Installation and Files

SmartGit stores its settings files per-user. Each major SmartGit version has its own default settings directory, so you can use multiple major versions independent of each other. The location of the settings directory depends on the operating system.

Default Location of SmartGit's Settings Directory

- **Windows** `%APPDATA%\syntevo\SmartGit\<major-smartgit-version>` (`%APPDATA%` is the path defined in the environment variable APPDATA)
- **Mac OS** `~/Library/Preferences/SmartGit/<major-smartgit-version>`
- **Linux/Unix** `~/.smartgit/<major-smartgit-version>`

> **Tip**
>
> You can change the directory where the settings files are stored by changing the property smartgit.settings. This is used by the portable bundle for Windows.

Notable Files in the Settings Directory

- `license` stores your SmartGit *license key*.
- `logs/*` contains debug log information, for which `logs/log.txt.0` contains the most recent logging. It can be configured via `logger.properties`. You may remove this file: afterwards, SmartGit will return to its default logging settings.
- `passwords` is an encrypted file and stores the *passwords* used throughout SmartGit. You may remove this file: afterwards, all passwords are lost.
- `accelerators.xml` stores the *accelerators* configuration. You may remove this file: afterwards, all accelerators will be reset to their defaults.
- `credentials.xml` stores authentication information (not including the corresponding passwords). You probably do not want to remove this file: afterwards, all credentials (user names, private keys, certificates) will be lost.
- `hostingProviders.xml` stores information about configured hosting provider accounts (not including the corresponding passwords). You probably do not want to remove this file: afterwards, all connect details for all hosting provides will be lost.
- `notifications.xml` stores information about the state of notifications which show up in the status bar in various cases. You may remove this file: afterwards, various notifications may show up again.
- `projects.xml` stored all former *SmartGit projects* (up to SmartGit 5) including their settings. Beside that it contains all repository root specific settings.
- `repositories.xml` stores the information about known repositories, their names and repository groups.
- `repository-cache.xml` stores all cached information about repository states, e.g. what local branch is checked out, whether there are incoming or outgoing changes.
- `settings.xml` stores the application-wide settings (e.g. the preferences) of SmartGit. You should not remove this file, unless you want to completely reset SmartGit.
- `tools.xml` stores *external* tools which have been configured in the Preferences. You probably do not want to remove this file: afterwards, all you external tools configurations will be lost.
- `ui-config.xml` stores UI related, more stable settings, e.g. the toolbar configurations. You may remove this file: afterwards, various aspects of the UI will be reset to defaults.
- `ui-settings.xml` stores UI related, volatile settings, e.g. window sizes or column widths. You may remove this file: afterwards, various aspects of the UI will be reset to defaults.

Resetting certain parts of the configuration to defaults

To reset certain parts of SmartGit's configuration ("settings") to the defaults:

1. locate the appropriate configuration file (`*.xml`)
2. Exit SmartGit, using **Repository|Exit**
3. Get rid of the file(s)
4. Start SmartGit again

Program Updates

SmartGit stores program updates which have been downloaded automatically through SmartGit itself by default in the subdirectory `updates` of the *Settings* root directory (see Default Location of SmartGit's Settings Directory). This allows "light weight", *patch-like* updates which do not require write access to the actual SmartGit installation directory. As a consequence, your SmartGit installation directory is usually not up-to-date, but it will launch the downloaded updates from the `updates` directory. Only under specific conditions, SmartGit will detect that an upgrade of the installation directory itself is necessary ("installation update").

Tip

You can manually trigger the update of the installation directory from the **About** dialog, section **Information**, **...**-button right beside **Version**.

If you prefer to keep your SmartGit installation *always* up-to-date, you can select **Update SmartGit application in place** in the Preferences, section **SmartGit Updates**. Note, that updating with this option selected may require administrator privileges.

Technical Details

The root directory of the *Updates* directory contains sub-directories for every major version. Such a *major version* directory contains a `control` file for the latest downloaded build and a `current`-file which points to the currently used build. Usually, this will be the highest build which shows up in this directory. The `control`-file only *configures* which binaries are part of the build by linking to the actual binaries which are stored in the `repo`-subdirectory and which are shared among all builds.

Each new build has a corresponding, digitally-signed control file which contains information about all required application files with their download location and the expected file content hash. To reduce band-width, application files only will be downloaded if they are not yet locally available. After download, the content will be verified with the hash from the control file.

When starting SmartGit, the `bootloader.jar` from the installation directory is launched. This uses the `control` file from the *Updates* directory to determine which updated SmartGit files to launch that contain the actual application code.

Warning

By modifying the `control` file or any other contents within the *Updates* directory, you may easily screw up your SmartGit installation. Hence, do not touch these files unless you have good reasons to do so.

JRE Search Order (Windows)

On Windows, the `smartgit.exe` launcher will search for a suitable JRE in the following order (from top to bottom):

- Environment variable SMARTGIT_JAVA_HOME
- Subdirectory `jre` within SmartGit's installation directory
- Environment variable JAVA_HOME
- Environment variable JDK_HOME
- Registry key HKEY_LOCAL_MACHINE\SOFTWARE\JavaSoft\Java Runtime Environment

Installing/running multiple SmartGit versions in parallel

You can install multiple versions of SmartGit in parallel and you can even run them at the same time. This will be useful if you want to primarily work with the *Preview* version, but have the *latest released* version still present as *fall back*. SmartGit has separate settings areas for different versions, so it's only an issue of "installation" :

- On **Linux**, simply unpack every SmartGit bundle you want to use to a different directory
- On **OS X**, simply unpack every SmartGit DMG you want to use to a different directory
- On **Windows**,
    - either use **only portables bundles** for every SmartGit version you want to work with and unpack them to different directories
    - use **exactly one installer bundle** for the primary SmartGit version you want to work with and **additional portable bundles** for the other version(s)