Releases

TABLE OF CONTENTS [[toc]]

Targeted Release

Benefits

Targeted release allows admins, change managers, or anyone else responsible for SalesTim updates to prepare for the upcoming changes by letting them: - Test and validate new updates before they are released to all the users in the organization. - Prepare user notification and documentation before updates are released. - Prepare internal help-desk for upcoming changes. - Go through compliance and security reviews. - Use feature controls, where applicable, to control the release of updates to end users.

Setup

Access to targeted release is controlled by an RBAC policy. To grant a user access to targeted release features: 1. Open the `Settings` tab 2. Open the `Roles (RBAC)` section 3. Assign the `Change Manager` role to the user 4. Click `Save`

Validation Rings

Our release process is comprised of multiple "rings" of validation that are related to specific environments, to a specific audience and a specific compliance labeling level:

Ring	Environments	Primary Audience	Purpose
4	**production**	Customers (All)	Obvious isn't it? ;-)
3.5	**staging**	DevOps Team	Test automated deployments and upgrades in an iso-production environment
3	**beta**	Customers (Preview)	Preview environment designed to help some customers prepare for updates, from a technical and change management standpoint
2	**uat**	Product Team	The product team tests the app to verify whether it meets their expectations

Ring	Environments	Primary Audience	Purpose
1.5	**alpha**	Partners (SI/ISV)	Give strategic partners an early look at the features we're currently working on
1	**dogfood**	Internal	Internal Dogfooding
0	**integration**	DevOps Team	Integrations and functional testing by the tech team

Using this kind of rings has many advantages: * Clear and common understanding of each ring purpose * Separation of concerns * Real isolation between environments * Enforced security

Versioning Strategy

Our versioning strategy adheres to Semantic Versioning.
A version number may be comprised of 3 to 4 components and takes this form:

`MAJOR.MINOR.PATCH-BUILD`

Meaning of each component: * MAJOR: version that includes incompatible changes (data schema, api signatures...) * MINOR: version that includes functionality in a backwards-compatible manner * PATCH: version that includes backwards-compatible bug fixes * BUILD: incremental development-only version