Veeva

WHITE PAPER

Leveraging Snapshots for Compliant Validation in Veeva Vault

AUTHORS & CONTRIBUTORS

Llinos Cooper | Veeva Systems, Services Lead, Validation Strategy
Krisztián Csobályka | Veeva Systems, Product Expert, Vault Platform
Michael Ferrell | Veeva Systems, Senior Product Expert, Vault Platform
Darek Mankowski | Sanofi, Service Owner
DJ Joshi | Viridian Therapeutics, IT Director
Hemal Shah | Sanofi, Technical Expert
Slawek Sugier | GSK, Technical Lead, Vault Platform

Executive summary

This whitepaper presents best practices for using Snapshots to validate changes in Veeva Vault, focusing on customer configuration releases. This whitepaper is intended to complement and extend the Risk-based Approach to Change Management of GxP Systems whitepaper.

Snapshots are point-in-time sandbox copies containing *configuration*, *documents*, and *data* that can support *testing*, *training*, and *validation*. Snapshots enable concurrent validations, especially during emergency changes.

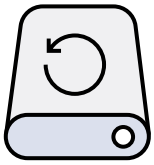
Validation Environments are Vaults that mirror production configuration for change testing. They are verified through *Compare Reports*, and multiple validation environments can be available simultaneously.

Pre-release Vaults allow validation of Veeva's updates before General Releases, showing the feasibility of multiple *Validation Environments*. Veeva Vault employs a GAMP 5 risk-based approach, classifying it as Category 4 (Configured Software) with tailored validation efforts based on change risk.

Contents

Background

What are Snapshots?

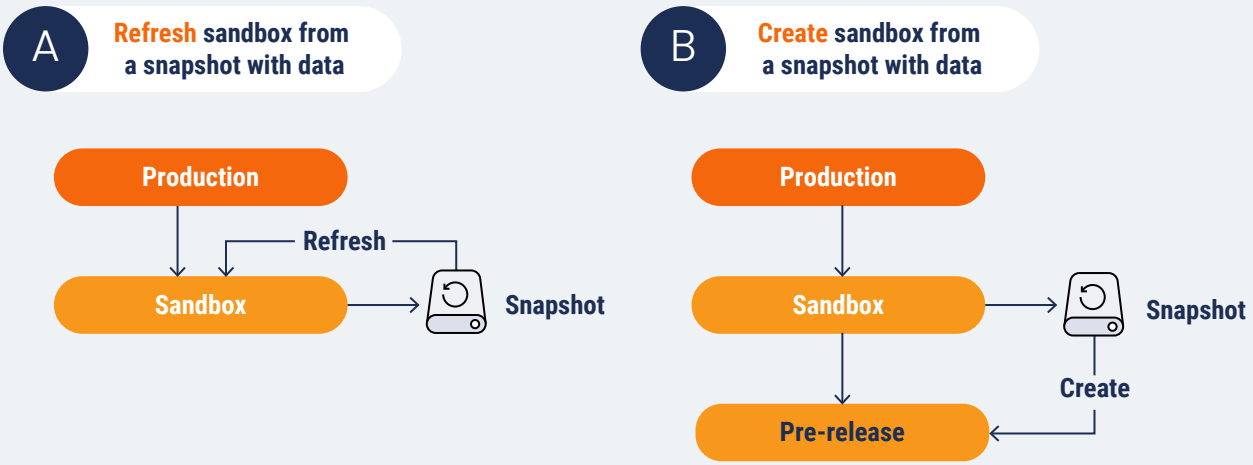


Veeva Vault Snapshots provide a powerful mechanism for managing and validating changes within a regulated environment. A **Snapshot** is a *point-in-time* copy of a sandbox vault. This copy encapsulates the Vault's configuration, system-managed data, and *optionally* the transactional data and documents, preserving the state of the vault at the moment the snapshot was taken.

Why use snapshots?

Snapshots offer several key benefits that streamline validation activities and enhance compliance.

