System Properties

> Starting with version 18.1, many system properties can be edited directly in the **Preferences**. These properties won't be mentioned here anymore. For a list of system properties for version 17.1, have a look at System Properties.

The vast majority of SmartGit's system properties can be configured by editing the file `smartgit.properties` in the settings directory.

> **Note**
>
> The file `smartgit.properties` contains only settings for SmartGit itself. If you want to configure your Git repositories, have a look at the various Git configuration files instead, such as `.git/config` for the configuration of individual Git repositories, and `~\.gitconfig` (in your HOME directory) for global configuration options.

First, open the settings directory. Its default location is described in Default Location of SmartGit's Settings Directory. In the settings directory, you will find the `smartgit.properties` file. Open it with a text editor, such as Windows Notepad.

Each of the settings in `smartgit.properties` is specified on a separate line, according to the following syntax: `key=value`. If a line starts with #, the entire line is treated as a comment and ignored by the program.

The following list shows the available system property keys.

Interacting with Mercurial

smartgit.hg.executable.maxVersion

Use this system property to change the maximum allowed Mercurial version SmartGit will be able to work with.

> In case of Mercurial API changes, this might break SmartGit's Mercurial integration.

Background: SmartGit is accessing internal Mercurial API which may change for newer versions of Mercurial, for example when switching from version 3.7 to version 3.8 (this actually happened already multiple times in the past, breaking SmartGit's integration). Thus, SmartGit is limited to a certain version of Mercurial. We are updating this "dependency" with newer (major) SmartGit versions. If you want to use a very recent Mercurial version before SmartGit officially supports it, you may do so by changing this property. **We always appreciate input on known working configurations and on problems with new Mecurial versions at** `smartgit@syntevo.com`.

Example

To allow SmartGit to work with any major Mercurial version (up to `99.9`) , set following system property:

```
smartgit.hg.executable.maxVersion=99.9.99
```

Networking

java.net.preferIPv4Stack

By default, SmartGit prefers to connect via IPv4. To connect via IPv6 instead, set this option to `false`.

http.nonProxyHosts

Use these properties to specify servers to connect directly to, bypassing the configured proxy, for example: `*.foo.com|localhost`. Note, that only internal code of SmartGit is honoring `http.nonProxyHosts`. *This does not include Git itself.*

Company-wide configuration

smartgit.setupFinishedUrl

This setting specifies the URL to open after SmartGit has been started for the first time and the setup wizard was completed.

smartgit.preferences.<category>.visible

You can use this system property to hide certain **Preferences** pages. Available categories are:

- `executables`
- `externalTools`
- `compareTools`
- `conflictSolver`
- `spellCheck`
- `proxy`
- `updateCheck`
- `bugReports`

To hide a specific page, set the corresponding property to `false`.

Example

To hide the **Tools** page, set:

```
smartgit.preferences.externalTools.visible=false
```

smartgit.contactSupportEnabled

Set this option to `false` to hide the menu item **Help|Contact Support**.

smartgit.registerEnabled

Set this option to `false` to hide the menu item **Help|Register**.

Update Check

smartgit.updateCheck.enabled

Set to `false` to disable the automatic checking and disallow the manual checking for new program versions by hiding the corresponding menu items **Help|Check for New Version** and **Help|Check for Latest Build**. You should only turn this check off for network installations where SmartGit users may not be able to perform the update themselves. When settings this option, you will probably also want to hide the corresponding page from the **Preferences**, using smartgit.preferences.updateCheck.visible.

Note that this will also disable notifications of new bugfix releases which you can upgrade to for free and which improve the stability or reliability of SmartGit.

