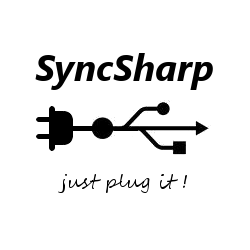
**Project name: SyncSharp**

****

**Team 13: EXCALIBUR**

**Azhar**

**Chris**

**Hong Lei**

**Jia Yuan**

**Tian Shuang**

**Yew Kang**

***Table of Content***

***Chapter 0: Introduction***

***Typographic Conventions 1***

***Introduction 2***

***Product Features***

***System Requirement***

***Support & Feedback***

***Chapter 1: Developers Guide***

***Part 1: Design Methodology***

***Part 2: Design Flow***

***System Architecture***

***Domain Model Analysis***

***Class Diagram***

***Sequence Diagram***

***Use Case Analysis***

***Activity Diagram - Detector***

***Activity Diagram - Reconciler***

***Algorithm Description - Detector***

***Algorithm Description - Reconciler***

***Chapter 2: Users Guide***

***Part 1: Product Description***

***Part 2: Using SyncSharp***

***On first run 13***

***Creating/deleting new Sync task 12***

***Modifying new Sync task 12***

***Review sync result 12***

***How to import/export Task 12***

***View Logs 25***

***PlugSync***

***Part 3: Troubleshooting***

***PlugSync is not fully functioning? 53***

***Chapter 4: Known Bugs***

***Chapter 3: Glossary***

***Chapter 4: Appendix***

**Typographic Convention**

All typographic conventions used in this documentation are shown below:

|  |  |  |
| --- | --- | --- |
| Visual cue | **Meaning** | |
|  | |  |
|  |  | |
|  |  | |
|  |  | |

**Chapter 0**

**Introduction**

**Introduction**

This documentation is written in response to develop a file (and folder) synchronization tool for CS3215 Software engineering Project.

File/Folder synchronization tools are commonly used to synchronize files and folders across multiple computers. They allow users to modify and update files and folders in two or more locations through certain rules. Most of synchronization tools provide users a one-way sync, where files and folders are copied from the source location to the target location, or two-way sync, where files and folders are copied in both locations so that files and folders are identical in both source and destination directories.

However, most of the sync tools that are available in the market required installation which may considered as a hassle to some users. It is because not all computers pre-installed with file synchronization software, and users may not be granted with administrative rights to install software, and this poses problems for users who need to perform file synchronization.

In order to solve the abovementioned problems, our team has developed a file (and folder) synchronization tool called SyncSharp which provides users with a streamline file (and folder) synchronization operation and installation free application.

**Product Features**

SyncSharp is a Windows based application which allows users to sync files and folders between multiple computers through an immediate device with no installation required. Our product provides one-way synchronization action since many users may just want to use file synchronization tool for backup purpose. It can be very helpful when users do not know which files already exist in the backup storage. The tool will automatically check for any new updates and perform synchronization action to files and folders while users perform other tasks on the computer. Our product also provides two-way synchronization action which can automatically detect which files and folders have been modified and perform synchronization actions between source and target directory based on pre-determined user preferences. In addition, SyncSharp is easy to use and supports all the basic features of a typical synchronization tool. SyncSharp offers users with automated synchronization with minimal user interaction.

A summary of SyncSharp features

* Create, edit and delete synchronization profiles
* Import/export synchronization profiles
* Ability to use environment variables in folder paths
* User settings
* Compare source and target directories
* Preview file contents before synchronization action
* Perform 2-way synchronization between the file/folder pairs
* Set inclusion/exclusion filters
* Backup files
* Generate log file after each synchronization operation

**System Requirements**

|  |  |
| --- | --- |
| Operating System: | Windows 2000,  Windows 2003,  Windows XP,  Windows Vista,  Windows 7[[1]](#footnote-1) |
| PC Configuration requirement | 128 RAM or more  50 MB of hard disk space. |

**Support & Feedback**

**Technical support**

For technical support, please contact us by email at [13-cs3215@gmail.com](mailto:13-cs3215@gmail.com).

**Feedback**

If you have any comments or suggestions for the next release, please direct them to [13-cs3215@gmail.com](mailto:13-cs3215@gmail.com). Your feedback is highly important for us. In order to get idea of how to make SyncSharp a better product for you, the current release is highly influenced by comments from users.