Repeatable testing	One of the primary advantages of snapshots lies in their ability to facilitate repeatable testing. By loading a sample dataset onto a sandbox and capturing it via a snapshot, any subsequent modifications to the sandbox can be reset to the initial state. This allows for repeated testing based on different criteria and scenarios, ensuring comprehensive validation coverage.
Sandbox version control and rollback	When using a controlled process that takes snapshots before a set of changes are made to a sandbox environment, the snapshot can be used as a form of version control and enables users to roll back the Vault to the state it was in prior to the set of changes being deployment.
Pre-release sandboxes	Veeva provides a pre-release sandbox license four weeks before each general release. Customers can create the pre-release sandboxes from snapshots, which are pre-loaded with datasets that allow repeatable regression testing. This allows for early testing and validation of new features and functionalities, reducing the time required for validation activities after the general release is deployed.
Validation use cases	Snapshots are invaluable tools for various validation-related tasks. They enable the creation of multiple validation vaults, which is particularly beneficial for validating emergency changes that occur during planned release validation. This approach minimizes disruption to ongoing validation efforts and ensures timely deployment of critical bug fixes.



Veeva Vault validation

What is a validation environment?

In the context of Veeva Vault, a **validation environment** is any Vault that replicates the production environment's configuration. It is used to thoroughly test and validate changes before they are deployed to the live production system. **Any Vault** that can be verified to have the same configuration as the production Vault can be considered a validation environment.

The key to establishing a Vault as a validated environment lies in demonstrating that its configuration matches the production environment. Vault Configuration consists of three elements:

- **Components** (Document Types, Objects, Lifecycles, etc.)
- **Configuration Data** (Data used by used Component, and optionally Reference Data)
- **Custom Code** (Vault Java SDK, Custom Pages)

Veeva provides the ability to generate a Compare Report, which is a validated report used to compare **Components** and **Custom Code** from the Vault Configuration. For Vaults that do not have data-driven *User Requirements Specifications* for the changes delivered in a customer release, this report is enough when following a *Risk-Based Approach*.

Comparing the validation Vault with the production Vault using the Compare Report Analysis process, organizations can confidently assert that a sandbox is a true reflection of the production environment.

You can find more information about Vault Compare in ComplianceDocs > Validation Packages > Vault Platform and Core Applications, by searching for "Vault BRD: Loader and Migration" in the current releases Vault Validation binder.

To compare **Configuration Data** or *Reference Data* only when a customer release involves changes with data-driven *User Requirements Specifications*, administrators have to extract the records and then perform a comparison on them with any method outlined in their process documentation.

Compare report analysis

A Compare Report will contain **Impacting** and **Non-Impacting** differences, which can be determined by analyzing it. If a Compare Report only contains Non-Impacting differences, the two Vaults can be considered identical for the purposes of validation.

Non-Impacting differences do not alter the way the system behaves or changes user experience noticeably to end users. *System Administrators* may still see these differences on the back end. Organizations should build a repository of accepted **Non-Impacting** differences, that can be referenced when documenting the *Compare Report* in the change control process.

Impacting differences are anything that does not fall into the Non-Impacting differences category.

Is the pre-release vault a validation environment?

Veeva offers **pre-release vaults** four weeks before each general release. These vaults can serve as temporary validation environments, enabling customers to test and validate the new features and changes introduced in the upcoming general release. A pre-release vault for a specific general release is considered a validation environment for validating (verifying at a minimum) the auto-on changes associated with that general release.

Pre-release Vaults are essentially copies of the production environment with the new general release changes applied. They provide customers with an opportunity to assess the impact of these changes, perform required testing, and ensure their systems are ready for the general release deployment.

The existence of pre-release vaults exemplifies that the concept of a single validation environment doesn't strictly apply to SaaS and cloud-based systems like Veeva Vault. In reality, organizations can have multiple validation environments, as illustrated by the availability of both the customer's designated validation vault and the temporary pre-release vault provided by Veeva.

To learn more about Pre-release Vaults please visit the Pre-Release FAQ on Vault Help.

Veeva validation methodology overview

Veeva adheres to a robust validation methodology based on the principles of Computer System Validation (CSV). CSV involves establishing documented evidence that a computer system meets predefined requirements. This methodology ensures that Veeva Vault operates as intended and complies with relevant regulatory requirements.

Veeva's responsibility lies in the qualification of the hosted environments and the validity of the core software. This includes:

- **Installation Qualification (IQ):** Veeva performs IQ for the core system components, including the operating system, security features, and data privacy controls. This ensures the proper installation and configuration of the underlying infrastructure.
- **Operational Qualification (OQ):** Veeva conducts OQ to verify that the core system functionalities meet the defined requirements. This encompasses functionalities like standard objects, lifecycles, and workflows, which are integral parts of the Veeva Vault platform and core application suite.

