Compressed Custom Tag Store File Format Specification

Compressed Custom Tag Store (CCTS) files are meant to be an extremely efficient means of storing tag data from Graffiti on disk. These file sacrifice the external compatibility that something like a Property List or JSON file provides but allows for far less disk space used than a more general data format such as those. The specification is purposefully extremely simple as it is purpose built for this program to maximize compactness.

The file is modeled by the `TagStore` class and the relationship is meant to be one-to-one reversible. A single `TagStore` produces a file that is read back into an instance of `TagStore`. The file is a binary format composed of the following pieces

	Length	Description	Req'd	Semantic Meaning
	2 Bytes	2 LSB bytes of the patch of the version of `TagStore` that is being saved	YES	The version of the `TagStore` class that was saved to this file
	2 Bytes	2 LSB bytes of the minor of the version of `TagStore` that is being saved	YES	
	2 Bytes	2 LSB bytes of the major of the version of `TagStore` that is being saved	YES	
A	4 Bytes	Big Endian encoding of the Int number of all **unique** tags across all files in the saved `TagStore`	YES	The total number of tags across all files
	4 Bytes	Big Endian encoding for the length of the Serialized format of a Tag	NO	The length of the subsequent serialized Tag Data
	Variable	The bytes of a serialized Tag. The length is given by the preceding encoded Int	NO	A serialized Tag
B	4 Bytes	Big Endian encoding of the Int number of all files across all files in the saved `TagStore`	YES	The total number of files in the `TagStore`
	4 Bytes	Big Endian encoding for the length of the path of the file that was serialized	NO	The length of the serialized path to the given file
	Variable	The bytes of a serialized File Path. The length is given by the preceding encoded Int	NO	A serialized File path
C	4 Bytes	Big Endian encoding of the Int of all tags belonging to just the preceding file	NO	The total number of tags in the preceding file
	4 Bytes	Big Endian encoding for the length of the ID of a Tag that the preceding file was tagged with	NO	The length of the ID one of the preceding file's tags
	Variable	The bytes of a serialized Tag ID. The length is given by the preceding encoded Int	NO	The serialized ID of the tag

Notes about implementation:

- The data resulting from serialization is compressed using LZMA compression
- All integers are encoded using Big-Endian format
- Strings are all encoded using UTF-8
- The green box represents a single semantic unit: a `Tag` and is repeated a number of times. That number is given by the Int represented by A

- The blue box semantically represents a file and everything in it is repeated a number of times. That number is given by B
- The pink box semantically represents the ID of a `Tag*` that belongs to the file represented by its contained blue box. This pink box may also be repeated a number of times. This number is given by C
- Files in this format should be saved with the `.ccts` extension

**Note that the pink box represents only a* `Tag` *ID and does not contain full serializations of* `Tag`*s. The* `Tag`*s are serialized earlier in the file in the green box section.* `Tag` *instances are unique by ID so files all share a reference to the same* `Tag` *if they are tagged by the same data.*