Safire

Safire Source Vault System V1

End User and Software Developer Guide

Product-facing edition

This guide describes Source Vault concepts, checkpoint lists, repository state, restore preview, recovery, database backup, database restore preview, diagnostics, and developer-facing usage as part of Safire software development.

- Checkpointed source-first recovery for Safire development.
- Restore preview before live recovery.
- Explicit approval before checkpoint creation.
- Visible evidence, logs, acceptance files, and clean review.

Contents

1. Purpose and audience

This guide explains Safire Source Vault for end users, support staff, administrators, and software developers. It describes what Source Vault is, how checkpoints are used during Safire development, and how the Windows interface exposes safe actions such as status, compare, restore preview, recover, database backup, database restore preview, and database restore.

End users and administrators use this guide to understand safe recovery operations. Software developers use it to make checkpoints part of normal Safire development, package delivery, acceptance testing, and rollback planning.

The guide is product-facing. It describes supported concepts and workflows, not temporary build notes or private implementation experiments.

2. Product overview

Safire Source Vault is the project-level checkpoint and recovery product used by Safire development work. It protects source trees, generated product files, documentation, acceptance evidence, and selected database backup evidence by creating named checkpoints that can be verified, compared, and restored through controlled commands.

The Source Vault core performs repository operations. The Safire Source Vault user interface presents those operations in a Windows desktop application named Safire Source Vault. The application gives a user a visible checkpoint list, details panel, operation log, A/B compare controls, restore preview, and gated recovery controls.

- Checkpoints provide known-good project states.
- Status and clean review show whether the current tree is changed.
- Compare shows differences between two checkpoints.
- Restore preview shows what would change before recovery.
- Recover is blocked until the user intentionally confirms the action.
- Database restore is also blocked until preview and typed confirmation are complete.

3. End user summary

For an end user, Source Vault is the safety screen for the Safire project. It answers four practical questions: what checkpoints exist, what changed, what can be restored, and whether the project is currently clean.

- Use Refresh to reload the checkpoint list.
- Use the Filter box to find a checkpoint by name, package, product area, or date marker.
- Use Restore Preview before any recovery.
- Do not press Recover unless you intend to return the project to the selected checkpoint.
- Create a database backup before database restore tests or risky data maintenance.
- When reporting a problem, send the operation log or support bundle rather than changing repository files manually.

4. Key concepts

Concept	Meaning
Repository root	The folder that contains the Safire project and Source Vault control files.
Checkpoint	A named, restorable project state.
Current tree	The files currently visible in the working folder.
Clean tree	A working tree with no uncheckpointed changes according to Source Vault.

Concept	Meaning
Dirty tree	A tree with changes that are not yet part of a checkpoint.
Restore preview	A dry-run report showing what recovery would do.
Recover	A live restore action that changes the working tree to a checkpoint.
Compare A/B	A difference report between two selected checkpoints.
Database backup	A restorable backup of Source Vault-managed database state or related product data.

5. Repository and working folder model

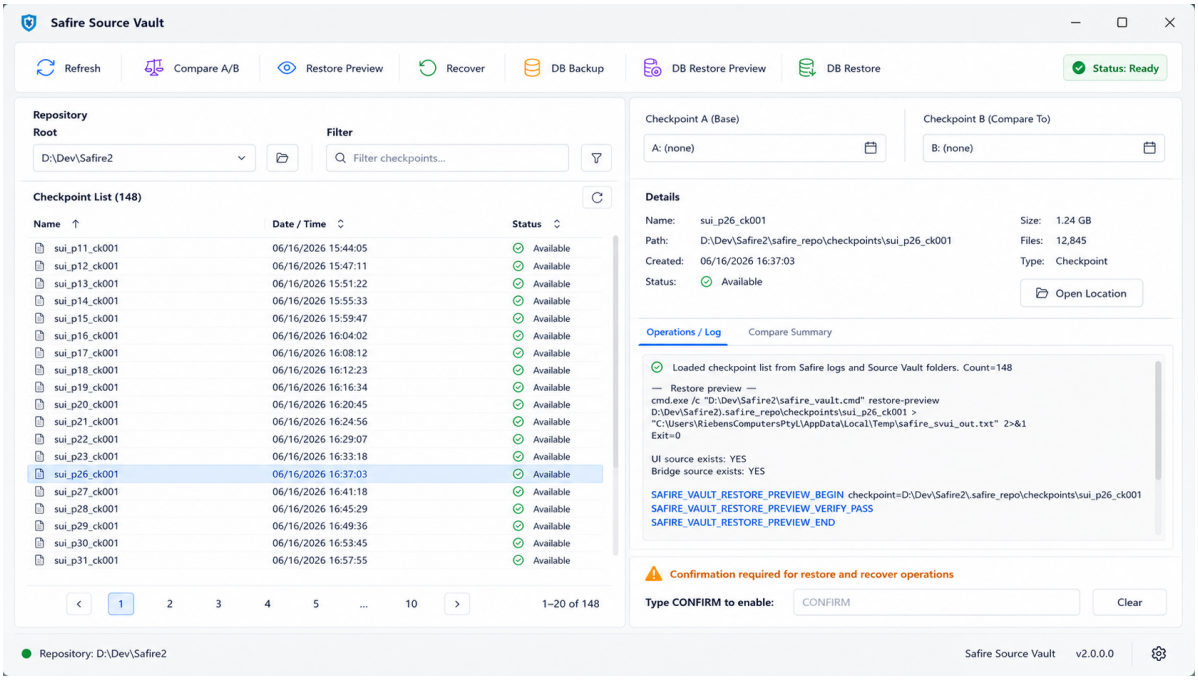
A Source Vault repository belongs to a project root. For a Safire developer preview installation, the root is normally selected in the application and passed to Source Vault commands.

