Vault

Import Tool Guide

SourceGear Corporation

Contents

Vault Overview

SourceGear Vault is a compelling replacement for Microsoft Visual SourceSafe.

Reliable

Vault's design places a strong emphasis on the integrity of the repository. The Vault repository is stored using *Microsoft SQL Server 2000*. Furthermore, Vault supports atomic check in transactions (change sets). When checking in a collection of files, the entire check in either succeeds or fails as an atomic unit.

Seamless transition from SourceSafe

An extremely simple to use, utterly painless and bulletproof tool is provided to convert SourceSafe databases to Vault repositories. It converts any SourceSafe database to a Vault repository, including all history information, with no loss of information.

Familiar features and interface

SourceSafe users will feel comfortable using Vault. The client user interface very closely resembles that of SourceSafe and SourceOffSite – all major features of SourceSafe are present.

.NET Architecture

Vault is built entirely on Microsoft's .NET platform, including C#, IIS and SQL Server 2000.

Open

The Vault server is built as a collection of .NET Web Services. The API provided by these services is fully documented and openly disclosed. Using the API documentation and commonly available tools, the Vault architecture allows clients to be created for any platform.

International

SourceSafe was created years before Unicode became the prevalent technology for i18n. Vault's design is thoroughly Unicode, largely because of its use of the .NET Framework. Vault is also localization-ready.

Concurrent Development

Vault's support for branching, automatic merging and multiple checkouts is superior to that of SourceSafe, even as the interface and paradigm are familiar to SourceSafe users.

History Explorer

The use of SQL as a repository storage allows Vault to easily provide more advanced ways of querying the history of a project.

SourceSafe Import Tool Overview

Goals

To import your SourceSafe database accurately in its *current state*.

By *current state* we mean that items that are shared will be shared and items that are pinned will still be pinned. And most importantly, all revisions of non-deleted files will be imported. (Keep in mind that a file shared to a project that is not being imported will not be shared after import.)

Background

The import tool uses the SourceSafe Automation Component API to query the SourceSafe database history for file and project information. This information is used to build a list of all the events that took place over time.

What is wrong with the SourceSafe Automation Component's API that makes this so difficult to do a 100% import?

Due to several inadequacies in the SourceSafe Automation Component API, a few compromises have been made. The decision was made that the current state of most customers' databases was more important than the actual history. To see how this may affect you, read the section below called "How will this affect me?".

The API is very limited in several areas, described below.

- There are several events that do not report the item that is being modified.

  Added, Deleted, Destroyed, Recovered - A parent project receives this event. The event does not contain the name of the item that was added, deleted, destroyed, or recovered.

- History may be re-written.

  Moved, Renamed – History is re-written to use the new name instead of the original name for events before the rename/move.

- Sometimes history is just wrong.

  Share a pinned file. The file in the new location is also pinned. However, the action describing the pin (in the new location) says that the old file was pinned. Unpinning this file will show the correct filename.

- Sometimes history is missing.

  We have a database that shows two files as being shared. Looking at properties shows them as shared. However, looking through history using Visual SourceSafe Explorer or through the API there is no evidence that the files were ever shared. We can tell that they are shared, but do not know when it happened.

How will this affect me?

You will not have history for several events. Specifically, the following:

- Branch: An item that has been branched will show up as if it had never been shared.

  Likewise, You will be missing history for “Share” under these circumstances.

- Shares: Items that are currently shared will be shared in your new Vault Repository. However, the time of the share will be just after the first item in the share (alphabetically) was created.

- Deleted, Destroyed: If it doesn’t currently exist in your database then it will not show up in your Vault Repository. No import of a deleted item occurs. No information about Destroyed items is available so they cannot be imported.

  Related to this are Recovered items. The item will be imported but history will not indicate that it was ever deleted and therefore never recovered.

- Move, Rename: Moves and Renames are not done. The item will be imported as if it had always existed at its current location.

- Pinned, Unpinned: If the item is currently pinned it will be pinned in your new Vault Repository. Multiple pin and unpin of an item is not imported. Pin will only appear if it is pinned at time of import.

  Also, due to version numbering differences between SourceSafe and Vault, every attempt is made to ensure that the correct version of the file is pinned.