**Chapter 1**

**SyncSharp Developers Guide**

**Part1: Design Methodology**

Our team used top-down incremental design methodology in this project. It is because incremental design methodology allows us to preserve what we have done and save compilation time by taking previous compilation results so that only the portion that has just been modified are compiled each time.

In the incremental compilation flow, we wrote the codes and compiled them incrementally. It is much easier for us to fix bugs when we encounter them. It can also save time by allowing us to modify the critical portions while processing others. Reconciler performs actual file synchronization on source and target directories obtained from the Detector. In the rise of conflicting updates, pre-determined users’ settings will be used to resolve the conflicts. The summary of the updates will then be passed to the Logger. At last, Reconciler will update the metadata of the replica.

**Part2: Design Flow**

**System Architecture**



**Components Description**

|  |  |
| --- | --- |
| Component | Description |
| GUI | Provides the interface between users and the application. |
| ATD | Provides automated testing of the application functionalities during development. |
| SyncSharp Logic | Get input from GUI and initiates a response by making function calls to various sub-components. |
| Detector | Evaluates the changes on the designated folders / files based on the last synchronization operation which stores a small amount of information (called metadata). Metadata captures a snapshot of every file and folders’ state. Detector then passes a list of files to the reconciler to perform synchronization. |
| Reconciler | Performs file synchronization on the list of files obtained from the Detector. The file synchronization operation is based on the analyzed results. In the rise of conflicting updates, pre-determined users’ preferences will be used to resolve the updating conflicts. The summary of the updates will be passed to the Logger. Reconciler then updates the metadata of the replica. |
| Filter | Provides a list of filter rules that will be used by Detector for files retrieval. |
| Sync Profile | Stores the machine identity and contains a list of SyncTask associated with the profile. |
| SyncTask | Defines the pair of folders to be used for synchronization |
| TaskSettings | Stores all the configuration settings for each SyncTask. |
| Logger | Generates the summary of the file synchronization tasks. |
| FileUnit | Abstract representation of a file or folder. |

**Domain Model Analysis**



**Class Diagram**



**Sequence Diagram**



**Use Case Analysis**

The following table is a summary of all use cases.

|  |
| --- |
| No User Case |
| 1 Create Synchronization Tasks |
| 2 Edit Synchronization Tasks |
| 3 Delete Synchronization Tasks |
| 4 Run PlugSync |
| 5 Compare Source and Target Directories |
| 6 Perform 2-way Synchronization Between Source and Target  Directories |
| 7 Backup Files |
| 8 Restore Files |
| 9 View Source/Target Folders |
| 10 Export Synchronization Profiles |
| 11 Import Synchronization Profiles |
| 12 View Log Files |
| 13 View Help Files |

**User Case(s) Description**

|  |
| --- |
| Use Case Number: 1  Use Case Name: Create Synchronization Tasks |
| Pre-Conditions: SyncSharp is running and at the main window |
| Post-Conditions: System creates new Synchronization task and is displayed on the  main window |
| Actors: User, System |
| Main Success Scenario:   1. User clicks on “New” 2. System displays new SyncTask setup wizard 3. User enters name for SyncTask 4. System requests for SyncTask type: ‘Synchronize’ or ‘Backup’ 5. User selects SyncTask type 6. System requests path for Source and Target folder 7. User enters path for Source and Target folder 8. System creates new SyncTask and updates the main window   Extensions(s):  3a. User enters a name that already exists for a SyncTask  3a1. System displays name already exist error  Use case resumes from step 2.  5a. User did not select a SyncTask type before attempting to proceed  5a1. System prompts user to select a SyncTask type  Use case resumes from step 4.  7a. User enters non-existing/empty path for Source/Target folders  7a1. System prompts user to enter a valid path name  Use case resumes from step 6.  7b. User selects same path for Source/Target folders  7b1. System displays error that Source/Target folders cannot be the same  Use case resumes from step 6. |

|  |
| --- |
| User Case Number: 2  User Case Name: Edit Synchronization Tasks |
| Pre-Conditions: At least 1 SyncTask has already been created |
| Post-Conditions: SyncTask settings are updated and main window is updated to reflect  any changes |
| Actors: User, System |
| Main Success Scenario:   1. User selects a SyncTask from the main window and clicks on “Modify” 2. System displays the task setup window 3. User modifies the SyncTask settings as desired 4. System updates the SyncTask settings and updates the main window to reflect any changes   Extensions(s):  3a. User provides some invalid settings  3a1. System prompts user to correct any errors  Use case resumes from step 2. |