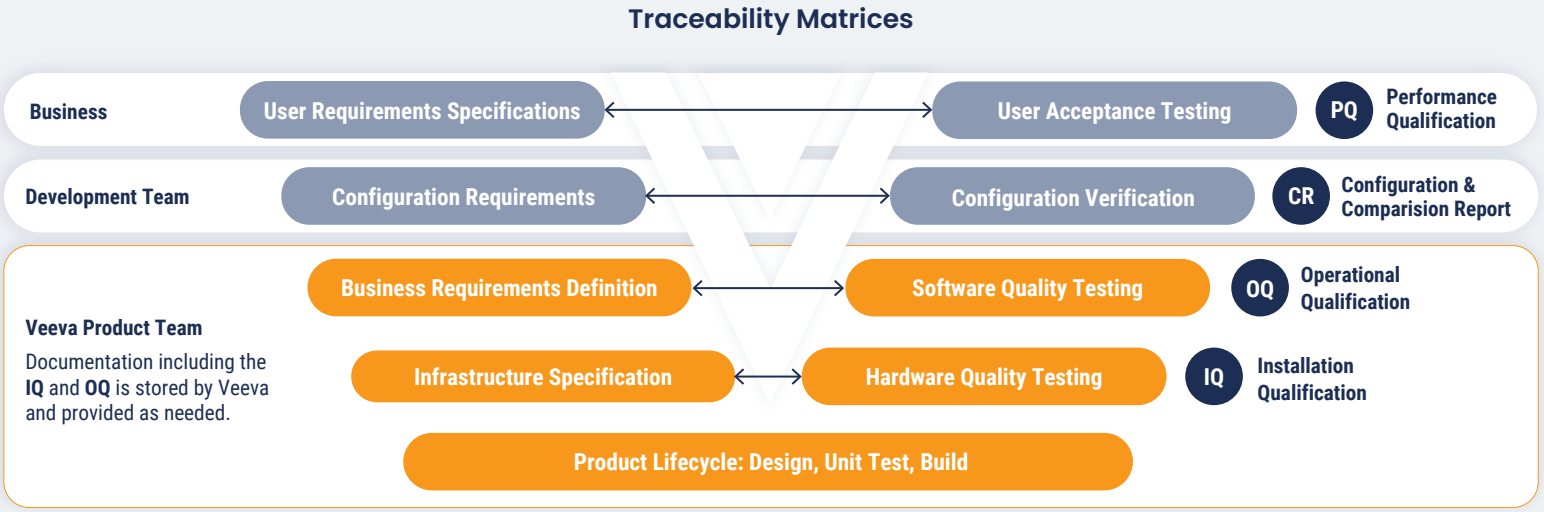
Customer's responsibility lies in the validation of the specific configurations and customizations implemented within their Veeva Vault environments. This typically involves:

- **User Acceptance Testing (UAT):** UAT focuses on verifying that the system meets the user requirements and business processes as configured.
- **Performance Qualification (PQ):** PQ assesses the system's performance under real-world conditions, ensuring it can handle the expected workload and maintain data integrity.

Layered approach: Veeva Vault employs a layered approach, consisting of:

- **Platform Layer:** This foundational layer provides the core infrastructure and functionalities upon which all Veeva Vault applications are built.
- **Core Application Layer:** This layer includes pre-built applications like Quality Docs, eTMF, and Safety. These applications come with core requirements and functionalities that have been validated by Veeva.
- **Configuration Layer:** Customers build their specific business processes and functionalities on top of the core applications through configurations.

Veeva's rigorous validation of the platform and core application layers ensures the underlying engine driving customer configurations is reliable and compliant. This layered approach allows customers to focus their validation efforts on their unique configurations and business processes.



Risk-based approach to validation

GAMP 5 and Veeva Vault

Veeva's validation methodology aligns with the principles outlined in GAMP 5 (Good Automated Manufacturing Practice), a set of guidelines developed by the International Society for Pharmaceutical Engineering (ISPE). GAMP 5 provides a framework for ensuring that automated systems used in pharmaceutical manufacturing are properly validated and compliant with regulatory requirements, such as those from the FDA and EMA.

