First Posted	31 Jan 2001
Views	504,781
Bookmarked	276 times
Licence	

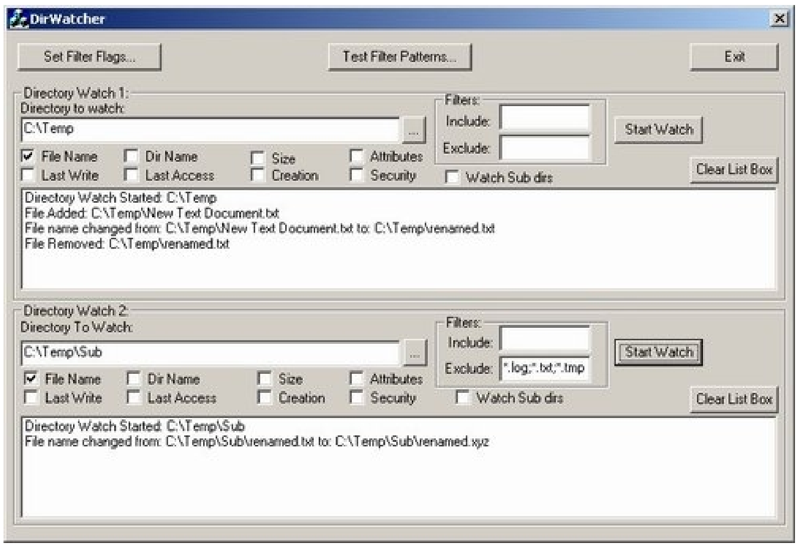
VC6, Win2K, MFC, Dev, Intermediate

CDirectoryChangeWatcher - ReadDirectoryChangesW all wrapped up

By **Wes Jones** | 11 May 2002

This class wraps up ReadDirectoryChangesW.

- Download demo project - 152 Kb
- Download sample executables - 32 KB
- Download source - 21 Kb



Directory to watch: C:\Temp
[✓ File Name]

```
Directory Watch Started: C:\Temp
File Added: C:\Temp\New Text Document.txt
File name changed from: C:\Temp\New Text Document.txt to: C:\Temp\renamed.txt
File Removed: C:\Temp\renamed.txt
```

Directory To Watch: C:\Temp\Sub
Exclude: *.log;*.txt;*.tmp
[✓ File Name]

```
Directory Watch Started: C:\Temp\Sub
File name changed from: C:\Temp\Sub\renamed.txt to: C:\Temp\Sub\renamed.xyz
```

Introduction

This code wraps up the Win32 API function ReadDirectoryChangesW so that your application only has to worry about responding to the events that take place when a file or directory is added, removed, modified, or renamed.

This code will only work on Windows NT, Windows 2000, or Windows XP, and the directory you wish to watch must also reside on a WindowsNT, Windows 2000, or Windows XP computer.

The Classes

There are two classes that must be used together to watch a directory, they are: `CDirectoryChangeWatcher` and `CDirectoryChangeHandler`.

About `CDirectoryChangeWatcher`:

The class `CDirectoryChangeWatcher` does all the grunt work of watching a directory. It creates a worker thread that is waiting for directory changes to take place via its use of the ReadDirectoryChangesW Win32 API function.   Multiple directories can be watched with a single instance of `CDirectoryChangeWatcher`, and directories can be added and removed from the watch at any time.

When file change notifications are received, `CDirectoryChangeWatcher` 'dispatches' these notifications to your application via the the class `CDirectoryChangeHandler`(see below).

About `CDirectoryChangeHandler`:

The class `CDirectoryChangeHandler` receives notifications about file changes from `CDirectoryChangeHandler`. You need to derive a class from `CDirectoryChangeHandler` in order for your application to handle the file change notifications.
A single instance of a `CDirectoryChangeHandler` derived class can be used to handle notifications for multiple directory watches at the same time, or you may specify different `CDirectoryChangeHandler` derived classes for each watched directory.