|  |
| --- |
| User Case Number: 3  User Case Name: Delete Synchronization Tasks |
| Pre-Conditions: At least 1 SyncTask has already been created |
| Post-Conditions: Select SyncTask is deleted and removed from the main window |
| Actors: User, System |
| Main Success Scenario:   1. User selects a SyncTask from the main window and clicks on “Delete” 2. System confirms with user to delete selected SyncTask 3. User selects ‘OK’ 4. System deletes selected SyncTask and removes it from the main window   Extension(s):  3a. User selects ‘Cancel’  Use case ends. |

|  |
| --- |
| User Case Number: 4  User Case Name: Run PlugSync |
| Pre-Conditions: At least 1 SyncTask has been created and PlugSync is enabled for this  SyncTask, SyncSharp is run from removable USB device, Computer’s  AutoPlay is enabled |
| Post-Conditions: Source/Target folder contents are synchronized |
| Actors: User, System |
| Main Success Scenario:   1. User inserts removable USB device 2. System automatically initiates 3. System retrieves a list of SyncTasks from current profile that has PlugSync enabled 4. System displays countdown that PlugSync is about to start 5. User waits for countdown period to end 6. System performs synchronization 7. System returns back to main window. Normal usage continues   Extension(s)  5a. User cancels PlugSync by clicking on “Back to Main”  Use case resumes from step 7 |

|  |
| --- |
| Use Case Number: 5  Use Case Name: Compare Source and Target Directories |
| Pre-Conditions: At least 1 SyncTask has been created |
| Post-Conditions: A window is displayed to the user that shows all the differences and  SyncActions that would be performed by synchronization |
| Actors: User, System |
| Main Success Scenario:   1. User selects SyncTask and clicks on ‘Analyze’ 2. System compares Source/Target folders and displays results to user   Extension(s):  2a. System determines that Source/Target folders are already synchronized, and displays message to user  User case ends |

|  |
| --- |
| Use Case Number: 6  Use Case Name: Perform 2-way Synchronization Between Source and Target Directories |
| Pre-Conditions: At least 1 SyncTask has been created, with ‘Synchronization’ type |
| Post-Conditions: Source/Target folder contents are synchronized |
| Actors: User, System |
| Main Success Scenario:   1. User selects SyncTask and clicks on ‘Synchronize’ 2. System proceeds to synchronize the Source/Target folders 3. System updates “Successful”, and last run time in the main window for the selected SyncTask   Extension(s):  2a. System encounters error during synchronization  2a1. System updates “Unsuccessful”, and last run time in the main window for  the selected SyncTask  Use case ends |

|  |
| --- |
| Use Case Number: 7  Use Case Name: Backup Files |
| Pre-Conditions: At least 1 SyncTask has been created with ‘Backup’ type |
| Post-Conditions: Any changes made to files/folders on Source directory will be updated  on Target directory |
| Actors: User, System |
| Main Success Scenario:   1. User selects SyncTask and clicks on ‘Backup’ 2. System proceeds to backup the Source folder to the Target folder 3. System updates “Successful”, and last run time in the main window for the selected SyncTask   Extension(s):  2a. System encounters error during backup  2a1. System updates “Unsuccessful”, and last run time in the main window for  the selected SyncTask  Use case ends |

|  |
| --- |
| Use Case Number: 8  Use Case Name: Restore Files |
| Pre-Conditions: At least 1 SyncTask has been created with ‘Backup’ type |
| Post-Conditions: Files/folders on the Target directory will be copied to the Source Directory |
| Actors: User, System |
| Main Success Scenario:   1. User selects SyncTask and clicks on ‘Restore’ 2. System proceeds to restore the Target folder to the Source folder 3. System updates “Successful”, and last run time in the main window for the selected SyncTask   Extension(s):  2a. System encounters error during restore  2a1. System updates “Unsuccessful”, and last run time in the main window for  the selected SyncTask  Use case ends |

|  |
| --- |
| Use Case Number: 9  Use Case Name: View Source/Target Folders |
| Pre-Conditions: At least 1 SyncTask has been created |
| Post-Conditions: Source/Target folders are opened and displayed to the user using  windows explorer |
| Actors: User, System |
| Main Success Scenario:   1. User selects SyncTask and clicks on ‘Action-> Open Source/Target Folder’ 2. System opens and displays Source/Target folders in windows explorer   Extension(s):  2a. Source/Target folder does not exist.  2a1. System displays error that Source/Target path cannot be found  Use case ends |