KEY ASPECTS OF GAMP 5

- **Risk-Based Approach:** Focuses validation efforts on critical system functionalities that impact product quality, patient safety, and data integrity. Identifies and mitigates risks by prioritizing areas of highest concern.
- **Lifecycle Approach:** GAMP 5 emphasizes the importance of validation throughout the entire system lifecycle. This encompasses initial implementation, ongoing changes and updates, and eventual retirement of the system.
- **Flexible Approach:** GAMP 5 recognizes that validation efforts should be proportional to the complexity and risk associated with the system. This means that less complex, lower-risk systems might require a streamlined validation approach, while more complex and higher-risk systems necessitate a more comprehensive validation strategy.
- **Supplier and Service Provider Involvement:** GAMP 5 acknowledges the crucial role of suppliers and service providers in providing evidence and documentation to support customer validation efforts.
- **Data Integrity:** GAMP 5 places significant emphasis on data integrity, ensuring that data is accurate, complete, consistent, attributable, available when needed, and protected from unauthorized alteration.

GAMP 5 CATEGORIES

GAMP 5 categorizes software based on its level of customization and associated risk:

- **CATEGORY 3 - Non-Configured Software:** This category comprises software that cannot be customized, such as commercially available applications like Microsoft Excel.
- **CATEGORY 4 - Configured Software:** This category includes software tailored to fit the specific needs of a process or system. Configuration might involve defining workflows, or configuring lifecycles within a document management system.
- **CATEGORY 5 - Custom/Bespoke Software:** This category encompasses software developed from scratch to fulfill unique business requirements. Custom software development poses a higher risk due to the complexity of the development lifecycle and the potential for introducing errors in the code.

Veeva Vault as GAMP 5 Category 4

Veeva Vault falls under GAMP 5 Category 4 - Configured Software, as it is a configured software system. The configurations applied to Veeva Vault do not involve writing Custom Code. They are implemented through a user-friendly interface that constrains options and enforces predefined rules, reducing the potential for errors.

Custom Code deployed to Vault would be considered *Category 5 - Custom/Bespoke Software*.

Leveraging Veeva documentation

As a GAMP 5 Category 4 software system, Veeva Vault allows customers to leverage Veeva's extensive documentation and verification activities to streamline their validation efforts. The Veeva validation team performs comprehensive testing and generates documentation that provides evidence of the system's functionality and compliance. Customers can utilize this documentation to support their own validation activities, focusing on verifying that their specific configurations meet their business requirements.

Risk classification examples

A **risk-based approach** is essential for efficient and effective validation of Veeva Vault configurations. System changes vary in risk level, and identifying the associated risk ensures the appropriate level of validation rigor is applied.

For detailed information about Veeva's approach, please read our white paper discussing this topic in detail: Risk-based Approach to Change Management of GxP Systems.

FACTORS INFLUENCING RISK

Several factors contribute to the overall risk associated with a change in Veeva Vault. These may include:

- **GxP Impact:** Changes that directly impact GxP-regulated activities, such as data integrity, audit trails, electronic signatures, or security controls, pose a higher risk and require more stringent validation.
- **Impact on Core Functionality:** Changes affecting the core functionalities of Veeva Vault, including workflows, lifecycle management, document management, and security settings, are generally considered medium risk, requiring a thorough assessment and validation. In other words, changes in Vault related to a business process that alters Vault's behavior and logic when responding to inputs are considered medium risk.
- **Impact on Non-Core Functionality:** Changes impacting non-core aspects of the system, such as metadata fields, notifications, reports, user interface elements, or administrative functions not related to security or regulatory compliance, are typically deemed low risk.

EXAMPLES OF RISK CLASSIFICATION

The following table provides examples of how different areas within Veeva Vault could be classified based on their potential risk:

Risk Level	Veeva Vault Areas
High	Permissions, Login, Security Policies, Security Profiles, Audit Trail, eSignature, Document Change Control, DAC (Dynamic Access Control)
Medium	Creating, updating, deleting documents; Creating, updating, deleting objects; Workflow operations; Lifecycle operations; Check In/Check Out; Configuring reports; Versioning
Low	Document library, Creating/editing fields (fields that do not contain VQL/constraints), Reporting and dashboards (where a quality decision is not made based on the results of these reports), Annotations, System Look & Feel (GUI), Localizations/translations, Labels, Exporting, Search, Field dependency, Breadcrumb navigation, Notifications, Managing data, Field level security

DOCUMENTING RISK DEFINITIONS

Organizations must clearly document their risk definitions and the criteria used to classify changes within their Veeva Vault environments. This documentation should be included in their validation methodology, validation plans, or other relevant SOPs. A well-defined risk classification system enhances consistency and transparency in validation efforts.

