WAL / MANIFEST

Made by Isu Kim

E-Mail: isukim16@gmail.com

Content

CURRENT and MANIFEST

- LevelDB knows which MANIFEST file to use by CURRENT.

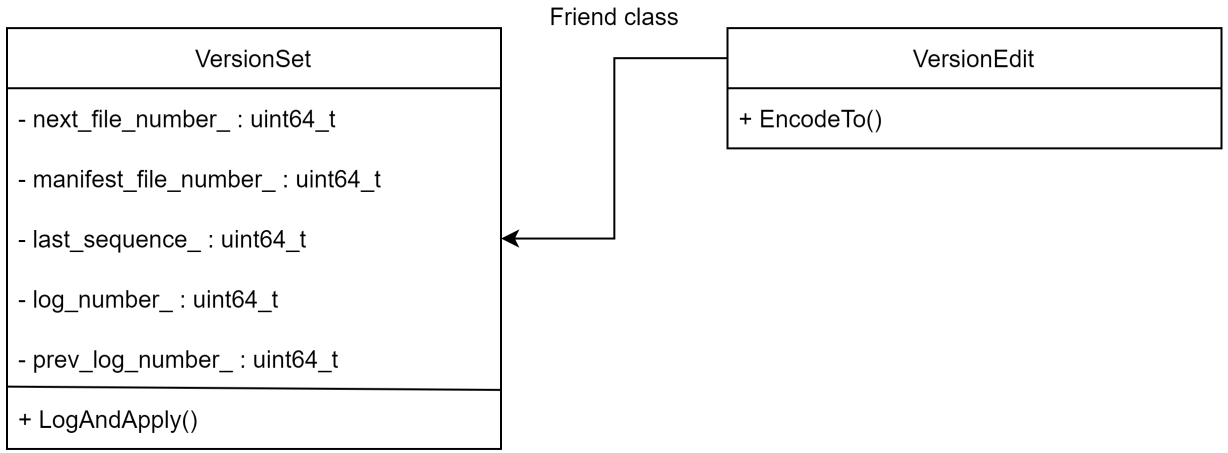
```
gooday2die@flagship:~/projects/School/2022_0.5/LevelDB/newnewnew/leveldb_debug/build$ ls -al /tmp/leveldbtest-1000/dbbench/
total 7392
drwxr-xr-x 2 gooday2die gooday2die    4096  8월  29 20:08 .
drwxr-xr-x 7 gooday2die gooday2die    4096  8월  29 20:08 ..
-rw-r--r-- 1 gooday2die gooday2die 1808353  8월  29 20:08 000005.ldb
-rw-r--r-- 1 gooday2die gooday2die 1808519  8월  29 20:08 000007.ldb
-rw-r--r-- 1 gooday2die gooday2die 2116942  8월  29 20:08 000008.log
-rw-r--r-- 1 gooday2die gooday2die 1809276  8월  29 20:08 000009.ldb
-rw-r--r-- 1 gooday2die gooday2die      16  8월  29 20:08 CURRENT
-rw-r--r-- 1 gooday2die gooday2die       0  8월  29 20:08 LOCK
-rw-r--r-- 1 gooday2die gooday2die     788  8월  29 20:08 LOG
-rw-r--r-- 1 gooday2die gooday2die     269  8월  29 20:08 MANIFEST-000002
```

- Content of CURRENT represents which MANIFEST to use.

```
gooday2die@flagship:~/projects/School/2022_0.5/LevelDB/newnewnew/leveldb_debug/build$ cat /tmp/leveldbtest-1000/dbbench/CURRENT
MANIFEST-000002
gooday2die@flagship:~/projects/School/2022_0.5/LevelDB/newnewnew/leveldb_debug/build$
```