Logical area	Purpose	User edits directly
Working tree	Current Safire source, project files, build files, docs, and generated product artifacts.	Yes, through normal tools.
Source Vault metadata	Checkpoint records, manifests, state, and recovery metadata.	No.
Build logs	Package, acceptance, diagnostic, and operation logs.	Read only.
Checkpoints	Named project snapshots.	No.
Database backups	Restorable data protection sets.	No, create through tools.
Support bundles	Diagnostic evidence for support.	Read and send as requested.

Users should not edit Source Vault metadata manually. All repository actions should go through Safire Source Vault or the supported command interface.

6. Source Vault application overview

Safire Source Vault Windows interface overview



The production interface presents one checkpoint table, A/B comparison selectors, restore preview, gated recovery, database backup/restore actions, and an operation log.

The Safire Source Vault window is organized around a modern layout: a toolbar, repository and filter section, one checkpoint table, a checkpoint details panel, operation tabs, and a confirmation area for dangerous actions.

Area	Purpose
Toolbar	Runs high-level actions such as Refresh, Compare A/B, Restore Preview, Recover, DB Backup, DB Restore Preview, and DB Restore.
Repository section	Shows the project root and filter controls.
Checkpoint list	Shows one checkpoint table with name, date/time, and status.
Checkpoint selectors	Stores Checkpoint A and Checkpoint B for comparison.
Details panel	Shows selected checkpoint metadata.
Operations / Log	Shows command output and status markers.
Confirmation area	Requires typed confirmation for recovery actions.

7. Opening Safire Source Vault

The Windows executable opens the Source Vault interface for a selected Safire project root. A normal launch passes the root as the first argument.

```
D:\Dev\Safire2\build\bin\SafireSourceVaultUi.exe D:\Dev\Safire2
```

If the application opens without a root argument, choose or type the repository root in the Root field and then press Refresh.

8. Selecting a repository root

The Root field identifies the Safire project to inspect. The root should contain the Source Vault command file and project structure. The folder button is used to browse to another repository when supported by the application shell.

- Use the exact project root, not a subfolder.
- Confirm the status pill changes to Ready after refresh.
- If the checkpoint list is empty, confirm that the root is correct.
- Do not select a backup folder as the root.

9. Checkpoint list and filtering

The checkpoint table is the main navigation surface. It replaces duplicated list boxes and shows checkpoint name, date/time, and availability. The filter box narrows the list without changing the repository.

Column	Meaning
Name	Checkpoint name or checkpoint folder identifier.
Date / Time	Creation or evidence time shown when available.
Status	Availability and verification status.

A selected row drives the Details panel and can be assigned as Checkpoint A or Checkpoint B for comparison.

10. Checkpoint details

The Details panel shows the selected checkpoint metadata. Typical fields include name, path, created date, status, file count, size, type, and an Open Location action.

- Use Details to confirm you selected the intended checkpoint.
- Use Open Location for inspection only; do not manually edit checkpoint contents.
- If the status is not Available, run Verify or review the operation log.

11. Checkpoint A and B selection

Source Vault uses two selected checkpoints for compare operations. Checkpoint A is the base. Checkpoint B is the checkpoint being compared to the base.

Selector	Use
Checkpoint A (Base)	The older or trusted checkpoint.
Checkpoint B (Compare To)	The checkpoint to compare against A.

The UI should keep A and B visible so the user does not accidentally compare or recover the wrong checkpoint.

12. Comparing checkpoints

Compare A/B runs a Source Vault comparison between the selected checkpoints and writes the result to the operation log. It does not change the working tree.

```
safire_vault.cmd compare sv103_svui_v1_001 sv104_svui_run_v1_001
```

- Set Checkpoint A and Checkpoint B first.
- Run compare before recovery when you need to understand what changed.
- Use the Compare Summary tab to show a shorter result when available.