Conclusion

We have made every attempt to accurately import your SourceSafe database. We realize that our current solution may not work for everyone and continue to try to resolve the issues mentioned above.

Please let us know of any problems you encounter while trying to do an import. We are also interested in any ideas you may have for improving the tool.

Thanks!

The Vault Team

Tips for SourceSafe Users

Vault has been designed to make a seamless transition from SourceSafe. However, many improvements have been made. This has resulted in the following differences.

History Explorer

The **History** dialog is replaced by a top-level, non-modal window which allows browsing the repository history through queries. also does not include a destructive rollback function except in the **Admin Tool**.

Branch exists separately from Share

In *SourceSafe*, **Branch** can only be performed after a **Share** operation, but *Vault* allows a **Branch** operation to happen by itself. The functionality is equivalent.

Atomic Check Ins and Check Outs

Add and **Delete** operations can be scheduled to happen as part of the next check in. Options exist that allow you to specify whether the operations happen immediately, or whether they are included with the next transaction.

SQL Server database

Vault's repository is stored in SQL Server rather than a collection of files in an undocumented format.

Request Exclusive Lock

In lieu of **Allow Multiple Checkouts**, *Vault* has **Request Exclusive Lock**. The sense of the option is reversed and the default is to not request exclusive locks on files which are considered to be mergable.

Terms

We have changed the following terms:

- In *Vault*, "Repository" *(SourceSafe -- "Database")* refers to the place where source control files and history are stored.
- In *Vault*, "Folder" *(SourceSafe -- "Project")* refers to folders in the repository

Menu Items

The following menu items are no longer used:

- The **File** / **Create** Shortcut
- The **Edit** / **Select**
- The **Web** / **Check Hyperlinks**
- The **Web** / **Create Sitemap**
- Reports (In SourceSafe, these features serve the same purpose as **Print** commands.)

Show Differences Feature

- *Vault* only supports the “visual” diff, not “SourceSafe”, not “UNIX”.
- *Vault* does not include the option to **Ignore White Space** *(planned for future release).*
- *Vault* does not include the option to **Ignore Case** *(planned for future release).*
- *Vault* does not include the option to **Ignore OS Differences** because *Vault*’s diff code already ignores OS differences.

Miscellaneous Differences

- The **Options** dialog has no pane for **Command Line Options**.
- **Check In** and **Check Out** only function from and to the working folder.
- *Vault* does not always require files to be explicitly checked out before editing and checking in. *Vault* may be configured to create working copies without the read-only bit set.
- The **Delete** command does not include **Destroy Permanently**. This functionality is present in the **Admin Tool** where it is called **Obliterate**.
- The **Label** command functions rather differently in its presentation although the concept and application is the same. A label is presented to you as a **Virtual Folder** and can be viewed in the folder tree based on an option setting.
- Vault does not have a built-in text editor.
- There is no option to force MS-DOS compatible filenames.
- The **Store Only Latest Version** feature is not supported. In SourceSafe, it appeared in the **File Properties** dialog box.
- The **Get Latest Version** command is used to retrieve changes from the repository into the working copies, including any merge or conflict resolution. A file may not be checked in to the repository unless all changes from the server have been merged into the working copy using the **Get Latest Version** command.
- *Vault* makes use of a hidden state folder (named *_sgvault*) located in the local file system. These state folders keep track of the current version of files downloaded by the client and also store baseline versions of files for later merges. By default the state folders are kept in a user's application data folder, but a general option can be set to keep the state folders in the current working folders.
- All access to the *Vault* repository occurs through the *Vault Server*. There is no “local-mode client” to manipulate the repository through the file system.

Hardware Requirements

Both server and client requires Windows 2000 (SP3) and above for OS.

The .Net Framework is required. For information on the system requirements and installation, please refer to Microsoft.

Server Hardware Requirements

- Server class machine with 512MB RAM.
- Disk space requirements dependent on size of source code tree.

Client Hardware Requirements

- Client machine capable of running .Net apps.

Software Requirements

Both server and client requires Windows 2000 (SP3) and above for OS.