The Interfaces

The following is the public interface of `CDirectoryChangeWatcher`:

⊟ Collapse | Copy Code

```
class CDirectoryChangeWatcher{
public:
   enum   {  //options for determining the behavior of the filter tests.
           FILTERS_DONT_USE_FILTERS     = 1,
            FILTERS_CHECK_FULL_PATH = 2,
            FILTERS_CHECK_PARTIAL_PATH       = 4,
            FILTERS_CHECK_FILE_NAME_ONLY     = 8,
            FILTERS_TEST_HANDLER_FIRST       = 16,
            FILTERS_DONT_USE_HANDLER_FILTER = 32,
            FILTERS_DEFAULT_BEHAVIOR         = (FILTERS_CHECK_FILE_NAME_ONLY ),
            FILTERS_DONT_USE_ANY_FILTER_TESTS = (FILTERS_DONT_USE_FILTERS |
                                                 FILTERS_DONT_USE_HANDLER_FILTER)
          };
  CDirectoryChangeWatcher(bool bAppHasGUI = true,
                          DWORD dwFilterFlags = FILTERS_DEFAULT_BEHAVIOR);
  virtual ~CDirectoryChangeWatcher();//dtor

  DWORD  WatchDirectory( const CString & strDirToWatch,
                         DWORD dwChangesToWatchFor,
                         CDirectoryChangeHandler * pChangeHandler,
                         BOOL bWatchSubDirs = FALSE,
                         LPCTSTR szIncludeFilter = NULL,
                         LPCTSTR szExcludeFilter = NULL);

  BOOL IsWatchingDirectory(const CString & strDirName)const;
  int  NumWatchedDirectories()const;

  BOOL UnwatchDirectory(const CString & strDirToStopWatching);
  BOOL UnwatchAllDirectories();

  DWORD  SetFilterFlags(DWORD dwFilterFlags);
  DWORD GetFilterFlags() const;

...
};
```

The class `CDirectoryChangeHandler` has the following interface:

⊟ Collapse | Copy Code

```
class CDirectoryChangeHandler{
   ...
   BOOL UnwatchDirectory();
   ...
 protected:
  //override these functions:
   //Notification related:
   virtual void On_FileAdded(const CString & strFileName);
   virtual void On_FileRemoved(const CString & strFileName);
   virtual void On_FileModified(const CString & strFileName);
   virtual void On_FileNameChanged(const CString & strOldFileName,
                                   const CString & strNewFileName);
   virtual void On_ReadDirectoryChangesError(DWORD dwError);
   virtual void On_WatchStarted(DWORD dwError, const CString & strDirectoryName);
   virtual void On_WatchStopped(const CString & strDirectoryName);
   //Filter related:
   virtual bool On_FilterNotification(DWORD dwNotifyAction,
                                      LPCTSTR szFileName, LPCTSTR szNewFileName);
   ...
  };
```

To handle the events that happen when a file or directory is added, deleted, modified, or renamed, create a class derived from `CDirectoryChangeHandler` that does all of the things that you want to do when these events happen.

The Notifications

There are 7 notifications that you can receive when using these classes.

Notification	`CDirectoryChangeHandler` **function**	**Notification Description**	**Flag(s) required to receive notification--** (*dwChangesToWatchFor* parameter