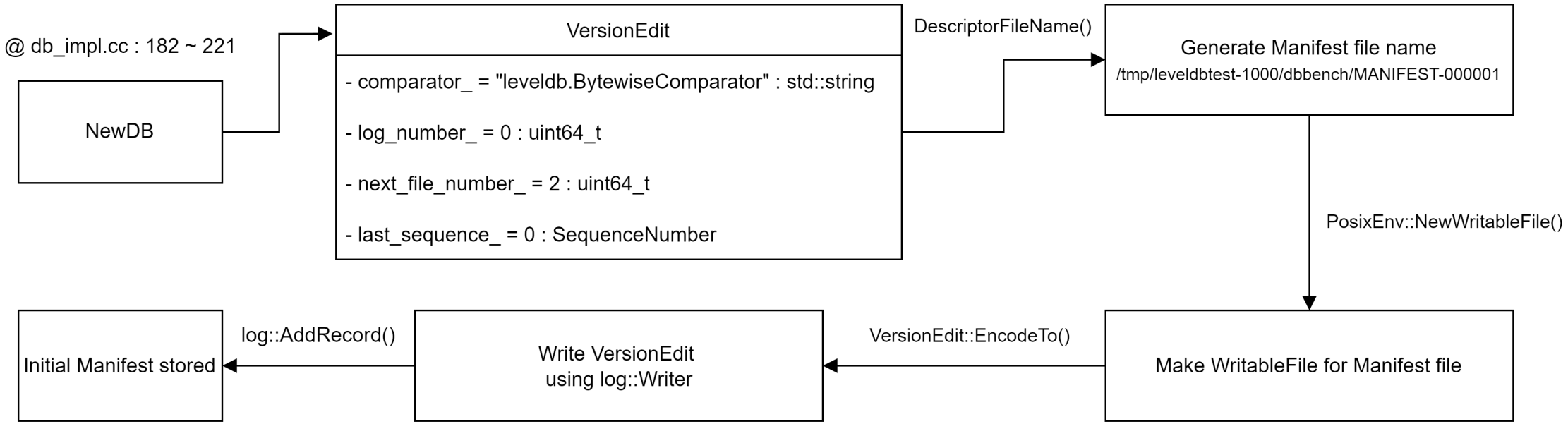
VersionSet & VersionEdit

- VersionEdit is friend class of VersionSet
- They both store information about versions.
- LogAndApply in VersionSet, EncodeTo in VersionEdit is for recording to MANIFEST file.



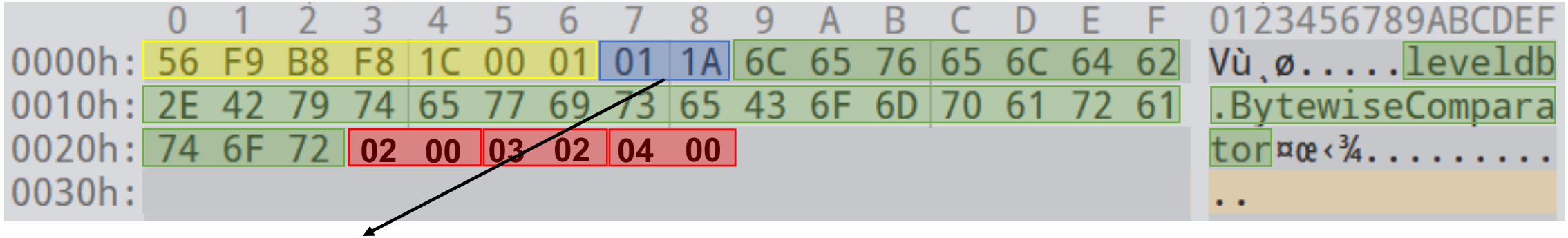
Initial MANIFEST

- When DB is being created by DB::NewDB(), this automatically generates MANIFEST file.
- MANIFEST file is also a PosixEnv::WritableFile object and uses log::Writer



Initial MANIFEST - Example

- An example with initial MANIFEST
  - MANIFEST when LevelDB is being **created.** (Defined in db_impl.cc @ 182 ~ 221)



- 0x01 means this is a kComparator
- 0x1A represents total length of “leveldb.BytewiseComparator”
- This information is generated by VersionEdit::EncodeTo()