Emergency change scenario

Overview of the scenario

Let's consider a scenario where an organization is in the midst of validating a planned release in their Veeva Vault validation environment. UAT/PQ testing is progressing smoothly when a critical bug is discovered in the production environment. This bug requires immediate resolution to ensure the continued operation of the system.

In this situation, the organization faces a dilemma:

- **Disrupt Planned Release Validation:** They could halt the ongoing validation activities for the planned release, roll back the changes deployed to the validation environment, and use that environment to validate the emergency bug fix. This approach would result in delays and require the validation team to redo the work already completed for the planned release.
- **Delay Bug Fix:** They could postpone addressing the production bug until the validation of the planned release is finished. This would leave the production system in a compromised state, potentially impacting critical business processes and user productivity.
- **Utilize an Additional Validation Environment:** This is where snapshots offer a valuable solution.

Recommended snapshot strategy for creating and refreshing validation environments

Leveraging snapshots allows organizations to maintain compliance and validate emergency changes without disrupting ongoing planned release validation activities. Two types of snapshots are recommended to be created and maintained in the primary validation environment. These snapshots should be used to setup baseline development sandboxes, and to create additional validation environments in case of emergency changes. The process of how these environments and snapshots should be used is covered in Version Control and Rollback with Snapshots.

Recommended snapshots in the primary validation environment:

- The **Steady State Snapshot** represents the current configuration matching the production environment. It should not contain any active users other than *Vault Owners*, *System Administrators*, and *Integration Users*. It should also not contain any transactional data other than the ones defined for testing purposes.
- The **Future State Snapshot** represents the potential next *Steady State Snapshot*, and it should be created *after* the deployment of the planned release in the Validation environment, but *before* any testing begins.

Please note: Neither of the snapshots should be literally named “Steady State” or “Future State” as the *Future State Snapshot* will evolve to the Steady State Snapshot. Instead use the name of the customer release, like *24R3-VernBio-R1*.

Steps for using snapshots to validate emergency changes

The below process contains steps that are done as part of a regular validation deployment and testing process, as well as the emergency steps. Non-emergency steps will be marked with the [Standard] suffix.

Steps Involved in an Emergency Change Scenario:

1. **Establish the Validation Environment Baseline [Standard]**

   To demonstrate compliance with QA and any additional stakeholders, use the compare report and any additional configuration data verification steps to verify that the *primary validation environment* matches the current production environment. This establishes the environment as a validated environment.

2. **Prepare the Validation Environment prior to Deployment [Standard]**

   - **If Steady State Snapshot does not exist** then *CREATE* it after verifying that the Validation Environment matches the Production Environment.
   - **If Steady State Snapshot is superseded** then *UPDATE* it after verifying that the Validation Environment matches the Production Environment.
   - **If Steady State Snapshot is up-to-date** then *REFRESH* the Validation Environment to ensure no accidental changes are present.

3. **Deploy Planned Release Changes [Standard]**

   Deploy the planned release changes to the primary validation environment.

4. **Create the Future State Snapshot prior to the Validation Testing [Standard]**

   Create the *Future State Snapshot* and once it is ready, the validation team can proceed with executing UAT/PQ scripts for the planned changes.

5. **Emergency Bug identified in Production**

   - While the Validation Testing is in progress a critical bug is identified in production.
   - Upon discovery of the critical bug in production, a change control is initiated.
   - A fix for the bug is developed in a sandbox environment created from the up-to-date *Steady State Snapshot*.

6. **Create a new Validation Environment from the Steady State Snapshot**

- Use the *Steady State Snapshot* to create a new *emergency validation environment*. This new environment will be a clean copy without any of the deployed changes for the currently ongoing planned release. The *Steady State Snapshot* always represents the current production environment configuration with the added benefit of test data being present.
- To demonstrate compliance with QA and any additional stakeholders, use the compare report and any additional configuration data verification steps to verify that e*mergency validation environment* created from the snapshot matches the current production environment. This establishes the newly created environment as a validated environment.

