Portable Patch Tool

In the diversified development environment of today, multiple code repositories are used by cross development teams. As each have an indigenous way to deal with patches, a generic method of creating and applying patches is much desired along with recursive folder comparison and viewing the differences.

Portable patch tool attempts to address and has following features.

## Create Patch

1. Create patches between two folders
2. Filter out special folders and files
3. Include additional files
4. Create patches from a file list
5. Include source files in the patch
6. Work with any file type – text or binary

## Apply Patch

1. Preview changed files hierarchically before applying the patch
2. Apply changed files in patch selectively
3. Files that are changed are flagged

## Compare Folders

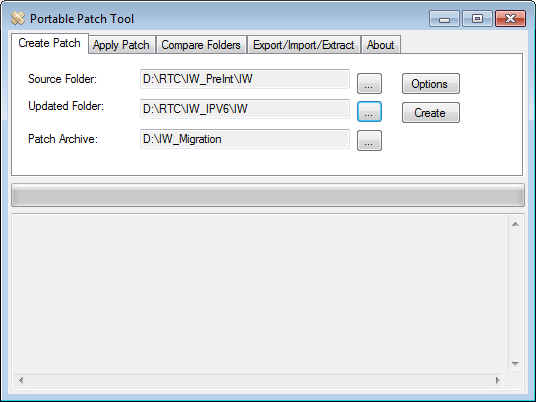
1. Recursively compare folders such that files are flagged as unique left or right and different
2. Options to view files and sort based on directory name, extension etc
3. Options to delete and copy files
4. Export file list
5. Options to view only left , right or diff files
6. Options to filter files including and excluding patterns

## Export/Import/Extract

1. Export patch files to a .ZIP file
2. Import patch files from zip file
3. Extract contents of a patch file

# Create patch

A Patch contains differences between two folders. It can also contain additional files.



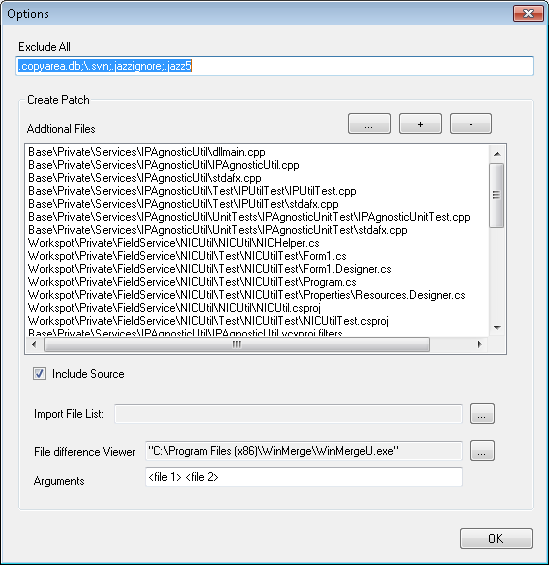
“Source Folder” contains the code downloaded from the source repository.

“Updated folder” contains developer changes that needs to be reviewed

“Patch Archive” is a temporary folder hosting the patches

### Options

The “Options” dialog gives additional options as discussed below:



Typically source control repositories use a hidden folder or file name such as .svn or .copyareadb etc. These need not be included in a patch.

Users can specify all such file extensions or folder names separated by a semicolon in “Exclude All”.

As mentioned earlier, the patch contains only files that are different. Additional files from “updated folder” can be added to or removed from patch using + or – buttons. The list of additional files are stored as additional\_input.txt in the patch folder.

In addition, a list of files can be inputted via text file using … button. This file can be exported in “Compare Folders” tab discussed below.

Typically the patch contains the files from “Updated Folder”. The “Include Source” option enables including corresponding files from “Source Folder” as well.

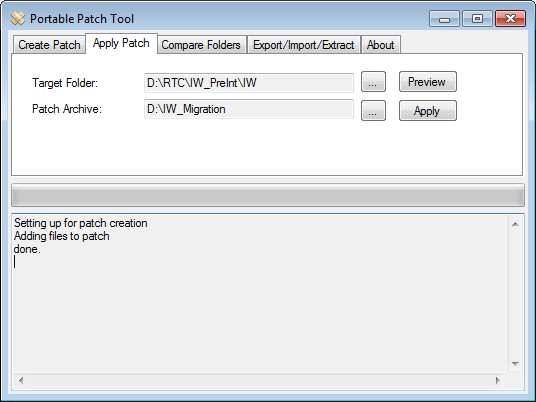
By default, every file that differs is included in the patch. Instead users can also provide a list of files to be included in a patch in the “Import File List”. Again this file can be exported in “Compare Folders” tab discussed below.