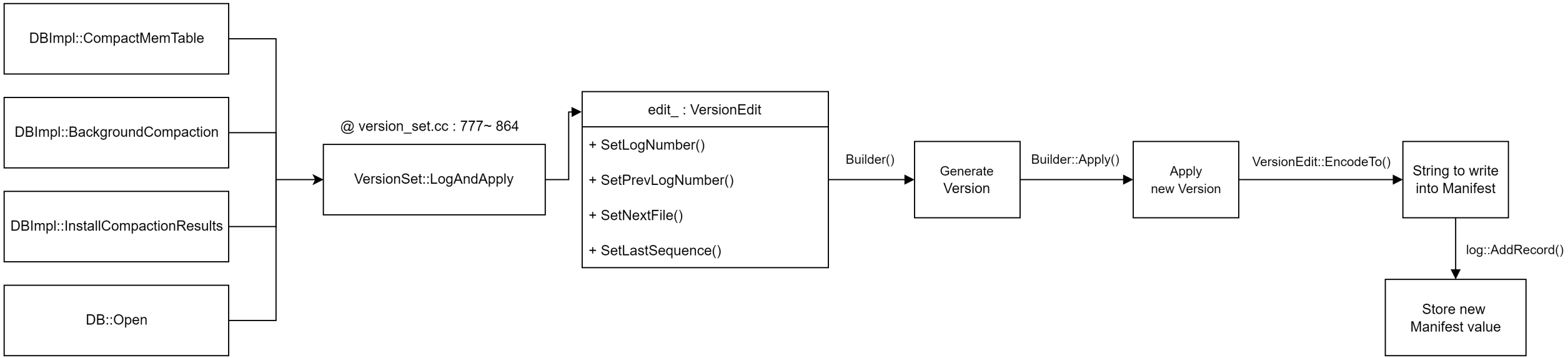
Values

Header

02 00	➡	0x02 = kLogNumber / Current log number = 0x00 (Default)
03 02	➡	0x03 = kNextFileNumber / Next file number is 0x02 (Default)
04 00	➡	0x04 is kLastSequence / Last sequence was 0x00 (Default)

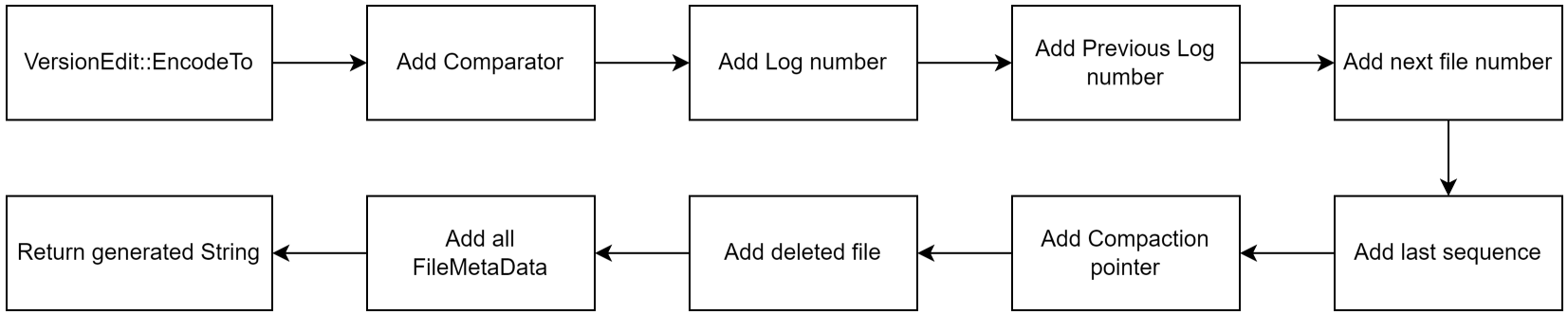
VersionSet::LogAndApply

- LogAndApply is for version control.
- Is called in following member functions.
- Stores data to MANIFEST file.
- DBImpl::CompactMemtable, DBImpl::BackgroundCompaction, DBIMpl::InstallCompactionResults, DB::Open calls this



VersionEdit::EncodeTo

- Generates actual data to write into MANIFEST file from VersionSet and FileMetaData
- By the generated string, the MANIFEST file is written.