7. **Validate the Emergency Bug Fix**

- Deploy the bug fix to the *emergency validation environment* newly created from the *Steady State Snapshot*.
- Perform necessary validation activities to ensure the bug fix effectively addresses the issue and doesn't introduce any unintended consequences or regressions.

8. **Deploy the Bug Fix to Production**

After successful validation in the *emergency validation environment*, deploy the bug fix to the production environment. This resolves the critical issue and brings the production system back to normal operation. You can use any method (manual or deployment packages).

9. **Verify that the Emergency Validation Environment and Production are aligned**

To demonstrate compliance with QA and any additional stakeholders, use the compare report and any additional configuration data verification steps to verify that *emergency validation environment* created from the snapshot matches the current production environment.

10. **Assess Impact on Planned Release Validation**

- Perform an impact assessment to determine the potential impact of the emergency bug fix on the planned release that's currently undergoing validation in the *primary validation environment*.
- Identify any areas where the bug fix might affect the planned release changes or the existing validation scripts. This can be done by comparing the affected components, code, and data, then discussing any items that were changed *both* by the emergency change and the planned release.

11. Merge the Emergency Change to the Primary Validation Environment

Merge the emergency changes (manually or with deployment packages) to the *primary validation environment* that is used for the planned release.

12. Perform Regression Testing in the Primary Validation Environment (if required)

If the impact assessment reveals potential conflicts, conduct regression testing on the affected areas to ensure the bug fix hasn't negatively impacted the functionalities associated with the planned release.

13. Deploy Planned Release to Production [Standard]

After the validation testing and any necessary regression testing is completed, deploy the planned release changes to the production environment. Make sure the deployment checklist and deployment steps are updated to account for the emergency change that has already been deployed to production.

14. Verify that the Primary Validation Environment and Production are aligned

To demonstrate compliance with QA and any additional stakeholders, use the compare report and any additional configuration data verification steps to verify that the configuration of the primary validation environment with the emergency changes matches the current production environment. After the verification is completed, the *emergency validation environment* can be destroyed.

15. Upgrade the Future State Snapshot to include the emergency changes

THIS STEP IS ONLY REQUIRED IN CASE EMERGENCY CHANGES WERE DEPLOYED IN PARALLEL TO A PLANNED RELEASE. It is required to ensure the snapshot remains clear of transactional data created during the execution of test scripts. These steps **do not** need to be executed for planned releases *without* emergency changes.

- Create a *post-release validation environment* from the Future State Snapshot and merge the emergency changes to the new Vault.
- To demonstrate compliance with QA and any additional stakeholders, use the compare report and any additional configuration data verification steps to verify that the configuration of the *post-release validation environment* matches the current production environment.
- Create the *Future State Snapshot* from the post-release validation environment.
- Upon successful creation of the new *Future State Snapshot*, delete the existing *Future State Snapshot* from the primary validation environment.
- Transfer the *Future State Snapshot* from the post-release validation environment to the *primary validation environment* by changing the source sandbox of the snapshot.