The differences of the files can be viewed by specifying the application in “File difference viewer”. The Arguments <File 1> and <File 2> must be specified along with others if any. They are replaced by actual file names when passed as command line arguments to the application.

Finally a patch is created by clicking the “Create” button. A patch consist of three files – patch\_meta.csv containing the metdata of the files in the patch. Patch\_data.dat containing the raw patch files and Patch\_source.dat containing the files from the |source folder” i

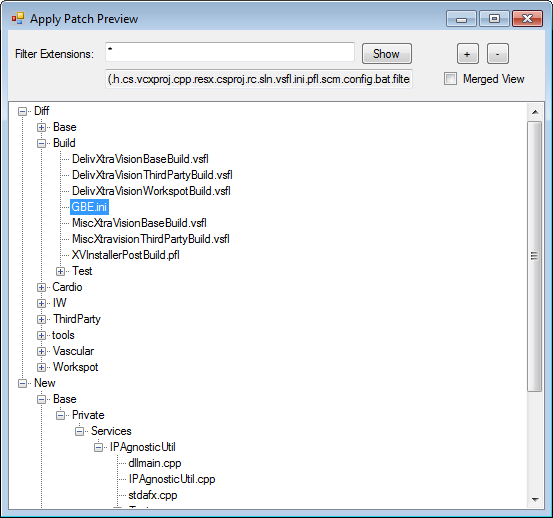
# Apply patch

Patches created in “Patch Archive” folder can be applied to the contents of the “Target Folder”.



Before applying the patch, changes can be previewed by clicking “Preview” button. This brings up GUI as shown below:

### Preview



Files are displayed hierarchically. The “Diff” branch contains files that are different. “New” contains files that are new.

If “Winmerge” is installed, changes of any file under “Diff” can be viewed by double clicking on it. Typing “Ctrl+C” copies the file name to clapboard.

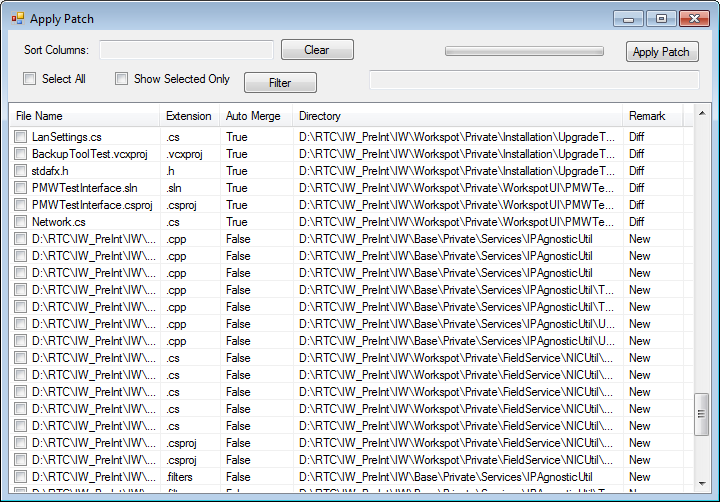
Clicking on any file under “New” loads it in “Notepad”.

Users can filter a specific file extension in “Filter Extensions”. Multiple file extensions can be separated by a semicolon. All possible file extensions available in the file set are also listed,

+ / \_ closes or expand tree.

“Merged View” combines “Diff” and “New” branches to present a unified view.

Patches are applied by clicking “Apply” button. This brings up GUI as shown below.



This displays a flattened view of the files “AutoMerge” column indicates weather Patch needs to be manually merged or not. True – means no and false means yes. In this case file can be manually merged using tool such as Winmerge.

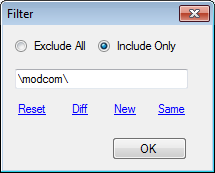
The “Remark” column has three values – “Diff” files are different. “Same” files are same. “New” the file is new.

The files can be sorted by clicking on any column header. This is listed in “Sort Columns”. The sorting can be cleared by clicking “Clear” button.

“Select All” selects all files.

### Filter

If the file set is huge, filtering of the files can be done by clicking the “Filter” button which brings up following GUI.



“Exclude All” option excludes all file containing the file pattern. And “Include Only” includes only files containing the file pattern.

Predefined patterns “Reset” removes any filtering. “Diff” ,“New” and “Same” shows files as in the “Remark” column discussed above.