|  |
| --- |
| Use Case Number: 10  Use Case Name: Export Synchronization Profiles |
| Pre-Conditions: At least 1 SyncTask has been created |
| Post-Conditions: All SyncTasks for profile are exported to a \*.profile file |
| Actors: User, System |
| Main Success Scenario:   1. User selects ‘Export Task’ from main menu 2. System requests from user location and filename for exported file 3. User selects location and enters filename for exported file 4. System exports all SyncTasks for current profile into location with filename selected by user |

|  |
| --- |
| Use Case Number: 11  Use Case Name: Import Synchronization Profiles |
| Pre-Conditions: A SyncProfile has been previously exported |
| Post-Conditions: SyncTasks from exported profile will be imported and added into  current profile |
| Actors: User, System |
| Main Success Scenario:   1. User selects ‘Import Task’ from main menu 2. System request from user location of file to import 3. User selects file to import 4. System imports all SyncTasks from the export file into the current profile   Extension(s):  4a. System determines that user selected file to import is not valid  4a1. System displays error message to user  Use case resumes from step 2 |

|  |
| --- |
| Use Case Number:12  Use Case Name: View Log File |
| Pre-Conditions: At least 1 SyncTask has been created |
| Post-Conditions: System displays log file to the user |
| Actors: User, System |
| Main Success Scenario:   1. User selects SyncTask and clicks on ‘Action -> View Log’ 2. System displays log file to user   Extension(s):  2a. System cannot find log file associated with selected SyncTask  2a1. System informs user that no log file exists for select SyncTask  Use case ends |

|  |
| --- |
| Use Case Number: 13  Use Case Name: View Help File |
| Pre-Conditions: - |
| Post-Conditions: Help file is displayed to user |
| Actors: User, System |
| Main Success Scenario:   1. User clicks on ‘Help’ on the main menu 2. System displays help file to user   End of User case |

**Activity Diagram - Reconciler**

The following diagram is the activity diagram for reconciler.



**Activity Description - Reconciler**

**Activity 1: Backup source**

|  |
| --- |
|  |
| Description: |

**Activity 2: Sync with metadata**

|  |
| --- |
|  |
| Description: |

**Activity 3: Check files conflicts**

|  |
| --- |
|  |
| Description:  Iterate through the dirty source list and compare every file in the dirty source list with its corresponding file in dirty destination list to see what synchronization actions should be performed. Passes the synchronization actions to next block which is “Sync local files”. For example, if the source file has been modified and its corresponding file in destination has been deleted; then this block will pass sync action “keep source copy” to next block. |

**Activity 4: Execute sync action**

|  |
| --- |
|  |
| Description: |

**Activity 5: Check and create folders**

|  |
| --- |
|  |
| Description: |

**Activity Diagram – Detector**

The following diagram is the activity diagram for reconciler.



**Activity Description - Detector**

**Activity 1: Compare folders**

|  |
| --- |
|  |
| Description:  Detect file/folder changes on both source and target folders based on folders’ current states and its metadata. |

**Activity 2: Get current source file info**

|  |
| --- |
|  |
| Description:  Get file information (name, size, hash code, last modified date, etc) for all files in the source directory and store them in a list. |

**Activity 3: Get current target file info**

|  |
| --- |
|  |
| Description:  Get file information (name, size, hash code, last modified date, etc) for all files in the target directory and store them in a list. |

**Activity 4: Compare source file unit**

|  |
| --- |
|  |
| Description:  For each file in the source directory, compare it with corresponding file unit in source meta data. If the file was modified or newly created, add it to sDirFiles (source dirty files list) and flag it as ‘M-‘ or ‘C-‘ . If the file has not been modified, add it to sCleanFiles (source clean files list). Lastly, delete this file unit from source meta data. |

**Activity 5: Compare source directories**

|  |
| --- |
|  |
| Description:  For each sub directory in source directory, compare it with source meta data. If the folder info was found in the metadata, add it to sCleanDirs (source clean directories). Else if there is no relevant metadata, add it to sDirtyDirs (source dirty directories). Lastly, delete this folder info from source meta data. |

