Walking a Buffer of Change Journal Records

The control codes that return update sequence number (USN) change journal records, **FSCTL_READ_USN_JOURNAL** and **FSCTL_ENUM_USN_DATA**, return similar data in the output buffer. Both return a USN followed by zero or more change journal records, each in a USN_RECORD_V2 or **USN_RECORD_V3** structure.

The target volume for USN operations must be ReFS or NTFS 3.0 or later. To obtain the NTFS version of a volume, open a command prompt with Administrator access rights and execute the following command:

FSUtil.exe FSInfo NTFSInfo *X*:

where *X* is the drive letter of the volume.

The following list identifies ways to get change journal records:

- Use **FSCTL_ENUM_USN_DATA** to get a listing (enumeration) of all change journal records between two USNs.
- Use **FSCTL_READ_USN_JOURNAL** to be more selective, such as selecting specific reasons for changes or returning when a file is closed.

Note Both of these operations return only the subset of change journal records that meet the specified criteria.

The USN returned as the first item in the output buffer is the USN of the next record number to be retrieved. Use this value to continue reading records from the end boundary forward.

The **FileName** member of USN_RECORD_V2 or **USN_RECORD_V3** contains the name of the file to which the record in question applies. The file name varies in length, so **USN_RECORD_V2** and **USN_RECORD_V3** are variable length structures. Their first member, **RecordLength**, is the length of the structure (including the file name), in bytes.

When you work with the **FileName** member of USN_RECORD_V2 and **USN_RECORD_V3** structures, do not assume that the file name contains a trailing '\0' delimiter. To determine the length of the file name, use the **FileNameLength** member.

The following example calls **FSCTL_READ_USN_JOURNAL** and walks the buffer of change journal records that the operation returns.

C++

```
#include <Windows.h>
#include <WinIoCtl.h>
#include <stdio.h>

#define BUF_LEN 4096

void main()
{
   HANDLE hVol;
   CHAR Buffer[BUF_LEN];

   USN_JOURNAL_DATA JournalData;
   READ_USN_JOURNAL_DATA ReadData = {0, 0xFFFFFFFF, FALSE, 0, 0};
   PUSN_RECORD UsnRecord;

   DWORD dwBytes;
   DWORD dwRetBytes;
   int I;

   hVol = CreateFile( TEXT("\\\\.\\c:"),
               GENERIC_READ | GENERIC_WRITE,
               FILE_SHARE_READ | FILE_SHARE_WRITE,
               NULL,
               OPEN_EXISTING,
               0,
               NULL);

   if( hVol == INVALID_HANDLE_VALUE )
   {
```

```
    printf("CreateFile failed (%d)\n", GetLastError());
    return;
}

if( !DeviceIoControl( hVol,
       FSCTL_QUERY_USN_JOURNAL,
       NULL,
       0,
       &JournalData,
       sizeof(JournalData),
       &dwBytes,
       NULL) )
{
    printf( "Query journal failed (%d)\n", GetLastError());
    return;
}

ReadData.UsnJournalID = JournalData.UsnJournalID;

printf( "Journal ID: %I64x\n", JournalData.UsnJournalID );
printf( "FirstUsn: %I64x\n\n", JournalData.FirstUsn );

for(I=0; I<=10; I++)
{
    memset( Buffer, 0, BUF_LEN );

    if( !DeviceIoControl( hVol,
          FSCTL_READ_USN_JOURNAL,
          &ReadData,
          sizeof(ReadData),
          &Buffer,
          BUF_LEN,
          &dwBytes,
          NULL) )
    {
        printf( "Read journal failed (%d)\n", GetLastError());
        return;
    }

    dwRetBytes = dwBytes - sizeof(USN);

    // Find the first record
    UsnRecord = (PUSN_RECORD)(((PUCHAR)Buffer) + sizeof(USN));

    printf( "************************************\n");

    // This loop could go on for a long time, given the current buffer size.
    while( dwRetBytes > 0 )
    {
        printf( "USN: %I64x\n", UsnRecord->Usn );
        printf("File name: %.*S\n",
                 UsnRecord->FileNameLength/2,
                 UsnRecord->FileName );
        printf( "Reason: %x\n", UsnRecord->Reason );
        printf( "\n" );

        dwRetBytes -= UsnRecord->RecordLength;

        // Find the next record
        UsnRecord = (PUSN_RECORD)(((PCHAR)UsnRecord) +
                 UsnRecord->RecordLength);
    }
    // Update starting USN for next call
    ReadData.StartUsn = *(USN *)&Buffer;
}
```

```
    CloseHandle(hVol);

}
```

The size in bytes of any record specified by a **USN_RECORD_V2** or **USN_RECORD_V3** structure is at most `((MaxComponentLength - 1) * Width) + Size` where *MaxComponentLength* is the maximum length in characters of the record file name. The width is the size of a wide character, and the *Size* is the size of the structure.

To obtain the maximum length, call the **GetVolumeInformation** function and examine the value pointed to by the *lpMaximumComponentLength* parameter. Subtract one from *MaxComponentLength* to account for the fact that the definition of **USN_RECORD** and **USN_RECORD_V3** includes one character of the file name.

In the C programming language, the largest possible record size is the following:

```
(MaxComponentLength - 1) * sizeof(WCHAR) + sizeof(USN_RECORD)
```

Community Additions

does not work on Win8

Win8 (and I suppose Server 2012 too) needs the min/max version fields initialized, otherwise fails with NOT_IMPLEMENTED

Should be:

READ_USN_JOURNAL_DATA
ReadData = {0, 0xFFFFFFFF,
FALSE
, 0, 0, 0, /* this is required: */ **2, 2**};


javcz
11/24/2012

The target volume for USN operations must be NTFS 3.0 or later.

<< To obtain the NTFS version of a volume, open a command prompt with Administrator access rights and execute the following command: >>

See: http://msdn.microsoft.com/en-us/library/aa364569(VS.85).aspx
DeviceIoControl + FSCTL_GET_NTFS_VOLUME_DATA

If you pass a large enough buffer.

ByteCount
MajorVersion
MinorVersion


MedievalKnight
1/23/2010

needs to run as admin too...

on vista be sure to elevate first

Thomas Lee
11/15/2008