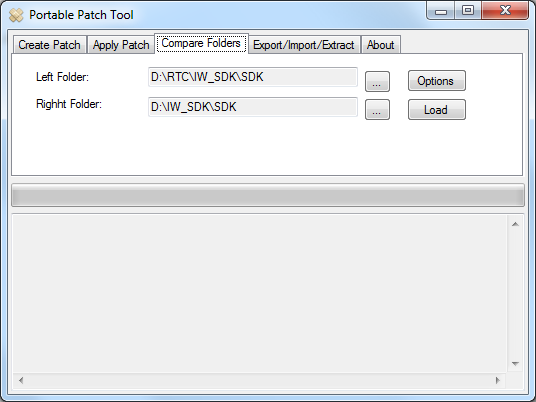
Finally files can be marked for applying. If the file list is huge, files can be selected based on the filtering. It’s possible that a certain filtering may not show previously selected files. In such cases “Show selected” options lists only the selected files overriding filtering.

“Apply Patch” button applies the patches. If “automerge” for a file is True, then the file in the “Target Folder” is overwritten. If the “Remark” is New, the file is copied into the “Target Folder”.

If the “automerge” is False and Remark is Diff, a new file is created appending “\_new” to the file name. Sometimes this is undesirable so in such cases, those files must be manually merged and selected out from patching. In other words, patching should be done to “automerge” True or New files only.

# Compare Folders

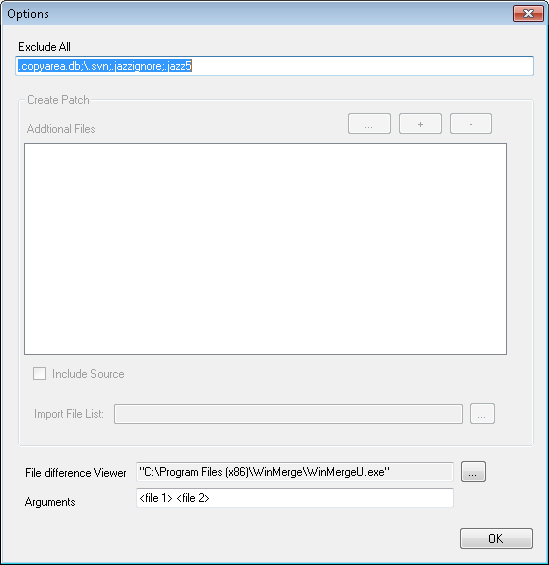
Two folders can be compared for differences in “Compare Folders”.



The “Left” and “Right” point to two folders with similar file structure. After comparison, a file can be categorized as “Diff” - file that exists in both folders but different, “Left” file only in Left folder. “Right” only in right folder.

### Options

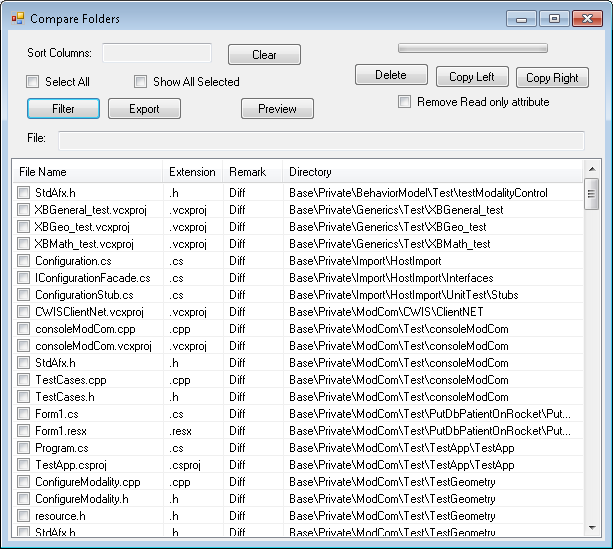
The “Options” dialog gives additional options as discussed below:



Typically source control repositories use a hidden folder or file name such as .svn or .copyareadb etc. These need not be included.

Users can specify all such file extensions or folder names separated by a semicolon in “Exclude All”.

The results of comparison are as shown below.

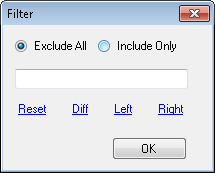


The “Remark” column has three values – “Diff” files are different. “Left” unique files. “Right” unique files. The files can be sorted by clicking on any column header. This is listed in “Sort Columns”. The sorting can be cleared by clicking “Clear” button.

“Select All” selects all files.

### Filter

If the file set is huge, filtering of the files can be done by clicking the “Filter” button which brings up following GUI.