**Activity 6: Compare target file unit**

|  |
| --- |
|  |
| Description:  For each file in the target directory, compare it with corresponding file unit in target meta data. If the file was modified or newly created, add it to tDirFiles (target dirty files list) and flag it as ‘M-‘ or ‘C-‘ . If the file has not been modified, add it to tCleanFiles (target clean files list). Lastly, delete this file unit from target meta data. |

**Activity 7: Compare target directories**

|  |
| --- |
|  |
| Description:  For each sub directory in target directory, compare it with target meta data. If the folder info was found in the metadata, add it to tCleanDirs (target clean directories). Else if there is no relevant metadata, add it to tDirtyDirs (target dirty directories). Lastly, delete this folder info from target meta data. |

**Activity 8: Add deleted file unit on source to list**

|  |
| --- |
|  |
| Description:  For each file unit left in source meta data, if the file unit is a directory, add it into sDirtyDir (source dirty directories). If the file unit is a file, add it into sFileDir (source dirty files). Flag the file unit as ‘D-‘. |

**Activity 9: Add deleted file unit on target to list**

|  |
| --- |
|  |
| Description:  For each file unit left in target meta data, if the file unit is a directory, add it into tDirtyDir (target dirty directories). If the file unit is a file, add it into tFileDir (target dirty files). Flag the file unit as ‘D-‘. |

**Algorithm Description – Detector**

**Algorithm Description – Reconciler**

The following table despites how the reconciler handles different synchronization cases, the letter ‘M’, ‘D’, ‘C’, ‘R’ is the state of a file in the source or target directory. Their meaning is listed as follow:

‘M’ – ‘Modified’. File was modified.

‘D’ – ‘Deleted’. File was deleted.

‘C’ – ‘Created’. File was newly created.

‘R’ – ‘Rename’. File was renamed from another file.

We allow users to decide how the sync action will be performed based on four user settings, keep both copies, keep source copy, keep target copy and keep last modified. The default action will be to keep both copies.

(In this algorithm, we treat ‘file move’ the same as ‘file rename’. So there will be no ‘move’ flag or actions exclusively design for move.)

|  |  |  |
| --- | --- | --- |
| Source | Target | Sync Actions |
| M | M | Keep the latest copy by comparing their last modified time. |
| D | Copy the modified file in source directory to destination directory. |
| C | Keep both copies. |
| R | Keep both copies. |
| D | M | Copy the modified file in destination directory to source directory. |
| D | No action will be performed since both copies have been deleted. |
| C | Copy the newly created file in destination directory to source directory. |
| R | Copy the renamed file in destination directory to source directory. |
| C | M | Keep both copies. |
| D | Copy the newly created file in source directory to destination directory. |
| C | Action will be taken based on user settings. |
| R | If their contents are the same; then no action will be performed. Otherwise, keep both copies. |
| R | M | Copy the modified file in destination directory to source directory. |
| D | Copy the renamed file in source directory to destination directory. |
| C | If their contents are the same; then no action will be performed. Otherwise, keep both copies. |
| R | Keep both copies. |

**Chapter 2**

**SyncSharp User Guide**

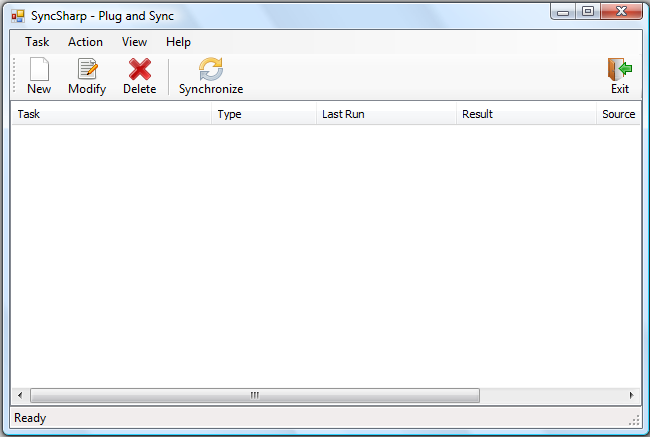
**Part 1 Product Description**

SyncSharp is a perfect product for tertiary students who work with multiple computers and need to synchronize files that reside in different computers through an immediate device such as a USB drive. It is a Windows based application that allows users to sync files between multiple computers through an immediate device with no installation required. The file synchronization process is streamlined through a list of pre-determined user preferences.