13. Restore preview

Restore preview is the safe dry-run before recovery. It shows what Source Vault would restore without changing the live tree.

```
safire_vault.cmd restore-preview sv104_svui_run_v1_001
```

A preview result should clearly show begin, verification, warnings, and end markers. If preview fails, do not recover. Review the output or create a support bundle.

14. Recovering a checkpoint

Recover is a live operation. It changes the current working tree to the selected checkpoint. For safety, the UI should require restore preview first and a typed confirmation such as RECOVER before enabling the action.

- Select the checkpoint.
- Run Restore Preview.
- Read the preview output.
- Type RECOVER only when you intend to proceed.
- Run Status and Clean Review after recovery.

Recovery should never be hidden behind an ordinary click. It must be intentional and visible in the log.

15. Database backup

Database backup creates a restorable backup set for product data or project-related database state that Source Vault tools manage. Use backup before risky database maintenance, restore testing, migrations, or bulk changes.

```
safire_vault.cmd db-backup
```

Good backup evidence	Meaning
Backup created	A backup set was written.
Manifest present	The backup can be identified.
Verify result	The backup can be inspected before restore.
Location shown	The user knows where the backup was stored.

16. Database restore preview

Database restore preview reports what database restore would do before live data is touched. It is the database equivalent of Source Vault restore preview.

```
safire_vault.cmd db-restore-preview
```

- Preview should show the source backup identity.
- Preview should show the target database or data set.

- Preview should show compatibility warnings.
- Preview should not change live data.

17. Database restore

Database restore is a live data operation. It must be gated by database restore preview and typed confirmation such as RESTOREDB. A normal user should not be able to restore a database by accidental click.

- Run DB Restore Preview first.
- Confirm the selected backup is correct.
- Type RESTOREDB only when the restore is intentional.
- Run verify or health check after restore.

18. Status and clean review

Status and clean review show whether the repository has uncheckpointed changes. A dirty state is expected while a developer is working. A clean state is expected immediately after a successful checkpoint.

Result	Meaning	Typical action
Clean	No uncheckpointed changes detected.	Safe to proceed or distribute evidence.
Dirty	Changes exist after the last checkpoint.	Review changes or create checkpoint after acceptance.
Error	Source Vault could not inspect the tree.	Check root, permissions, logs, and support bundle.

19. Using Source Vault during Safire development

In Safire development, Source Vault acts as the product-wide rollback and checkpoint mechanism. A normal development slice changes source, generated product files, documentation, tests, and acceptance logs. After the slice passes, a checkpoint captures the whole result.

- Start from a known checkpoint.
- Apply the implementation package.
- Run the focused acceptance.
- Review result markers.
- Ask for explicit checkpoint approval.
- Create and verify the checkpoint.
- Confirm the tree is clean.

20. When to create a checkpoint

Create a checkpoint after a meaningful accepted state, not after every small edit. Good checkpoint moments are successful feature completion, hotfix acceptance, release pack creation, migration completion, tool installation, or final verification after a recovery repair.

Do not create a checkpoint for a failed package unless support intentionally needs a diagnostic checkpoint. The normal policy is to checkpoint only passing work.

21. Checkpoint naming guidance

Checkpoint names should identify sequence, product area, and reason. Names should be short enough for Windows paths and clear enough for recovery decisions.

Pattern	Example	Use
svNNN_area_vN_001	sv104_svui_run_v1_001	Normal product checkpoint.
area_ck001	dbforms_pack_v1_ck001	Older compact package checkpoint style.
hotfix_area_001	svui_run_hf001_001	Only when the hotfix itself is the stable state.

22. Before risky work

Before broad changes, risky refactors, recovery operations, database restore, or migration work, review the current repository state.

- Run Status.
- Run Clean Review if available.
- Confirm the current checkpoint name.
- Create a restore preview before any live restore.
- Create database backup before database restore or destructive data work.
- Do not continue if the root is unclear.

23. After a successful package

After a package passes, Source Vault captures the accepted state only after explicit approval. A checkpoint script should verify the package acceptance markers, create or verify the new checkpoint, run final status, run clean review, and write a compact acceptance file.

```
Expected final evidence: CurrentCheckpoint=sv104_svui_run_v1_001 SourceVaultStateAfterCheckpoint=CLEAN
Result=PASS
```

24. Logs and acceptance evidence

Source Vault and Safire package work rely on logs and acceptance files. These files are not merely console output; they are structured evidence used for review and support.

Evidence	Purpose
Acceptance file	Records pass/fail markers for a package, hotfix, or checkpoint.
Operation log	Shows command output from UI actions.
Compile log	Shows build failures or warnings for generated executables.
Support bundle	Collects relevant logs and state for review.