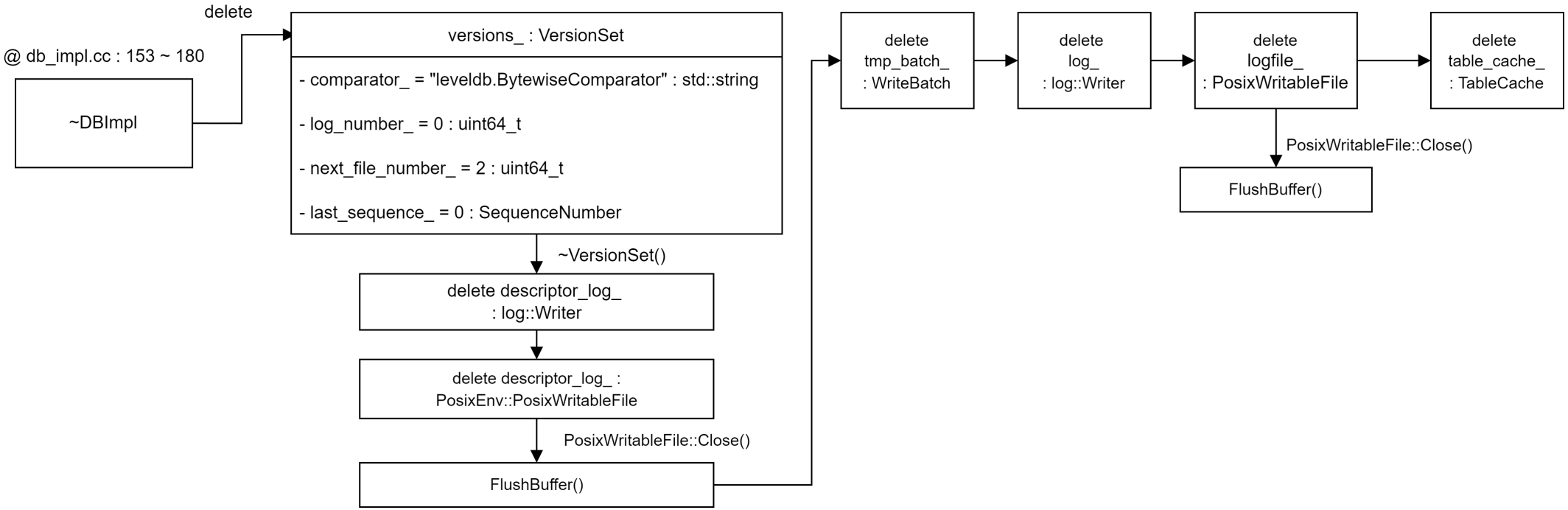



```
[+] LogNumber : 6
[+] Previous Log number : 0
[+] Next file : 7
[+] Last Sequence : 10
[+] EncodeTo was called!
[+] DST VALUE (LogNumber): 02 06
[+] DST VALUE (PrevLogNumber): 02 06 09 00
[+] DST VALUE (NextFileNumber): 02 06 09 00 03 07
[+] DST VALUE (LastSequence): 02 06 09 00 03 07 04 0a
[+] DST VALUE (kNewFile): 02 06 09 00 03 07 04 0a 07
[+] DST VALUE (Level): 02 06 09 00 03 07 04 0a 07 00
[+] DST VALUE (FileMetaData number): 02 06 09 00 03 07 04 0a 07 00 05
[+] DST VALUE (FileMetadata file size): 02 06 09 00 03 07 04 0a 07 00 05 ef 05
[+] DST VALUE (FileMetadata smallest): 02 06 09 00 03 07 04 0a 07 00 05 ef 05 18 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 01 01 00 00 00 00 00 00
[+] DST VALUE (FileMetadata largest): 02 06 09 00 03 07 04 0a 07 00 05 ef 05 18 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 01 01 00 00 00 00 00 00 18 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 39 01 0a 00 00 00 00 00 00
[+] Record value: new record to MANIFEST: 02 06 09 00 03 07 04 0a 07 00 05 ffffffef 05 18 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 01 01 00 00 00 00 00 00 18 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 39 01 0a 00 00 00 00 00
```

```
0020h: 74 6F 72 CE 6F F5 52 3F 00 01 02 06 09 00 03 07  torÎoõR?........
0030h: 04 0A 07 00 05 EF 05 18 30 30 30 30 30 30 30 30  .....ï..00000000
0040h: 30 30 30 30 30 30 30 30 01 01 00 00 00 00 00 00  00000000........
0050h: 18 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  .000000000000000
0060h: 39 01 0A 00 00 00 00 00 00                       9........
```