			to `CDirectoryChangeWatcher::WatchDirectory()`)
Watch Started	On_WatchStarted	A directory watch has been started because `CDirectoryChangeWatcher::WatchDirectory` was called.	N/A
Watch Stopped	On_WatchStopped	A directory watch has been stopped because `CDirectoryChangeWatcher::UnwatchDirectory` or `CDirectoryChangeWatcher::UnwatchAllDirectories` was called. Can also be called when `CDirectoryChangeHandler`'s desctructor is called and there are 1 or more directories being watched at that time. ** See the comments in DirectoryChanges.h near the On_WatchStopped function to avoid RTFM errors which can occur under certain circumstances.**	N/A
Directory Watch Error	On_ReadDirectoryChangesError	An error occurred while watching a directory. In this condition, the watch is stopped automatically. You will not receive the On_WatchStopped notification.	N/A
File Added	On_FileAdded	A file was added to a watched directory (newly created, or copied into that directory).	FILE_NOTIFY_CHANGE_FILE_NAME and/or FILE_NOTIFY_CHANGE_DIR_NAME (for directories)
File Removed	On_FileRemoved	A file was deleted, or removed from the watched directory(ie: sent to the recycle bin, moved to another directory, or deleted)	FILE_NOTIFY_CHANGE_FILE_NAME FILE_NOTIFY_CHANGE_DIR_NAME
File Name Changed	On_FileNameChanged	A file's name has been changed in the watched directory. The parameters to this notification are the old file name, and the new file name.	FILE_NOTIFY_CHANGE_FILE_NAME FILE_NOTIFY_CHANGE_DIR_NAME
File Modified	On_FileModified	A file was modified in some manner in a watched directory. Things that can cause you to receive this notification include changes to a file's last accessed, last modified, or created timestamps. Other changes, such as a change to a file's attributes, size, or security descriptor can also trigger this notification.	FILE_NOTIFY_CHANGE_ATTRIBUTES FILE_NOTIFY_CHANGE_SIZE FILE_NOTIFY_CHANGE_LAST_WRITE FILE_NOTIFY_CHANGE_LAST_ACCESS FILE_NOTIFY_CHANGE_CREATION FILE_NOTIFY_CHANGE_SECURITY

The Filters

One of the new features to this class is the ability to filter out notifications so that you can receive notifications only for files that you want to receive notifications about. This feature enables you receive file notifications only for files you wish to know about based on the name of the changed file, or based on a function that you implement.

Without a filter, you will receive notifications for any and all file changes, as specified by the combination of flags passed as the *dwChangesToWatchFor* parameter to the `CDirectoryChangeWatcher::WatchDirectory` function(which is the default by the way).

There are 3 types of filters: An Include Filter, an Exclude Filter, and a programmer implemented virtual function which can decide whether or not the appropriate `CDirectoryChangeHandler::On_Filexxxxx()` function is called.

The Include, Exclude filters:

The Include and Exclude filters are string parameters that are passed to `CDirectoryChangeWatcher::WatchDirectory` when a directory watch is started. The filter is a pattern which can contain the DOS wildcard characters * and ?. Multiple patterns can be specified by separating each pattern with a semi-colon (;).

The following are examples of valid pattern strings that can be used to filter notifications:

"*.txt" <-- specifies only a single file pattern.
"*.txt;*.tmp;myfile???.txt;MySubFolder???*.doc" <-- specifies multiple file patterns.

Note that the supporting code for these string filters uses the PathMatchSpec Win32 API function to determine a pattern match. PathMatchSpec is implemented in shlwapi.dll version 4.71 or later. If you are running on NT4.0, and Internet Explorer 4.0 is not installed, a modified version of wildcmp is used instead.

What does the Include Filter mean?

The Include filter is used to tell `CDirectoryChangeWatcher` that you wish to receive notifications ONLY for files that match a certain pattern. All other file notifications are implicitly excluded because the file name does not match the 'Include Filter' pattern. ie: If you pass an 'Include Filter' of "*.txt", you will only receive notifications(File Added, File Removed, etc) for files with names that end with ".txt". You will not be notified of changes to any other files.

Note: It's better to specify a NULL or empty string, than to specify an Include Filter of "*.*".

What does the Exclude Filter mean?

With the Exclude Filter, you can choose to tell `CDirectoryChangeWatcher` to explicitly ignore notifications for changes to files based on the name of the changed file. For example, you can choose to ignore changes to .log files in a watched directory. To do so, you would specify an Exclude Filter of "*.log"

The Include and Exclude Filters are a powerful way of customizing the notifications that you wish to receive. To test your pattern strings, there is a dialog provided as part of the sample application. Press the "Test Filter Patterns..." button on the Sample App.