**Part 2 Using SyncSharp**

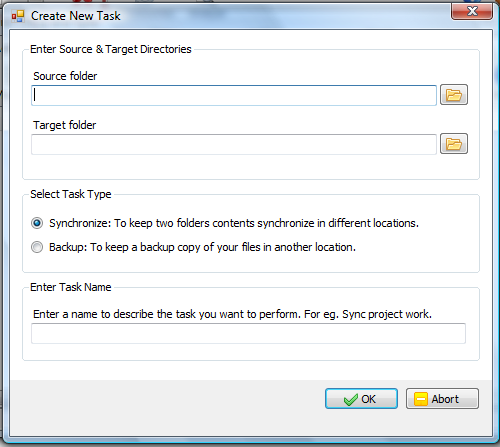
**On First Run**

On first run, user will need to activate SyncSharp by manually activate the program. Subsequently, the program will automatically run by itself. Upon double clicking the executable, the following main window will pop up.



**Creating/Deleting New Sync Task**

Create new Sync Task.



*Step 1:* Click on ‘***New***’, the ‘***Create New Task***’ window will show up.

*Step 2:* Click on the icon  to locate the two folders (“Source folder” and “Target folder”) that you would like to synchronize.

*Step 3:* ‘***Select Task Type***’.

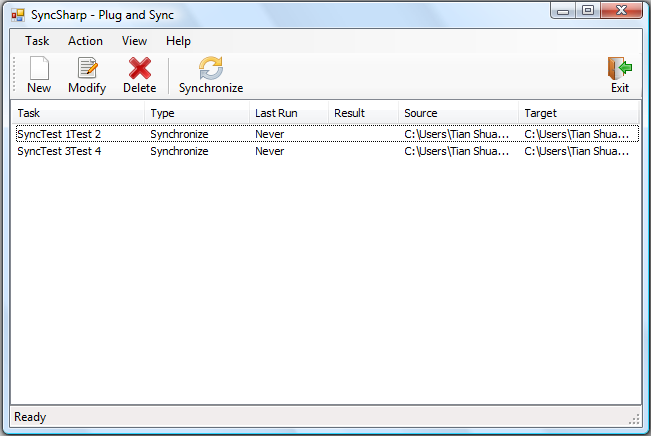
Synchronize – Synchronize source and target folders

Backup – Keep a copy of files in “Source folder” in “Target folder”.

*Step 4:* ‘***Enter Task Name***’. Specify the name of this Sync Task. (This task will be saved, so next time the program is activated. It will automatically try to complete this task)

*Step 5:* Click on “***OK***” to finish the wizard.

Delete Sync Task:



*Step 1:* Choose the Sync Task you want to delete.

*Step 2:* Click “***Delete***”.

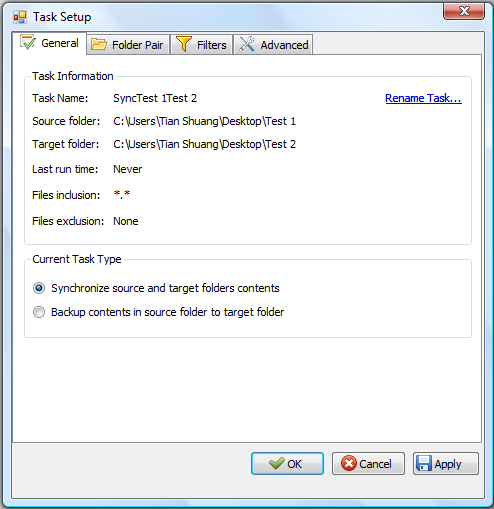
**Modify Sync Task**

*Step 1:* Click on the Sync Task you want to delete.

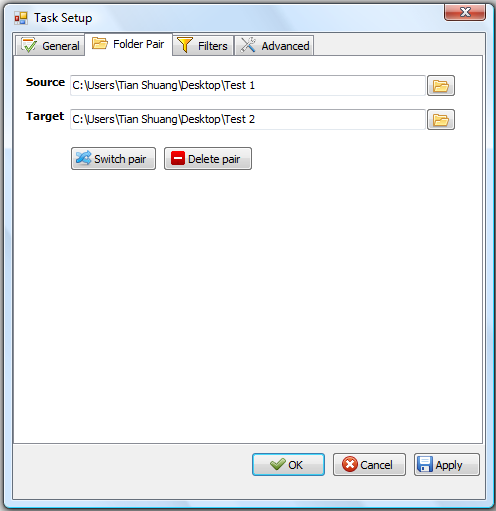
*Step 2:* Click ‘***Modify***’. The ‘***Task Setup***’ window will pop up. It consists of 4 tabs.