16. Update Snapshot References in Documentation after Release is Complete [Standard]

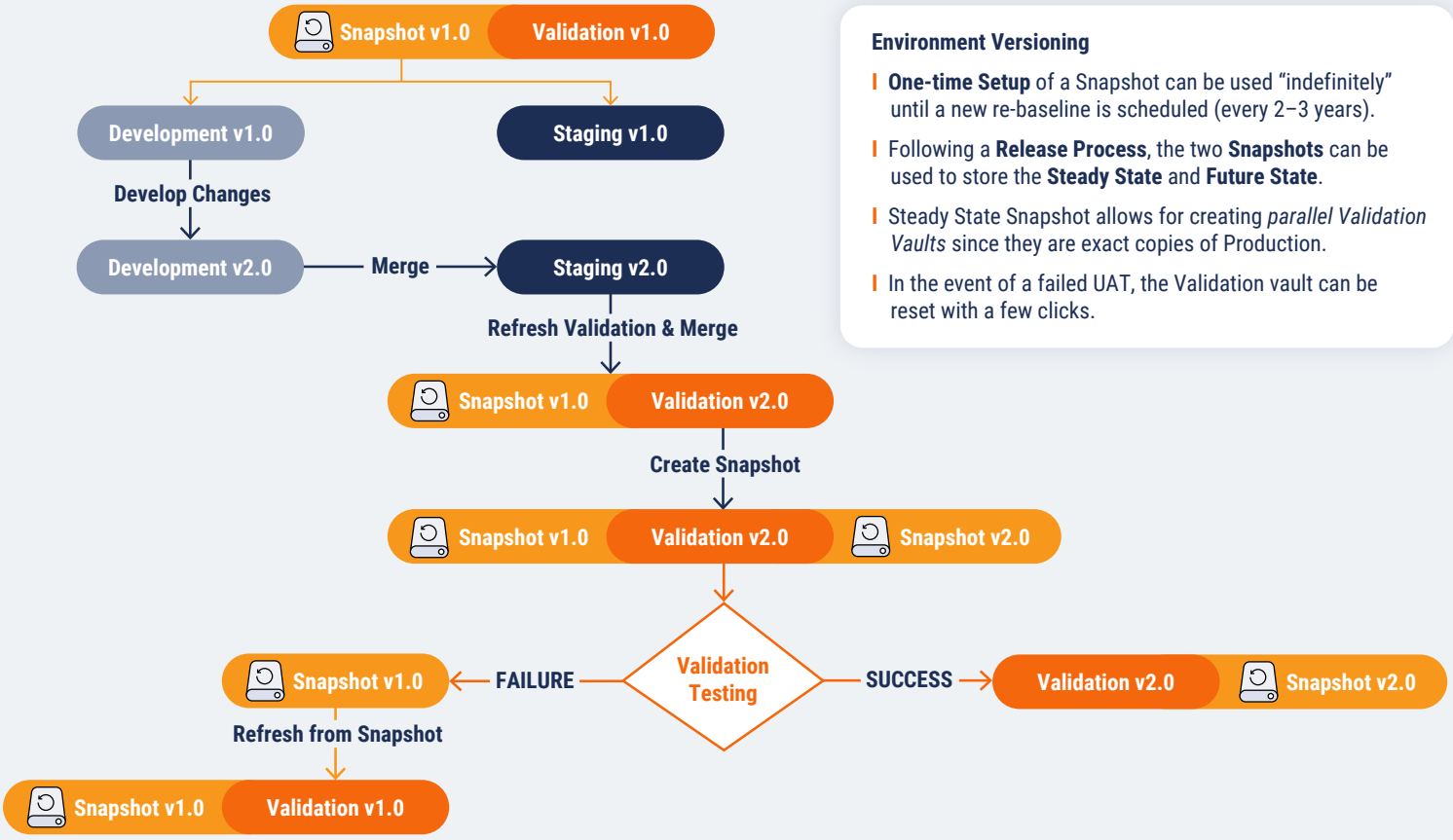
The current Steady State Snapshot should become the new *Legacy State Snapshot*, while the current *Future State Snapshot* should become the new Steady State Snapshot. THIS IS JUST DOCUMENTATION OUTSIDE OF VAULT; no actual actions are required inside Vault unless it is to delete the *Legacy State Snapshot* to avoid confusion in the future.

17. Port Back Bug Fix to Sandboxes

After completion of the impact assessment and confirmation of synchronization, port the bug fix back to all other relevant sandbox environments. This ensures consistency across all development and testing environments.

Version control and rollback with snapshots

Process map



Environment Versioning

- **One-time Setup** of a Snapshot can be used "indefinitely" until a new re-baseline is scheduled (every 2–3 years).
- Following a **Release Process**, the two **Snapshots** can be used to store the **Steady State** and **Future State**.
- Steady State Snapshot allows for creating *parallel Validation Vaults* since they are exact copies of Production.
- In the event of a failed UAT, the Validation vault can be reset with a few clicks.

Process steps

1. The process begins with a validation environment that has a *Snapshot for Version 1.0*, which represents the current configuration matching the production environment, also known as a *Steady State Snapshot*. This snapshot should **not** contain any active users, other than *Vault Owners*, *System Administrators*, and *Integration Users*. If there are already users in there that shouldn't be, those can be inactivated and the snapshot can be *updated* so the inactivated users won't show up as active users when an environment is made from a snapshot.
2. The next step is to create or refresh development and staging environments from *Snapshot for Version 1.0*.
3. Configuration changes for *Version 2.0* are continuously implemented in the development environments.
4. All changes from the development sandboxes are merged into the staging vault (one by one or all at once), making it *Version 2.0*. Not all development sandboxes need to be part of the merge, but if they are not, after the successful Validation Testing, all changes in Version 2.0 need to be backported to all development sandboxes that weren't part of the release. These change must be *unconditionally* accepted by the development sandboxes.
5. The validation environment is refreshed from the *Snapshot for Version 1.0*. This step ensures alignment between the validation and production environments. This removes accidentally added configuration, data, or users.
6. The *Version 2.0* changes are applied to the refreshed validation environment. These changes can include deployment packages, manual configurations, and reference data.
7. Before adding any transactional data and test users, create a new snapshot of the validation environment. This Snapshot for Version 2.0 (Future State Snapshot) represents the potential future steady state if the validation testing is successful.