If you just want to switch off automatic checking, use `smartgit.updateCheck.automatic=false` instead.

smartgit.updateCheck.automatic

Set to `false` to disable the **automatic** check for new versions on a global level which can be convenient e.g. for network installations. To disable the check for an individual installation/user, better do that in the **Preferences**, section **SmartGit Updates**.

smartgit.updateCheck.alwaysUpgradeToLatestBuild

Set to `true` to make SmartGit check for the availability of a new *latest build* on start up. *Latest Builds* are the "bleeding edge" builds between subsequent (minor) *release* builds, like between version 8.0.1 and 8.0.2 or 8.1 preview 3 and 8.1 preview 4. They will contain the latest

improvements and bugfixes. Usually we will ask you to manually fetch the latest build using **Help|Check for Latest Builds**.

smartgit.updateCheck.checkForLatestBuildVisible

Set to `false` to hide **Help|Check for Latest Build**.

smartgit.updater.directory

Use this property to customize the program updater's temporary directory, which is by default located in your home directory/profile. This should only be necessary if updating is not possible due to (file system) restrictions in this default directory, e.g. if execution of files is prevented by the system. On Windows, paths have to be specified using forward-slashes, like `c:/temp`.

Bug Reporting

smartgit.disableBugReporting

Set to `true` to disable sending of crash footprints (even if configured in the **Preferences**) and skip the option to send bug reports to us. When setting this option, you will probably also want to hide the corresponding page from the **Preferences**, see smartgit.preferences.bugReports.visible .

> When using this option, be sure to provide an alternative way for your users to report SmartGit bugs to you, otherwise they will go unnoticed.

License user/seat tracking

For licenses with a large number of users, it can be helpful to track the number of active SmartGit users over time. For this purpose, SmartGit can optionally access ("ping") a configurable URL which can be used to collect these user statistics on the server side. This requires customer-specific configuration because in different companies different means define a person (user), e.g. by a user-specific unique environment variable value. A simple implementation on the server-side would just be a virtual host logging to a separate access log file. The resulting log files can be analyzed, e.g. using `grep`.

smartgit.license.defaultPath

By default, SmartGit will look for a "default" license file the installation default directory. You can use this system property to specify a different **file system path** for the default license to look for.

Example
To have SmartGit take the default license from `\\license-server\smartgit\license`, set: `smartgit.license.defaultPath=\\\\license-server\\smartgit\\license`

smartgit.license.alwaysCheckForNewerDefaultLicense

By default, SmartGit will only look for a default license, if there is **no** or **no valid** existing license. Sometimes, it may be desirable to replace even **valid** licenses by newer default licenses. To do so, set:

Example
`smartgit.license.alwaysCheckForNewerDefaultLicense=true`

smartgit.license.serverPing.url

Enables the license user tracking feature and specifies the URL (template) which SmartGit will connect to on startup and once per day. The template is basically a URL for which following keywords will be replaced:

- `${email}` will be replaced by the user's email found in the `~/.gitconfig` file.
- `${smartgit.version}` will be replaced by SmartGit's version followed by a # and the build number, e.g. `17.1.1#11176.`

- `${smartgit.license}` will be replaced by a unique string of the registered SmartGit license file.
- `${smartgit.license.x}` will be replaced by the value of the corresponding (case-sensitive!) key from the registered SmartGit license file, e.g. `${smartgit.license.SupportUntil}`.
- `${prop.x}` will be replaced by the Java system property `x`. For instance, `${prop.user.name}` will be replaced by the Java system property `user.name`.
- `${env.X}` will be replaced by the environment variable `X`. For instance, `${env.USERID}` will be replaced by the environment variable `USERID`.

smartgit.license.serverPing.method

Specifies the HTTP request method to be used, either `GET` or `HTTP` (defaults to `GET`).

Example

Following configuration will track users on (including a custom `USERID` environment variable) at `syntevo.com`.

```
smartgit.license.serverPing.method=get
smartgit.license.serverPing.url=http://www.syntevo.com/smartgit/server-ping?email=${email}&name=${prop.user.name}&id=${env.USERID}
```

Debug Properties

log4j.[category]

Use this property to enable debug logging for certain SmartGit modules; `[category]` has to be replaced by the appropriate module identifier.

Example

To enable debug logging for the Refreshing modules, set following properties:

```
log4j.smartgit.refresh=DEBUG
log4j.sc.vcs.model.refresh=DEBUG
```