* ‘***General***’ – Display general information.
* ‘***Folder Pair***’ – Modify folder pairs.
* ‘***Filter***’ – Include/exclude file/folders in sync action.
* ‘***Advanced***’ – Advanced syncing options.

‘***General***’ Tab



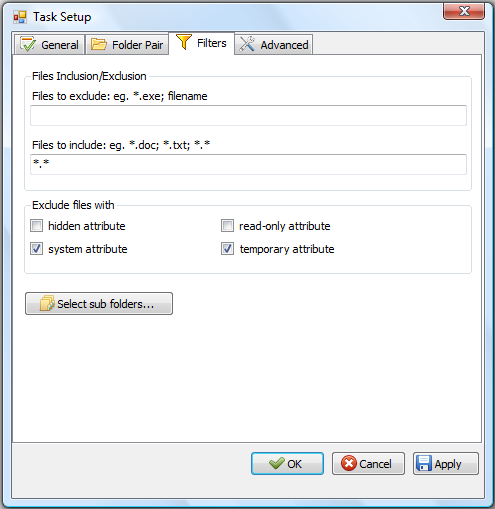
“***Folder Pair***” Tab



‘***Switch pair***’ – Switch the path of “Source” and “Target”

‘***Delete pair***’ – Delete the path of “Source” and “Target”

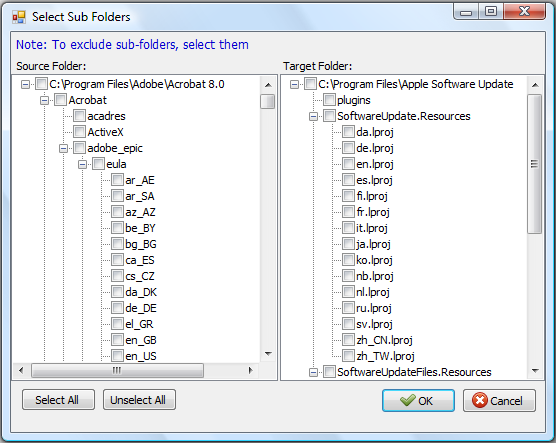
“***Filter***” Tab



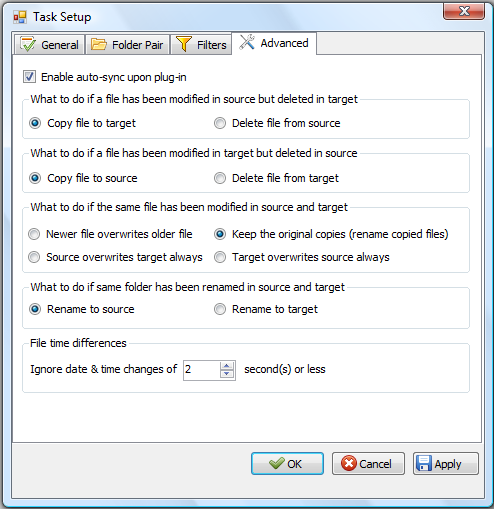
‘***File Inclusion/Exclusion***’ - Users are allowed to include/exclude files/subfolders in “source” and “target” directory in the sync process. Filtering is based on file name or extension, (For e.g., to exclude all files with extension ‘***.doc***’, put ‘***\*.doc***’ in ‘***Files to exclude***’ textbook). Users is allowed to set multiple filtering criteria separated by ‘***;***’.

‘***Exclude files with***’ – Filtering can also be achieved based files/folders attributes. The default files to exclude are files with ‘***system***’ and ‘***temporary***’ attribute.

‘***Select Sub folders…***’ will list out all sub folders in “Source” and “Target”. User is able to exclude sub-folders in the syncing process. (Below is a screenshot of its window)



**‘*Advanced*’** Tab

****

**‘*Enable auto-sync upon plug-in****’* – If checked, PlugSync will be enabled.

‘***File time differences***’ – While comparing two files or folders, if the difference in last modified time is less than or equals the pre-set value. These files or folders are considered to be with the same modification time.

**How to Import/Export Sync Task?**

Import SyncTask.