Closing DB

- ~DBImpl is called when closing DB (delete)



Closing DB

- Let's see how manifest file is written with fillseq, num == 10
- Similar to previous file.

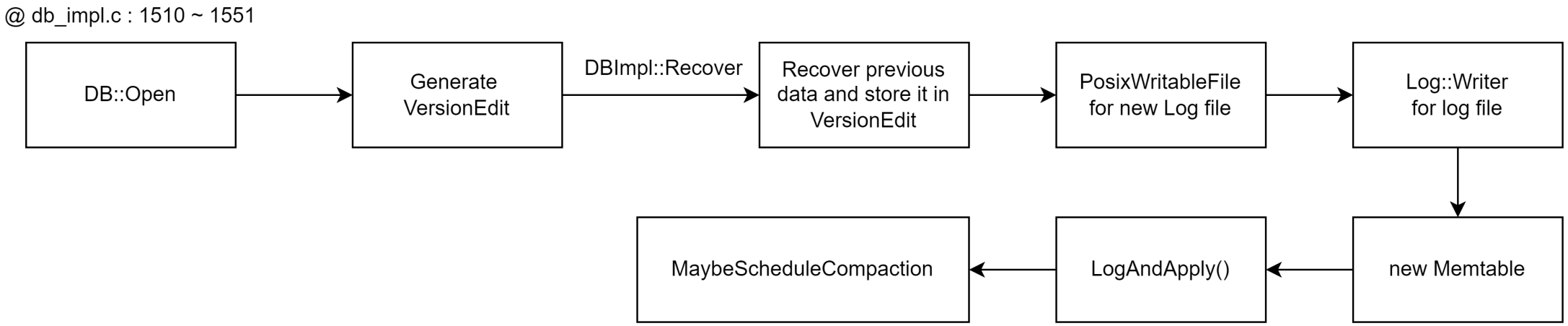
```
0020h: 74 6F 72 A4 9C 8B BE 08 00 01 02 03 09 00 03 04  tor¤œ‹¾.........
0030h: 04 00                                            ..
```

02 03	➡	0x02 is kLogNumber & Current log number is 0x03 (Current .log file's number is 000003)
09 00	➡	0x09 is kPrevLogNumber & Previous log number is 0x00
03 04	➡	0x03 is kNextFileNumber & Next file number is 0x04 (When next MANIFEST file is generated, the number will be 4)
04 00	➡	0x04 is kLastSequence & Last sequence was 0x00 (This will be updated in next LogAndApply)

```
[+] LogNumber : 3
[+] Previous Log number : 0
[+] Next file : 4
[+] Last Sequence : 0
[+] EncodeTo was called!
[+] DST VALUE (LogNumber): 02 03
[+] DST VALUE (PrevLogNumber): 02 03 09 00
[+] DST VALUE (NextFileNumber): 02 03 09 00 03 04
[+] DST VALUE (LastSequence): 02 03 09 00 03 04 04 00
[+] Record value: new record to MANIFEST: 02 03 09 00 03 04 04 00
```