The Programmer Defined Filter:

You can also override the function : `virtual bool CDirectoryChangeHandler::On_FilterNotification()` . If `On_FilterNotification` returns true(the default), you will receive the notification. If `On_FilterNotification` returns false, the file notification is ignored. See the comments in the source code for more information about this function.

Filter Options

There are several options related to how `CDirectoryChangeWatcher` works with filters.

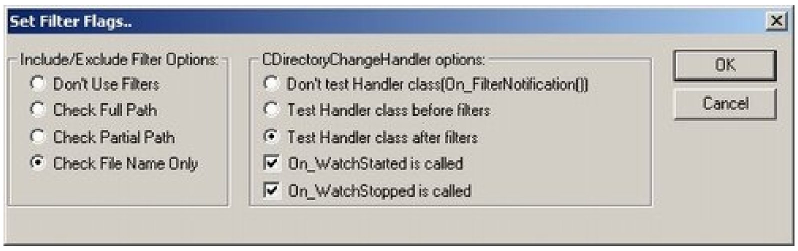
Filter Flag	**Flag Description**
FILTERS_DONT_USE_FILTERS	Specifies that the string filters for the Include and Exclude filter are not checked. All notifications are sent unless `CDirectoryChangeHandler::On_FilterNotification()` returns false.
FILTERS_DONT_USE_HANDLER_FILTER	Specifies that `CDirectoryChangeHandler::On_FilterNotification()` is not tested. If the File name passes the test against the Include and Exclude filter, the notification is passed on.
FILTERS_DONT_USE_ANY_FILTER_TESTS	Specifies that NO filter tests are to be performed. The Include and Exclude filters are not tested, and `CDirectoryChangeHandler::On_FilterNotification()` is not called. This is a combination of FILTERS_DONT_USE_FILTERS and FILTERS_DONT_USE_HANDLER_FILTER flags.
FILTERS_NO_WATCHSTART_NOTIFICATION	`CDirectoryChangeHandler::On_WatchStarted` won't be called
FILTERS_NO_WATCHSTOP_NOTIFICATION	`CDirectoryChangeHandler::On_WatchStopped` won't be called
FILTERS_NO_STARTSTOP_NOTIFICATION	`CDirectoryChangeHandler::On_WatchStarted` and `CDirectoryChangeHandler::On_WatchStopped` won't be called. Is a combination of FILTERS_NO_WATCHSTART_NOTIFICATION and FILTERS_NO_WATCHSTOP_NOTIFICATION.
FILTERS_TEST_HANDLER_FIRST	Specifies that `CDirectoryChangeHandler::On_FilterNotification()` is to be called BEFORE the file name is tested against the Include and Exclude filter patterns. The default is that `CDirectoryChangeHandler::On_FilterNotification()` is tested AFTER only if the file name matches the patterns used that may have been specified as Include or Exclude Filters.
FILTERS_CHECK_FULL_PATH	Specifies that the entire file name and path is to be tested against the Include and Exclude Filter pattern.
FILTERS_CHECK_PARTIAL_PATH	Specifies that only a portion of the file path and name are to be tested against the Include and Exclude Filter. The portion of the path checked is the part of the path following the watched folder name. For example, if you are watching "C:\Temp" (and are also watching subfolders) and a file named "C:\Temp\SomeFolder\SomeFileName.txt" is changed, the portion of the file name that is checked against the Include and Exclude filters is "SomeFolder\SomeFileName.txt"

FILTERS_CHECK_FILE_NAME_ONLY	Specifies that only the file name part of the file path is to be checked against the Include and Exclude filter.
FILTERS_DEFAULT_BEHAVIOR	Specifies the 'default' filter handling behaviour. Currently, it's only set to FILTERS_CHECK_FILE_NAME_ONLY. This implies that the Include/Exclude Filters are tested before `On_FilterNotification`, and that `On_WatchStarted` and `On_WatchStopped` are called.

To specify these options, see the constructor of `CDirectoryChangeWatcher` , or use the function `CDirectoryChangeWatcher::SetFilterFlags()` .

