GIT Plug-in Users Manual

Symbolic Nuclear Analysis Package (SNAP)

Version 2.0.2 - August 02, 2021

Applied Programming Technology, Inc.

240 Market St., Suite 301
Bloomsburg PA 17815-1951

GIT Plug-in Users Manual

Applied Programming Technology, Inc.
by Kasey O'Connor, Bill Dunsford, and Don Ulshafer

***** Disclaimer of Liability Notice ******

The Nuclear Regulatory Commission and Applied Programming Technology, Inc. provide no express warranties and/or guarantees and further disclaims all other warranties of any kind whether statutory, written, oral, or implied as to the quality, character, or description of products and services, its merchantability, or its fitness for any use or purpose. Further, no warranties are given that products and services shall be error free or that they shall operate on specific hardware configurations. In no event shall the US Nuclear Regulatory Commission or Applied Programming Technology, Inc. be liable, whether foreseeable or unforeseeable, for direct, incidental, indirect, special, or consequential damages, including but not limited to loss of use, loss of profit, loss of data, data being rendered inaccurate, liabilities or penalties incurred by any party, or losses sustained by third parties even if the Nuclear Regulatory Commission or Applied Programming Technology, Inc. have been advised of the possibilities of such damages or losses.

Table of Contents

Chapter 1. Introduction

The Symbolic Nuclear Analysis Package (SNAP) consists of a suite of integrated applications designed to simplify the process of performing thermal-hydraulic analysis. SNAP provides a highly flexible framework for creating and editing input for engineering analysis codes as well as extensive functionality for submitting, monitoring and interacting with the analysis codes. A single analysis model is stored in a platform independent binary format (MED).

Revision control systems are software packages that are used to maintain changes to documents. These systems allow multiple users to create, and maintain documents while preserving a history of what those changes are, and when those changes were made. The most recent version of a document is called the "head" revision. Applying local changes to a document into a repository is called committing. Retrieving changes made remotely to a document is called "updating" the document. If a document has both local and remote changes, the document is said to contain "conflicts".

This plug-in provides a user interface for interacting with the GIT revision control system. Models may be added to a GIT repository if they have been saved to a directory that is managed by GIT. The plug-in also provides the ability to set the saved file directory to a git managed directory. There are two user interfaces for accessing the GIT commands: the revision control toolbar, and the Tools menu. The toolbar contains the five most used features of revision control: add, commit, change branch, logs and status. The Tools menu contains the full list of GIT capabilities supported by SNAP.

Chapter 2. Revision Control Toolbar

The primary user interface for the GIT plug-in is the revision control toolbar. The buttons in this toolbar provide quick access to revision controls for the currently selected model. Changing the currently selected model will automaticlly update the toolbar to reflect the new model. When a model is graphically editing a restart case the toolbar will be disabled.

Figure 2.1. Revision Control Toolbar

2.1. Add Button

 This button adds the current model to the staging area in preparation for committing a file to the repository. This button will enable whenever a change has been made to the working model and it has not already been staged.

2.2. Commit Button

 This button commits local changes to the GIT repsository. A commit message dialog, shown in Figure 2.2, “Commit Model Dialog” will prompt the user for a log message, after which the model will be added as the new head of the repository. This button is only enabled when the current working model is staged by the add button.

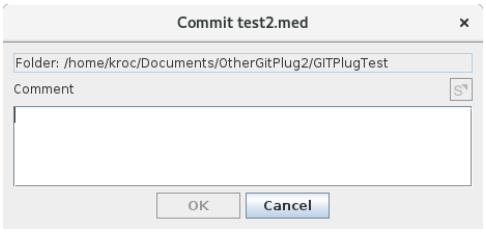

Figure 2.2. Commit Model Dialog

2.3. Select Branch Button

 This button allows the user to select a different branch in the repository. Pressing this button will display a list of the current branches that exist within the repository and allow the selection of which branch to check out. The model will then reload when a branch is changed. If the model no longer exists due to it not being present in the newly checked out branch, the model will close and a warning will be displayed.

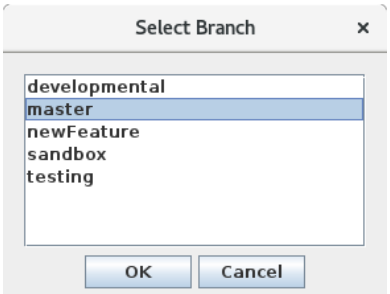

Figure 2.3. Select Branch Dialog

Note: Changing the current working branch for one model may affect other files as the enitre branch is checked out.