*Step 1*: On the main menu, click ‘***Task***’

*Step 2*; Click ‘***Import Task …***’ or simply use ‘Ctrl’ + ‘I’

*Step 3*: Specify which SyncTask to import.

Export SyncTask.

*Step1*: On the main menu, click ‘***Task***’

*Step 2*: Click ‘***Export Task …***’ or simply use Ctrl + X

*Step 3*: Specify the location you want the SyncTask to be saved.

**PlugSync**

The program will run by itself while the USB is plugged into a computer. Upon execution, the program will count down for 5 seconds (default) waiting for user interruption. If there is no interruption, the program will automatically try to perform all the saved SyncTask.

**Part 3 Troubleshooting**

**PlugSync is not fully functioning?**

On some Windows machines, ‘auto play’ of external storage device is disabled. However, in order to enable the PlugSync function, user will need to modify the ‘auto play’ settings. This guide will walk you through step by step on how to change the ‘auto play’ settings.

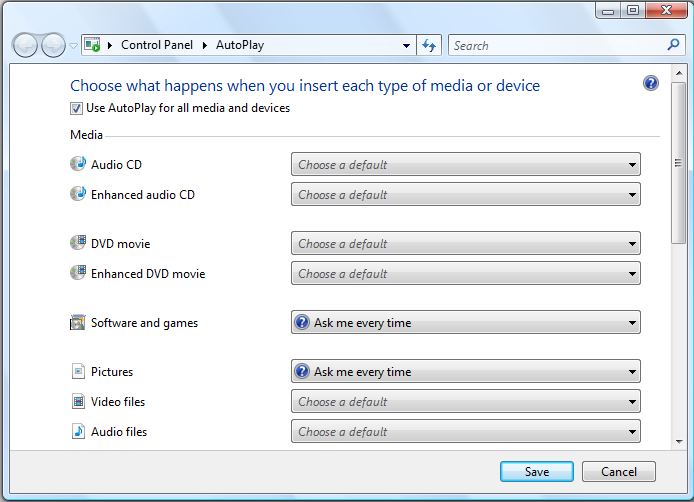
On Windows Vista:

*Step 1*: Click on ‘***Start***’ 🡪 In the search box, type in ‘***autoplay***’

*Step 2*: Choose the first returned result, which looks like



Step 3: Click on the icon, the following window will pop up.

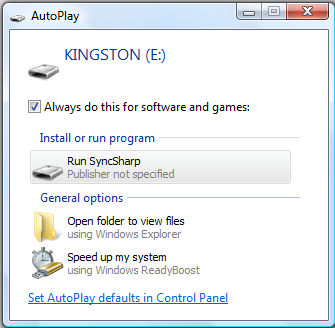


*Step 4*: Check the box ‘***Use AutoPlay for all media and devices***’

*Step 5*: In the drop-down list for ‘***Software and games***’, choose “***Ask me every time***’.

*Step 6*: Click on ‘***Save***’ to save the setting.

*Step 7*: The next time user plug in his external storage device, system will prompt user for actions. A window looks like the following will show. Check the box ‘***Always do this for software and games:***’ and click on ‘***Run SyncSharp***’ to activate the program. By doing this, this action is saved on the machine, and the PlugSync feature will function properly.

****

**Glossary**

|  |  |
| --- | --- |
| Term | Definition |
| SyncSharp | Name of our sync tool |
| SyncTask | Configuration file that contains source and target directory information to be synchronized |
| SyncProfile | Contains list of SyncTasks for a Particular PC/Laptop |
| FileUnit | Abstract representation of a file or folder, contains information such as name, size, hash code, last modified date, etc. |
| PlugSync | A feature in our program. Program will run by itself while the USB is plugged into a computer. Upon execution, the program will count down for 5 seconds waiting for user interruption. If there is no interruption, the program will automatically try to perform all the saved SyncTask. |
| 1 way sync | Update destination directory to have the same content as source directory |
| 2 ways sync | Update source and destination directories to have the same state |
| Report/Logger | Log file that records the operations perform in the synchronization process |
| Target | The destination directory to be sync or compared |
| Detector | The sub-system that detect changes of the source or destination directory |
| Reconciler | The sub-system that resolves conflicts between the source & destination directories |
| TaskSettings | Contains configuration settings made for each SyncTask |

1. For current release, PlugSync feature is not fully functioning in Windows 7 due to system restrictions. [↑](#footnote-ref-1)