Note that the Filter Flags are used for the next call to `CDirectoryChangeWatcher::WatchDirectory` and that calling `CDirectoryChangeWatcher::SetFilterFlags()` will have no effect on any currently watched directories.

The sample app includes a dialog which allows you to test this out:


Set Filter Flags..

Include/Exclude Filter Options:
- Don't Use Filters
- Check Full Path
- Check Partial Path
- [✓ Check File Name Only]

CDirectoryChangeHandler options:
- Don't test Handler class(On_FilterNotification())
- Test Handler class before filters
- [✓ Test Handler class after filters]
- [✓ On_WatchStarted is called]
- [✓ On_WatchStopped is called]

Thread Safety, and Message Pumps.

While `CDirectoryChangeWatcher` uses a worker thread to get the job done, all notifications are called in the context of the main thread . The 'main' thread is the thread that first calls `CDirectoryChangeWatcher::WatchDirectory` . `CDirectoryChangeWatcher` uses a hidden notification window to dispatch notifications from the worker thread to the main thread. Because it uses a window, the calling application must have a message pump implemented somewhere in the program's 'main' thread.

Console Applications

For console applications, or applications/threads without a message pump, `CDirectoryChangeWatcher` can still be used. Just pass false as the value of the *bAppHasGUI* parameter to the constructor of `CDirectoryChangeWatcher` . Instead of using a hidden notification window, `CDirectoryChangeWatcher` uses an additional worker thread. Note that when you pass false as the value of *bAppHasGUI* parameter to the constructor of `CDirectoryChangeWatcher` , that all `CDirectoryChangeHandler` functions are called within the context of a worker thread, and NOT the main thread.

⊟ Collapse | Copy Code
```
CDirectoryChangeWatcher watcher(false); //safe to use in a console app.
                                   //
                                   //Note: CDirectoryChangeHandler functions are called
                                   //       in a worker thread.
```

A Sample:

⊟ Collapse | Copy Code

```
class CMyDirectoryChangeHandler : public CDirectoryChangeHandler
{
public:
        CMyDirectoryChangeHandler(){}
        virtual ~CMyDirectoryChangeHandler(){}

protected:
        void On_FileNameChanged(const CString & strOldFileName, const CString & strNewFileName)
        {
            MessageBox(NULL, _T("The file ") + strOldFileName + _T(" was renamed to: ") +
                             strNewFileName,_T("Test"),MB_OK);
        }
        bool On_FilterNotification(DWORD dwNotifyAction, LPCTSTR szFileName, LPCTSTR szNewFileN
        {
           //
           // This programmer defined filter will only cause notifications
           // that a file name was changed to be sent.
           //
                if( dwNotifyAction == FILE_ACTION_RENAMED_OLD_NAME )
                 return true;
                return false;
        }
};

....
       CDirectoryChangeWatcher watcher;
       CMyDirectoryChangeHandler MyChangeHandler;
       watcher.WatchDirectory(_T("C:\\Temp"),
                              FILE_CHANGE_NOTIFY_FILE_NAME,
                              &MyChangeHandler,
                              FALSE, //<-- watch sub directories?
                              NULL, //<-- Include Filter
```

Fin

CDirectoryChangeWatcher was based on the FWatch example program in the SDK and uses an I/O completion port so that there will only be one worker thread (per instance of CDirectoryChangeWatcher) for any number of watched directories.

When using this source code in your application, you only need to use CDirectoryChangeWatcher and class(es) derived from CDirectoryChangeHandler. Any other classes in these source files are used to implement CDirectoryChangeWatcher.

Download the sample or source code for more details.

Acknowledgements

- CDirectoryChangeWatcher is based on the FWatch sample in the SDK.
- The sample application uses CFolderDialog by Armen Hakobyan.
- CDirectoryChangeWatcher uses a modified version of wildcmp by Jack Handy

Feel free to email me with bugs, bug fixes, tips, comments, accolades, or admonitions at wesj@hotmail.com.