2.4. Status Button

 This button displays the current status of the file. Pressing this button will update the current status from the server and display the status dialog. The full list of possible status icons are displayed below.

-  *No Working Directory* - The current model either does not have a local save file or the save file is not in a directory managed by GIT.
-  *New File* - The current model is saved to a GIT managed directory but has not been added to the repository.
-  *Up To Date* - The current model is a clean copy of the head of the repository. No local or remote modifications have been made.
-  *Local Modifications* - The current model has local modifications, and no remote modifications have been made.
-  *Remote Modifications* - The current model has no local modifications but newer versions are available on the repository.
-  *Conflict (pre-merge)* - The current model has local modifications and new versions are available on the repository.
-  *Conflict (merging)* - Conflict resolution between local modifications and remote modifications has begun.

Chapter 3. GIT Tools Menu

The Tools menu on the main menu bar, and the Tools sub-menu on the right-click pop-up menu both include the GIT menu. This menu contains the full list of GIT operations supported by SNAP.

3.1. Create GIT Repository

This menu item exists on the main tools menu and allows a user create a local GIT repository. The command requires that the model be a saved model and not be currently part of a GIT repository. When selected, the current model's save directory will be used to initialize the git repository. The model editor user-interface will update to provide git controls. The model will still need to be staged and committed if that is desired.

3.2. Add Model To Repository

 This menu item adds the current model to a GIT repository when it has been saved to a directory that is managed by GIT. When pressed, a commit message dialog, shown in Figure 2.2, “Commit Model Dialog” will promp the user for a log message. After entering the commit log message, the model will be added and commited to the repository. The commit operation in this instance is performed automatically.

3.3. Commit Local Changes

 This menu item commits the current model to the head of the repository. When pressed, a commit message dialog, shown in Figure 2.2, “Commit Model Dialog”, will prompt the user for a log message. After entering the commit log message the model will be commited as the new head of the repository. This button is only enabled when the current model has local modifications, and there are no remote changes on the server.

3.4. Rename Model in Repository

 This menu item renames the current file in the repository. This deletes the current file on the repository and adds the model under a new name. When pressed, the rename dialog described in in Section 3.4.1, “Rename Dialog” will prompt the user for a new file name and a commit message. The model is then renamed in the repository. This will also commit any local modifications, and change the local save file on disk.

3.4.1. Rename Dialog

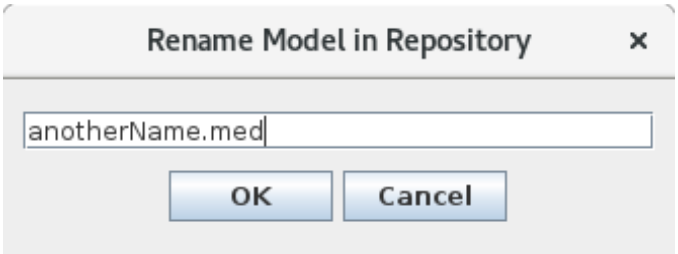

Figure 3.1. Rename Dialog

This dialog allows the user to select a new file name for the current model and to enter a commit log entry for the GIT rename operation. This dialog opens when the Rename Model In Repository menu item is selected.

3.5. Change Branch

 This menu item allows the user to select a different branch in the repository. Pressing this button will display a list of the current branches that exist within the repository and allow the selection of which branch to check out. The model will then reload when a branch is changed. If the model no longer exists due to it not being present in the newly checked out branch, the model will close and a warning will be displayed.

3.6. Create New Branch

 This menu item will create a new branch within the GIT repository. Creating a branch will make a copy of the current working branch with the specified name. This name cannot be a duplicate of another branch currently within the repository.

3.7. Merge Branch

 This menu item will allow the selection of another branch different from the current working branch in the repository. The selected branch and all associated files will then be merged into the current working branch. This may cause conflicts when merging different branches.

3.8. Resolve Conflicts

 This menu item only appears when the current local repository is in a state of conflict with unmerged paths, usually cause by a pull command. Selecting this item will open a dialog for resolving conflicts within the directory that contains the current model. The dialog will show a list of all files currently in conflict in the directory. It will provide columns to decide if the remotely pulled version of the file is desired or if the current local file is desired. After all conflicts are marked as resolved, pressing the Ok button will bring up a commit dialog in order to commit the merged paths. Once this process is complete it is recommended to push local changes to the remote repository in order to maintain proper workflow.