25. Working tree states

A working tree can be clean, dirty, partially repaired, or unknown. Source Vault commands should make the state explicit rather than relying on assumptions.

State	Description	Safe next step
Clean	Matches a known checkpoint or has no uncheckpointed changes.	Continue or start new work.
Dirty expected	Work completed but not yet checkpointed.	Checkpoint after acceptance.
Dirty unexpected	Unknown changes exist.	Review before checkpoint or restore.
Unknown	Status command failed.	Investigate root, permissions, command log.

26. Restore and recovery policy

Recovery policy is strict because restore changes live files. The supported policy is restore preview first, then explicit confirmation, then live restore, then verification.

- Never run live restore without preview.
- Never hide restore behind an ordinary button click.
- Do not restore a production or active working tree unless the user intends it.
- Record all restore actions in the operation log.
- Verify after restore.

27. Backup policy

Backups complement checkpoints. A checkpoint protects project state. A database backup protects data state. When a development operation affects both source and data, use both mechanisms.

Need	Use
Recover source/project files	Source Vault checkpoint and restore preview.
Recover database content	Database backup and restore preview.
Prove accepted development state	Source Vault checkpoint.
Support data issue	Diagnostic bundle plus backup if possible.

28. Source Vault UI action matrix

UI action	Real operation	Changes live state	Safety gate
Refresh	Reload repository and checkpoints.	No	None.
Status	Run Source Vault status.	No	None.
Compare A/B	Compare selected checkpoints.	No	A and B selected.

UI action	Real operation	Changes live state	Safety gate
Restore Preview	Dry-run recovery.	No	Checkpoint selected.
Recover	Live restore selected checkpoint.	Yes	Preview first and type RECOVER.
DB Backup	Create database backup.	Creates backup only	Backup target available.
DB Restore Preview	Dry-run database restore.	No	Backup selected.
DB Restore	Live database restore.	Yes	Preview first and type RESTOREDB.

29. Command line examples

The UI calls the same product operations exposed through Source Vault commands. The exact command set may grow, but the safety model remains stable.

```
safire_vault.cmd status safire_vault.cmd clean-review safire_vault.cmd checkpoint sv104_svui_run_v1_001
safire_vault.cmd verify sv104_svui_run_v1_001 safire_vault.cmd compare sv103_svui_v1_001
sv104_svui_run_v1_001 safire_vault.cmd restore-preview sv104_svui_run_v1_001 safire_vault.cmd db-backup
safire_vault.cmd db-restore-preview
```

30. Troubleshooting guide

Symptom	Likely area	Recommended action
Checkpoint list empty	Wrong root or missing Source Vault command.	Select the correct root and Refresh.
Restore preview fails	Checkpoint missing or incompatible.	Verify the checkpoint and review log.
Recover button blocked	Preview or typed confirmation missing.	Run Restore Preview and type RECOVER only if intended.
DB Restore blocked	Database preview or typed confirmation missing.	Run DB Restore Preview and type RESTOREDB only if intended.
Status shows dirty after package	Accepted changes not yet checkpointed.	Checkpoint after explicit approval.
Status dirty after checkpoint	Checkpoint did not complete or files changed afterward.	Run clean review and inspect acceptance.
Compare has no output	A or B not selected.	Set both checkpoints.

31. Administrator checklist

- Confirm the Source Vault root.
- Confirm the executable opens the expected project.
- Confirm Refresh shows checkpoints.
- Run Status and read the result.
- Run Restore Preview on a non-critical checkpoint to verify the safety path.
- Confirm Recover requires typed RECOVER.
- Confirm DB Restore requires typed RESTOREDB.

- Confirm logs can be copied for support.
- Confirm the current checkpoint after accepted work.

32. Developer checklist

- Start development from the latest confirmed checkpoint.
- Keep package outputs in predictable log folders.
- Use focused acceptance rather than broad reruns unless requested.
- Do not checkpoint failed work.
- Ask for explicit approval before checkpointing.
- After checkpoint, verify and prove clean state.
- Use restore preview before any recovery.
- Document feature-level checkpoint names clearly.

33. Glossary

Term	Definition
Acceptance file	Structured result file showing pass/fail markers.
Checkpoint	Named restorable Source Vault state.
Clean review	Inspection proving no unexpected uncheckpointed changes.
Compare	Difference report between two checkpoints.
Database restore preview	Dry-run report for database restore.
Dirty tree	Working tree with changes after last checkpoint.
Recover	Live restore of a checkpoint.
Repository root	Top folder of a Safire project protected by Source Vault.
Restore preview	Dry-run report before recovery.
Safire Source Vault	Windows interface and product surface for Source Vault operations.