License

About the Author

Wes Jones

Web Developer

United States

Member

Resco MobileForms Toolkit
The Richest set of components for .NET CF
Try it now
resco.net

Article Top **_Rate this article for us!_** _Poor_ ○ ○ ○ ○ ○ _Excellent_ Vote

Comments and Discussions

FAQ [Search]

Noise Tolerance [Medium] Layout [Normal] Per page [25] [Update]

New Message Msgs 1 to 25 of 418 (Total in Forum: 418) (Refresh) First Prev Next

Not being able to save watched binary file from the application [modified] NEW	**Jack123sweet**	**13:47 14 Jul '10**
Support Windows 7 ? NEW	**wxlwr**	**1:57 5 Jul '10**
missing files under directory	**panly1984**	**5:04 29 Jun '10**
Re: missing files under directory	Wes Jones	8:37 29 Jun '10
when the msg happend?	木瓜	**2:13 8 Jun '10**
Re: when the msg happend?	Wes Jones	17:42 8 Jun '10
USB Drives, and missed files	**kamocio**	**15:32 11 Dec '09**
Re: USB Drives, and missed files	Wes Jones	22:52 11 Dec '09
Re: USB Drives, and missed files	krishnakumartm	1:23 9 Jan '10
good job~~	**zjj9850**	**7:06 13 Oct '09**
Re: good job~~	Wes Jones	9:28 22 Oct '09
nice work	**IicFox**	**0:21 18 Sep '09**
Re: nice work	Wes Jones	14:08 2 Oct '09
Could you give me the answer about How to monitor file system?	**YUN Daqing**	**21:21 7 Jul '09**
Re: Could you give me the answer about How to monitor file system?	Victor43	9:45 3 Sep '09
Re: Could you give me the answer about How to monitor file system?	YUN Daqing	1:49 4 Sep '09
C# code ??	**alhambra-eidos**	**8:05 2 Jul '09**
Re: C# code ??	Wes Jones	8:42 2 Jul '09
ReaddirectorychangesW + samba	**raphaelverdier**	**7:48 7 May '09**
Re: ReaddirectorychangesW + samba	Wes Jones	10:22 7 May '09
Re: ReaddirectorychangesW + samba	dan neely	13:23 13 Jul '09
Multiple Watcher	**NewVC++**	**22:10 18 Mar '09**
error C2248	**MsmVc**	**0:21 17 Mar '09**
Re: error C2248	Wes Jones	8:56 17 Mar '09
Re: error C2248	MsmVc	9:16 17 Mar '09

Last Visit: 20:42 4 Jul '10 Last Update: 20:42 22 Jul '10 **1** 2 3 4 5 6 7 8 9 10 11 Next »

General News Question Answer Joke Rant Admin

Use Ctrl+Left/Right to switch messages, Ctrl+Up/Down to switch threads, Ctrl+PgUp/PgDown to switch pages.

Sponsored Links

dtSearch® Engine for Win & .NET (64-bit / 32-bit)
Add Instant Searching to Web-Based or Other...
www.dtsearch.com

Data Dynamics Analysis for .NET by GrapeCity
First Free-form, on-line analytical processing...
www.datadynamics.com

OffByZero Cobalt
OffByZero Cobalt is a turnkey software...
cobalt.offbyzero.com

See Also...

Creating Custom Controls
An introduction to creating custom controls...

HTML 4.01 Programmer's Reference - Chapter 1: Introduction
Using the <object> tag to embed functionality...

XSLT Programmer's Reference 2nd Edition
This chapter is about the purpose of XSLT and...

Summer 2004 NYC CPian Get Together
An article on a NYC get together

Multi-tier Enterprise Application Architecture
Multi-tier Enterprise Application Architecture

Announcements

Have an app on Windows Azure? Win a laptop!

Monthly Competition

The Daily Insider

A 40-year-old computer demo that still amazes
Daily News: Signup now.

THE CODE PROJECT™ Daily Insider