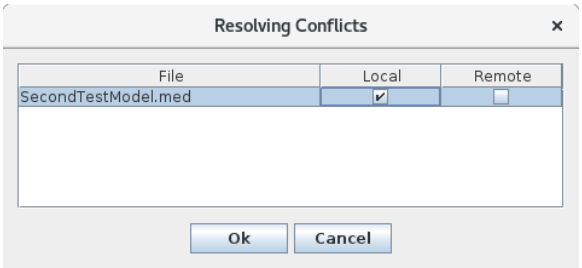

Figure 3.2. Conflict Resolution Dialog

3.9. Push Changes to Remote

 This menu item brings up a list of potential remote hosts to push current changes to. When using this command it is possible that it may cause conflicts and start a conflict resolution due to not being able to perform a fast-forward merge with the specified remote hosts. In this case the model editor will produce an error message and prompt the user to pull and start conflict resolution.

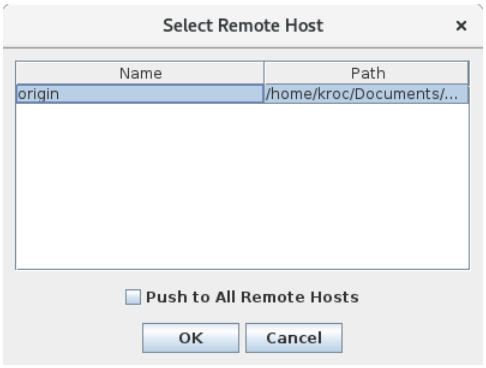

Figure 3.3. Push Changes Dialog

3.10. Pull Changes

 This menu item pulls remote changes from a specified remote host's repository. When using this command it is possible that it may cause conflicts and start a conflict resolution due to not being able to perform a fast-forward merge with the specified remote hosts.

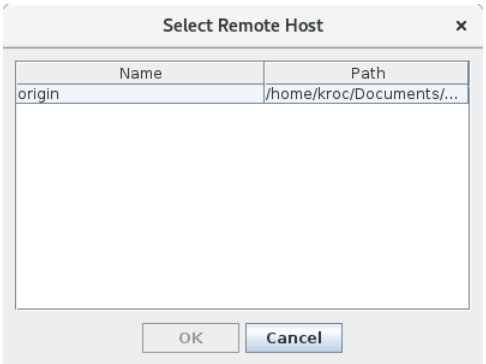

Figure 3.4. Pull From Remote Dialog

3.11. Compare to Base


This menu item is enabled only when the model supports graphical value comparison and the model is not new, in a directory not managed by GIT, or un-changed from the base. When pressed the multi-component comparison dialog opens showing the changes that have been made locally from the repository. Updates may only be merged from the repository head to the local version.

3.12. Compare to Revision


This menu item is enabled only when the model supports graphical value comparison and the model is not new or in a directory not managed by GIT. When pressed the Revision History dialog is shown. Once a specific revision has been selected the multi-component comparison dialog opens showing the changes between the local model and repository head. Updates may only be merged from the repository head to the local version.

3.13. Revert to Base


This menu item replaces the current file with the version checked out from the repository. Any local modifications will be lost.

3.14. Update To Revision


This menu item retreives the commit log for the current file on the server and displays it in a dialog. The local file will be replaced with the selected version. This operation supports navigating through renamed files, and will update both the current file and the selected file to the desired version. This operation will result in a new model being opened in a detatched head state. This operation will not discard local changes as they will still exist in the most recent current working branch.

3.15. Query File Status


This menu item updates the current file status. The icon of this menu item displays the current file status as described in Section 2.4, “Status Button”. When the operation is complete, the File Status Dialog will be displayed, which provides the details on the current file.