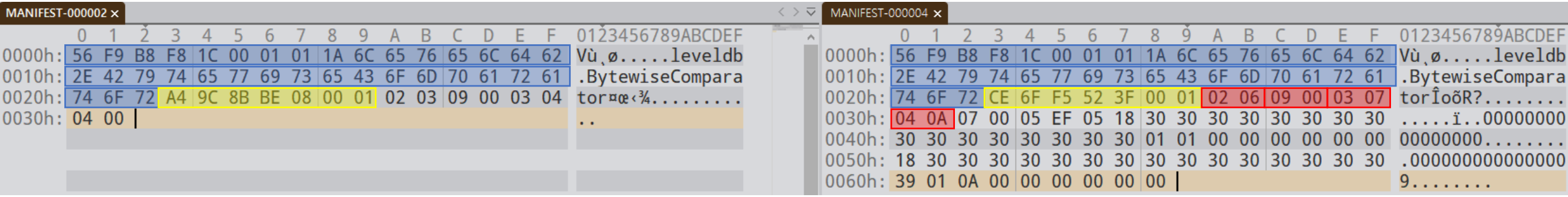
Opening DB

- Let’s see how manifest file is written with fillseq, num == 10, use_existing_db=1
- Opening DB calls LogAndApply!
- Thus, MANIFEST will be changed.



Opening DB - Manifest

- MANIFEST changed
- CURRENT was set to MANIFEST-000004



Before

After

Values

Header

02 06	➡	0x02 is kLogNumber & Current log number is 0x06 (00006.log is the WAL file)
09 00	➡	0x09 is kPrevLogNumber & Previous log number is 0x00
03 04	➡	0x03 is kNextFileNumber & Next file number is 0x07 (When next MANIFEST file is generated, the number will be 7)
04 00	➡	0x04 is kLastSequence & Last sequence was 0x0A (Yes, we had 10 entries that were put)

Opening DB - Manifest

- MANIFEST changed
- CURRENT was set to MANIFEST-000004

MANIFEST-000002

```
        0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
0000h: 56 F9 B8 F8 1C 00 01 01 1A 6C 65 76 65 6C 64 62 Vù¸ø.....leveldb
0010h: 2E 42 79 74 65 77 69 73 65 43 6F 6D 70 61 72 61 .BytewiseCompara
0020h: 74 6F 72 A4 9C 8B BE 08 00 01 02 03 09 00 03 04 tor¤œ‹¾.........
0030h: 04 00                                           ..
```

MANIFEST-000004

```
        0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
0000h: 56 F9 B8 F8 1C 00 01 01 1A 6C 65 76 65 6C 64 62 Vù¸ø.....leveldb
0010h: 2E 42 79 74 65 77 69 73 65 43 6F 6D 70 61 72 61 .BytewiseCompara
0020h: 74 6F 72 CE 6F F5 52 3F 00 01 02 06 09 00 03 07 torÎoõR?........
0030h: 04 0A 07 00 05 EF 05 18 30 30 30 30 30 30 30 30 .....ï..00000000
0040h: 30 30 30 30 30 30 30 30 01 01 00 00 00 00 00 00 00000000........
0050h: 18 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 .0000000000000000
0060h: 39 01 0A 00 00 00 00 00 00                      9........
```

Header

07	➡	0x07 is kNewFile
00	➡	0x00 is the current level of LevelDB (We have total 10 entries of 116 bytes, which resides in level 0)
05	➡	0x05 is FileMetaData's number
04 00	➡	0x04 is kLastSequence & Last sequence was 0x0A (Yes, we had 10 entries that were put)
EF 05	➡	0xef05 is the size of the file.
18 ~ 00	➡	The smallest FileMetaData's key value encoded. (Min key value was 0000000000000000) since this was fillseq
18 ~ 00	➡	The largest FileMetaData's key value encoded. (Max key value was 0000000000000009) since this was fillseq

Conclusion

- LogAndApply, EncodeTo takes care of version control.
- These functions generate MANIFEST file.
- Research how following functions work with MANIFEST file.
  - CompactMemTable
  - BackgroundCompaction
  - InstallCompactionResults

Final Conclusion

- Regrets on too much focus on WAL not Version & MANIFEST.
- Couldn’t see the foreset for the trees