













r/technicalminecraft

Posts

FAQ Getting Started ▼

Technical Wiki



Posted by u/bdm68 Chunk Loader 2 years ago



A new save file format for Java edition 1.15 has been decoded (MCC files)

1.15 (Java) introduces the first fundamental change to save file formats since 1.2 introduced the Anvil format. The new save file has an MCC extension and saves individual chunks that are too large to be saved with other chunks in the MCA file. This was introduced as a fix for the bug described in **MC-140507** (private issue).

NBT Explorer does not recognise this new file format and will ignore these files. The following is from an investigation using a hex editor and my own NBT codebase.

Anyone writing external tools such as mappers will need to be aware of this new file format. Current software may read these chunks as empty, may throw errors or may crash.

Primer for MCA file format

Chunk files in Java edition are saved in clusters of 4096 bytes. The first cluster of an MCA file has the cluster offsets for the chunks and the number of clusters for the chunk. Each chunk offset is saved as a 32-bit integer. The first 3 bytes are the offset in big endian format (high byte is first) and the other byte is the number of clusters used to save the chunk.

The second cluster has the modification times for each chunk as 32-bit integers.

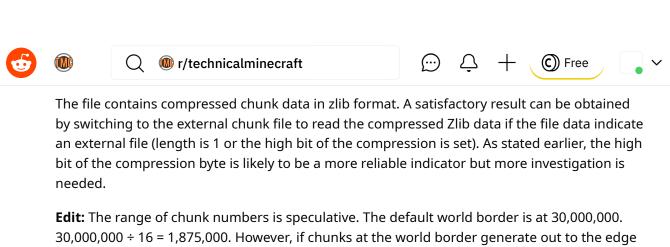
Chunk data starts at the third chunk. The first four bytes for a chunk are the number of bytes of file data in big endian format. The fifth byte is the compression algorithm used. This will usually be **02** to indicate Zlib. The chunk data follows the compression byte.

The new MCC file format

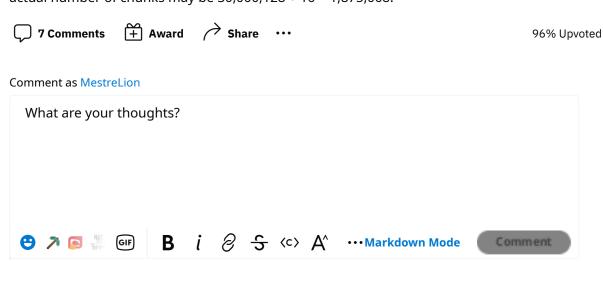
The MCC file format has been introduced to save chunks that are too large for MCA files.

The possible presence of this file can be inferred from a length of 1 in the MCA file headers. This means only one byte is saved in the MCA file. Reading this byte will give the compression. Normal Zlib files are indicated with a compression of **02**. For MCC files, this compression byte has a value of **82** hex (130 decimal). I assume that the **2** means Zlib compression is used (the same as other recent MCA files) and the **8** (high bit set) means the data is saved externally. When determining if a chunk file exists, the high bit being set is probably a more reliable indicator than the length being equal to 1.

The files containing individual chunks will be saved with the region files and have a name that matches the pattern "c.x.z.mcc" where x is the chunk x coordinate and z is the chunk z



of the region file, then a few chunks may be saved beyond this limit. Region files are 32×32 chunks or 512×512 blocks. $30,000,000 \div 512 = 58,593.75$. If the full region is populated, the actual number of chunks may be $30,000,128 \div 16 = 1,875,008$.





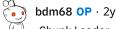


Hate_Feight · 2y

Did you find this?

Either way, this is an awesome discovery.



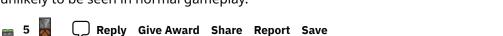


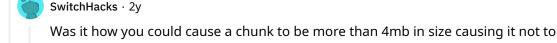
Chunk Loader

save?

Yes, I found this. I raised a private issue in Mojira that was a duplicate of MC-140507. I was testing it to see how the fix worked.

I won't provide details of the bug publicly because it is a private issue for a reason. It is unlikely to be seen in normal gameplay.

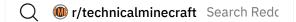




















r/technicalminecraft

Posts	FAQ Getting Started ▼ Technical Wiki
	Posted by u/bdm68 Chunk Loader 2 years ago
59	A new save file format for Java edition 1.15 has been decoded (MCC files)
	K Y
	7 Comments + Award > Share
	Sort By: Best ▼
,	View all comments
	bdm68 OP · 2y Chunk Loader
	Sample of an MCA file from a hex editor:

000000: 00 00 15 01 00 01 25 01 ...

Chunk at 0,0 is at location 15 hex (offset is 15000 hex) and has a length of 1.

015000: 00 00 00 01 82 78 9C ED ...

First four bytes (00 00 00 01) indicate a length of 1. The next byte 82 shows the compression and use of an external file. The remaining three bytes are spurious. They are the start of the original data that was in the file before the file was written to an external file. Minecraft does not overwrite the old data in the file if it shortens chunk data in the save file. These bytes turn out to be the first three bytes of ZLib compression.

By comparison, here is the data for the next chunk:

This chunk is at location 125 hex and has a length of 1.

125000: 00 00 01 EE 02 78 9C ED ...

This has a length of 1EE hex. The compression byte is 02 and the data follows. The next three bytes are identical to the previous example (78 9C ED) because this is the header for the ZLib compression. The length is quite short, only 1EE (494) bytes because this particular world is a superflat world.