The .Net Framework is required. For information on the system requirements and installation, please refer to Microsoft.

Server Software Requirements

- SQL Server 2000 and above; OR, MSDE 2000 and above
- IIS 5 or above with ASP.Net
- .Net Framework 1.0

Client Software Requirements

- .Net Framework 1.0

SourceSafe Import Tool Wizard

The SourceSafe Import tool is used to import information from your SourceSafe database into your Vault repository. Installation of this tool appears as a Wizard.

SourceSafe Database

Choose the SourceSafe Database to import. You may choose from the list of known databases, type in your own, drag and drop a srcsafe.ini file from Explorer, or browse to a database.

The database must be on a mounted file system and SourceSafe must be installed and available. It is recommended that the database be on the same computer as the Import Tool.

Vault Server

Choose the Vault server to import the SourceSafe data into. The server must already be installed. If you have not yet created a Vault server installation, please see the Server Installation in the Admin Guide. You will be prompted for the server hostname and administrator password.

Repository Information

If your SourceSafe database contains multiple “projects” which are unrelated, we recommend that you place each one in its own Vault repository. If you have a group of projects that are logically related to each other, it is better to place them in a repository together, enabling them to share and branch as a group. Note that you cannot share files across Vault repositories.

Select Project

Select the project to import into the repository. By default, the entire database is selected and the utility will import everything (excluding deleted items).

Select the folder to be imported.

Pre-Scan

Begin pre-scanning the SourceSafe database.

This page will tell you if the SourceSafe database is locked and will display all users that are currently logged in, including the user logged in through the import tool.

If the SourceSafe database is not locked, it is recommended that you do so now with the *Visual SourceSafe Admin* tool. Ensure that all users are logged out so that the contents of the database are not changing during the import.

This step may take several minutes but your data is not yet being imported.

Pre-scan in Progress

The tool is now scanning the SourceSafe database for files, revisions, users, labels, etc. It will scan all the selected folders in the SourceSafe database, count the files, count the revisions, build a list of all the users, and record project security permissions (if enabled).

Pre-scan Complete

The results of the pre-scan are displayed. This includes

- The number of files and projects to be imported
- The estimate of the number of hours for completion

Users

A list of all SourceSafe users is displayed. Another list of all Vault users is also displayed. Match the SourceSafe user to the Vault user. Create another Vault user if needed.

Select **Create new accounts for each SourceSafe user as needed** if necessary. A new account for each SourceSafe user that was not mapped to a Vault user will be created in a disabled state. No one will be able to login to such an account, and as such, the Vault server will not require a valid license for that user.

The Import Tool will not try to map SourceSafe users to Vault users with the same username unless it created the user. There is no way to guarantee that they are the same.

Checkouts

Any files in the SourceSafe database that are currently checked-out will not be checked-out after they have been imported to the Vault repository.

A list of the checked-out files were encountered during the pre-scan will be displayed. You may wish to notify these users of the impending transition to Vault now, to give them the opportunity to Check In any pending changes in their working folders.

Vault Repository

Choose or create a Vault repository. A single Vault server can maintain any number of virtual repositories, all of which exist within the same SQL Server installation.

Select to create a new repository on the specified Vault server or to import data into an existing repository on that server.

If you select to create a new repository, you will be prompted for a name and an optional folder into which all the new folders will be placed. If no folder is specified the root will be used.

If you select to import the data into an existing repository, select the repository and then folder into which all the new folders will be placed.

Project Security

If project security is enabled in your SourceSafe database you will be asked if you want to enable it for your Vault repository.

Ready to Import

The SourceSafe Import Tool is ready to begin importing data. Throughout the import process, the progress and estimated time to completion will be displayed.

No further questions will be asked, so you need not monitor the import as it proceeds.

Import in Progress

As the tool is importing SourceSafe database into your Vault repository, a progress meter will be displayed estimating the time to completion. The file/folder is currently being imported will also be displayed.

Import Complete

The import is complete and the final report is available. If there were any errors, they will be listed in the report. You can copy this report to the clipboard for further review. You may also review the full log file that is generated in the applications folder. Log files are named using the date and time the import began.

The SourceSafe Import Tool has completed.