Journal View

The Journal shows a compact history (Log) for the current branch:

- *orange* commits denote *ahead* commits which are just present on your *local* branch but not yet on the *remote* branch. These commits will be sent to the remote branch/repository for the next Push.
- *green* commits denote *incoming* commits which are only present in the *remote* branch and will be integrated into your *local* branch for the next Pull.
- *black* commits with *green* connectors denote commits which are currently present in the *remote* branch only, but have been present in your *local* branch at an earlier time. Usually, these are commits which have been *rewritten* using Rebase. To show such commits, invoke **Show Rewritten Commits** from the options menu.
- *black* commits with *blue* connectors denote commits from the *auxiliary* branch which can be toggled using **Show Auxiliary Branch** from the options menu.
- *black* commits with *gray* connectors denote commits which are common to more than one of the *local*, *remote* or *auxiliary* branch.

Depending on the type of commit(s) selected, you can invoke various operations from the context menu, most notably, you can easily rewrite the history:

- To squash adjacent commits, select them and invoke **Squash Commits** and provide the new commit message.
- To reorder commits, just use drag and drop.
- To coalesce two (not necessarily adjacent) commits *with the same commit message*, drag one of the commits onto the other one.
- To change the commit message, select the commit and invoke **Edit Commit Message**.
- To change the author, select the commit and invoke **Edit Author**.

The number of commits displayed per category is limited, so the graph will stay neatly arranged even if there are lots of commits per category (e.g. hundreds or thousands of incoming commits). If you want to see more commits for a certain category you can expand this category by clicking the dashed area after its last commit. To expand all categories at once, you can use **Show More Commits (Temporarily)** from the action menu. You can permanently change the default number of displayed commits using System Properties.

> **Note**
> The behavior of how commit times will (or will not) be adjusted can be configured by system properties (smartgit.pushableCommits.preserveAuthorDate).

> **Tip**
> To just change the commit message of the last commit (even for a merge commit or if the working copy is not clean), invoke **Local|Edit Last Commit Message**.

Modify or Split Commit

Modify

To amend something to a selected (unpushed) commit, use the **Modify** option. This will start an interactive rebase and stop after the selected commit. Perform the modification and then click **Commit/Continue** toolbar button, select **Create Commit** and in the Commit dialog select the **Amend last commit** checkbox. Now click again **Commit/Continue** toolbar button and select **Continue Rebase**. To abort the Modify command and go back to the previous state, use the **Abort** toolbar button.

Split

To split 1 selected (unpushed) commit into 2 commits (with the same message), use the **Split** option. This will start an interactive rebase and stop with the commit's changes in the Index. Discard all changes that you don't want to have in the first commit, click **Commit/Continue** and select **Continue Rebase**. Normally this should be sufficient, but it might be necessary to fix potential conflicts in the next commit.

> For the split command, the interactive rebase stops after the selected commit, a `git reset --soft ^HEAD` is performed to put the commit's changes into the Index. After selecting **Continue Rebase**, the commit is applied a second time to add all the remaining parts that were not committed in the first commit, followed by the further commits.

Interactive Rebase

As stated above, you may perform a couple of modification operations already using specific commands, e.g. reordering commits by drag and drop. They will be performed immediately. If you have to make multiple changes at once or the immediate commands aborted because of

conflicts, you should rather use the Interactive Rebase.

To start the interactive rebase command, select the first commit that should be changed. In the occurring Interactive Rebase dialog you will be able to perform similar operations like in the Journal directly, e.g. squash commits, reorder using drag and drop, edit commit messages, but they only are performed when clicking the **Rebase** button. In case of conflicts, the rebase will stop (like a normal rebase, too). After solving the conflicts, click the **Commit/Continue** toolbar button and select **Continue Rebase**. To abort the interactive rebase and go back to the previous state, use the **Abort** toolbar button.