooling in order to accomplish. Any administrator can simply open two browser windows and mimic configurations in each environment.

While this is doable, this does have its drawbacks. Managing configurations this way is fairly tedious, and since it is manual, it is prone to human error. For this reason, we do not recommend this for large, complex deployments.

The other drawback is that if there are advanced configurations made via REST APIs, these may not immediately evident in the user interface. These are typically things that are only noticed via REST APIs.

## REST APIs

The REST APIs are a technical interface which can configure or do anything in the system. The SailPoint SaaS user interface actually relies on the REST APIs to do its bidding. So anything you setup, administer, or configure in a tenant, is actually doable with REST API calls.

Because REST APIs are a technical interface, they can leveraged to configure the *Always-Managed* or *Conditionally-Managed Objects* in the system, in an automated fashion. This can eliminate the tedious configuration promotion that might be present with other methods.

SailPoint finds this process is often also used by customers who wish to plug SailPoint SaaS into a Continuous Integration / Continuous Deployment (CI/CD) pipeline or Change Management (CM) process. This is seen as a way of easily automating the tedious process from the user interface, while making sure that configuration changes are managed. The configuration data in this process is also often captured and committed in a Version Control (e.g. GitHub) system if so desired.

SailPoint does offer REST APIs specifically for managing bulk import and export of configurable objects. Detailed information can be found here: **SaaS**

### **Configuration Import and Export**

- Start Object Export
- Get Export Status

- Get Export Results
- Start Object Import
- Get Import Status
- Get Import Results

While REST APIs are a great way to ensure accuracy in any configuration promotion between tenants, it does have a few tradeoffs. REST APIs are fairly technical in nature, and require a decent amount of technical skill and knowledge in order to use. Invoking REST APIs may require some development or scripting knowledge, and many REST API calls will be necessary to migrate all configurations.

**Tip:** For a reference of available REST APIs to support promotion of configurations between tenants, see [developer.sailpoint.com](https://developer.sailpoint.com). This is a great resource around all SailPoint SaaS APIs.

## SailPoint SaaS Configuration Tools


Beyond the user interface and the REST APIs, there is a desire for field tooling to automate configurations import and export, and assist the configuration promotion process between tenants. This tooling strikes a balance between the ease of usage, with the comprehensive visibility of the REST APIs, without technical knowledge of the intricacies of the SailPoint SaaS REST APIs.

Other tools and frameworks also exist, and we encourage our partners to develop and share these kinds of tools on the SailPoint Developer Community.

API Best Practices Business Process Cloud Configuration  
 Deployment Tips and Tricks Getting Started Implementation Resources  
 Professional Services REST API

cd  
 change ci ckd configuration export identitynow idn import management  
 promote saas

Add tags

 12 Kudos

Submit Content Feedback

Comment

### Comments



chrisp

Jun 18, 2021 02:1

Is there any update on **IdentityNow IO** ?

---



ankit200

Jul 15, 2021 10:51

Where I can find/download **IdentityNow IO** ?

---



pagi

Dec 19, 2023 09

@neil\_mcglennon Should the Configuration Hub added in SailPoint SaaS Configuration Tools?

---



drosenbauer

Jan 08, 2024 11:1

Somebody needs to update this to include Configuration Hub.

---

Powered by