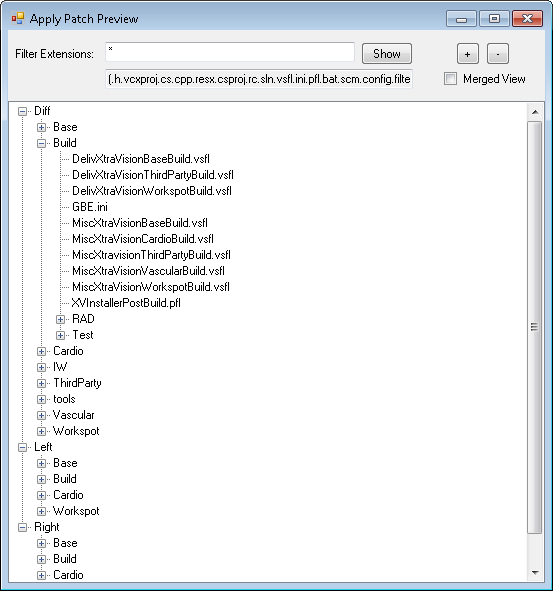


“Exclude All” option excludes all file containing the file pattern. And “Include Only” includes only files containing the file pattern.

Predefined patterns “Reset” removes any filtering. “Diff”, “Left” and “Right” files as in the “Remark” column discussed above.

Before applying the patch, changes can be previewed by clicking “Preview” button. This brings up GUI as shown below:

### Preview



Files are displayed hierarchically. The “Diff” branch contains files that are different. “Left” contains left unique files and “right” contains right unique files

If “Winmerge” is installed, changes of any file under “Diff” can be viewed by double clicking on it. Typing “Ctrl+C” copies the file name to clapboard.

Clicking on any file under “New” loads it in “Notepad”.

Users can filter a specific file extension in “Filter Extensions”. Multiple file extensions can be separated by a semicolon. All possible file extensions available in the file set are also listed,

+ / \_ closes or expand tree.

“Merged View” combines “Diff” ,“Left” and “Right” branches to present a unified view.

If the file list is huge, files can be selected based on the filtering. It’s possible that a certain filtering may not show previously selected files. In such cases “Show selected” options lists only the selected files overriding filtering.

The selected files can be exported using “Export” option to a txt file. This file can be later used to specify an additional file or input file during patch creation discussed earlier.

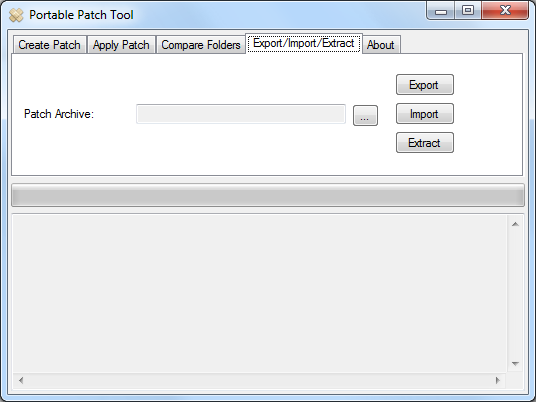
The “Delete” option deletes selected files.

The “Copy Left” option copies the selected files to the “Right Folder”. Similarly “Copy Right” option copies the selected files to the “Left Folder”.

The read only attribute of file can be also removed before delete or copy operation.

# Export / Import / Extract

As a Patch contain multiple files, it can be exported to a .zip file which can be imported.



“Patch archive” indicates folder that is used for export or import.

In addition, contents of patch can be extracted using “Extract” option, The differences are written to “left” folder the archive and if the source files are included in the patch, they are written to “right folder also.

# Storyboard Operation

The following discusses some of the examples.

## Deploy Patches

1. At the source the patch can be created as discussed in “Create Patch”
2. The patch can be exported as discussed in “Export/Import/Extract”
3. At the destination the patch can be imported as discussed in “Export/Import/Extract”
4. The patch can be applied as discussed in “Apply Patch”.

## Code Review

1. At the source the patch can be created as discussed in “Create Patch”
2. The patch can be exported as discussed in “Export/Import/Extract”
3. At the destination the patch can be imported and extracted as discussed in “Export/Import/Extract”
4. The reviewer views the changes and even adds review comments
5. Creates a patch and exports the patch back to the sender

## Code Rebase

During development of a feature, changes are not checked in to the repository for some time while the repository gets updated by others developers. In such cases the feature developer can stay updated as below.

1. Download latest code to repository workspace
2. Compare it with developer workspace as discussed in “Compare Folder”
3. Manually merge changes if necessary else overwrite.
4. Add in coming files and remove deleted files at source