   > ***Important:*** *Reference and configuration data should be added before taking the Future State Snapshot. Transactional or migration data should be entered after creating the snapshot. This helps prevent the snapshot from becoming too large and unusable with smaller sandboxes.*

8. During validation testing, if issues arise, use *Snapshot for Version 1.0* (*Steady State Snapshot*) to revert the validation vault to its previous state (rollback). If validation testing is successful, the *Snapshot for Version 2.0* (*Future State Snapshot*) becomes the new steady state snapshot.
9. Backport the changes for Version 2.0 to any development sandboxes that weren't part of the release.

This process can be repeated for each release, enabling consistent version control and rollback capabilities. Snapshots **must be upgraded** between Veeva General Releases to ensure the snapshots remain usable indefinitely.

Steady State Snapshot also allow creating parallel validation environments, enabling emergency changes on a separate track without affecting ongoing validation testing.

By following this process, organizations can leverage snapshots effectively for version control, rollback, and efficient environment management.

How to stay compliant

Suggested documentation

Organizations must maintain comprehensive documentation to demonstrate compliance with regulatory requirements and internal quality standards. This is especially important when utilizing snapshots for emergency change validation. The following documents or their equivalents within the organization's quality system may be updated to reflect the use of this approach.

- **Change Management SOP:** The change management SOP may outline the process for handling emergency changes, including the use of snapshots to create separate validation environments. It should detail the steps for creating, using, and decommissioning these environments.
- **Environment Management SOP/Work Instruction:** This document may define the procedures for managing different environments within the Veeva Vault landscape. It should include guidance on using snapshots to create and maintain validation environments.
- **Computer System Validation (CSV) SOP:** The CSV SOP should explicitly address the use of snapshots in the context of validation. It should explain how snapshots can be used to create compliant validation environments and describe the process for verifying that these environments accurately reflect the production configuration.
- **Validation Plan Template:** The validation plan template should be updated to include considerations for potential emergency change scenarios. This might involve outlining the steps for creating a snapshot of the validation environment before deploying planned release changes and defining the process for conducting impact assessments and regression testing if an emergency change is required during planned release validation.
- **Impact Assessment:** Each emergency change should be accompanied by a thorough impact assessment. This document should evaluate the potential impact of the change on other system functionalities, ongoing validation activities, and regulatory compliance. It should also propose mitigation strategies to address any identified risks.

ADDITIONAL DOCUMENTATION

- **Validation Summary Report:** The impact assessment for the emergency change should be appended to the validation summary report for the planned release. This provides a complete audit trail of the validation activities performed for both the planned release and the emergency change.
- **Test Plan:** If the impact assessment necessitates regression testing, the test plan should be updated to include the new test cases. These test cases should specifically address the areas where the emergency change might have introduced potential conflicts with the planned release functionalities.

Additional considerations

Risk Assessment: A thorough risk assessment is crucial for determining the appropriate level of validation rigor for the emergency change. Consider the potential impact on GxP activities, core system functionalities, and data integrity.

User Requirements Specifications (URS): The emergency change may or may not impact the URS. Assess and update the URS accordingly to ensure alignment with the implemented changes.

Communication and Collaboration: Maintain open communication with stakeholders throughout the process, including the validation team, QA, IT, and business users. Collaboration ensures everyone is aware of the emergency change, its potential impact, and the steps being taken to maintain compliance.

Leveraging Veeva Expertise and Resources: Remember that Veeva is a valuable resource for guidance and support. They can provide insights into risk assessment, validation best practices, and documentation requirements.

By implementing a well-defined process, maintaining thorough documentation, and collaborating effectively with stakeholders, organizations can leverage snapshots to validate emergency changes in Veeva Vault while adhering to regulatory requirements and minimizing disruption to ongoing validation efforts.