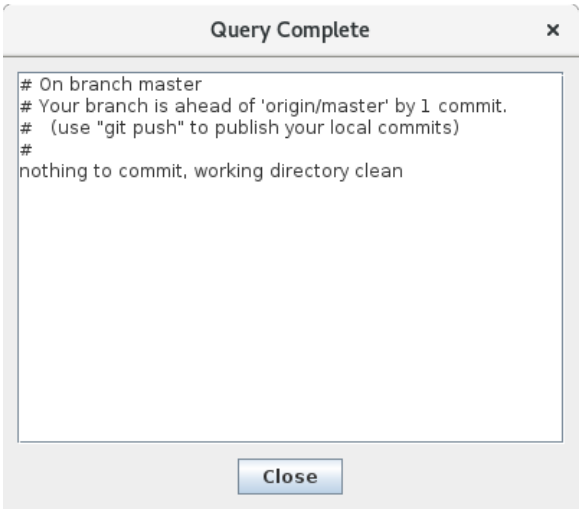

Figure 3.5. File Status Dialog

3.16. Show Commit Log

 This menu item retrieves the full commit log for the current file in the current working directory. This log will include file renames. An example of the Log History Dialog is shown below in Figure 3.6, “Log History Dialog”.

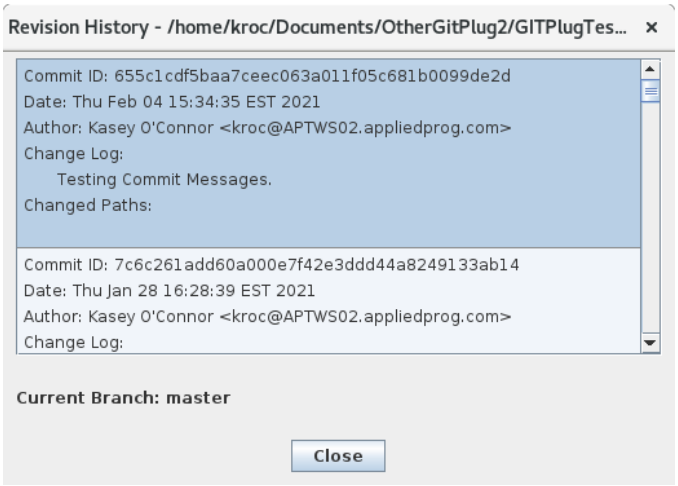

Figure 3.6. Log History Dialog

The Compare and Compare to Local buttons only appear if the working model supports graphical value based comparison. The Compare button allows comparing two revisions against one another or comparing a selected revision against the base revision. In either case the remote revisions will be downloaded and opened in the model editor background. The Multi-Component Comparison dialog will be opened to show the differences between the revisions. Differences will be flagged but values cannot be merged into either model. The Compare to Local button compares the local model (with modifications) to the selected revision.

3.17. Operation Availability

	Description							
 Add Model	Add current model to the staging area.	✗	✓	✗	✓	✗	✗	✗
 Rename Model	Rename the current model in the repository.	✗	✗	✓	✓	✗	✗	✗
 Commit Changes	Commit the staged changes to the local repository.	✗	✗	✗	✓	✗	✗	✓

Table 3.1. GIT Operation Availability

	Description							
 Switch Branch	Changes the current working branch and reloads the model.	✗	✗	✓	✗	✗	✗	✗
 Create Branch	Creates a new branch within the repository that is a copy of the current branch.	✗	✗	✓	✓	✓	✓	✓
 Merge Branch	Merges a branch into the current working branch.	✗	✗	✓	✗	✗	✗	✗

Table 3.2. GIT Operation Availability (continued)

	Description							
Resolve Conflicts	Opens the conflict resolution dialog.	X	X	X	X	X	✓	✓
Push Changes to Remote	Pushes local repository changes to a remote repository.	X	X	✓	X	X	X	✓
Pull Changes from Remote	Pulls changes from a remote repository into the local repository.	X	X	✓	✓	✓	✓	✓

Table 3.3. GIT Operation Availability (continued)

	Description							
Compare to Base	Compare local model to base.	X	X	X	✓	✓	✓	✓
Compare to Revision	Compare local model to a specific revision.	X	X	✓	✓	✓	✓	✓

Table 3.4. GIT Operation Availability (continued)

	Description							
 Revert to Base	Revert local model to base (discard any changes).	✗	✗	✗	✓	✗	✓	✓
 Update to Revision	Update local model to specific revision (discard any changes).	✗	✗	✓	✓	✓	✓	✓

Table 3.5. GIT Operation Availability (continued)

	Description							
 Query Model Status	Query and update the current model's status.	✓	✓	✓	✓	✓	✓	✓
 Show Commit Log	Show commit log	✓	✓	✓	✓	✓	✓	✓

Table 3.6. GIT Operation Availability (continued)