Tupperware Binary Format

9-4-2001

The Tupperware Binary file is composed of 3 parts:
the Main Header, the String Table, and the Atom Data.

Main Header

The Main Header contains information used for determing if the file is valid as well as how much data to load for the other 2 parts. The structure of this header is as follows:

```
unsigned long m_magicNumber; // magic number to signify this is a tupperware binary file 0x45017629
float m_revNumber; // which revision the file is
unsigned long m_cacheCRC; // used for caching a tree to disk for later retrevial
int m_stringTableLength; // how many bytes the string table has in it
int m_atomLength; // how many bytes for the atoms (the tree)

int m_reserved[10]; // for future use
```

String Table

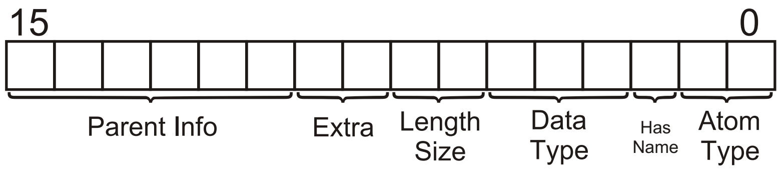
The String Table is a list of character strings separated by zeros. This string table is used for the key, name, and filename hint of the atoms. Each of these strings is referenced by the atoms by an unsigned short. Each value cooresponds to the string number within the string table. The first valid string would have an index of 1; the second would have an index of two and so on. The index of 0 is reserved for a null string.

Atom Data

The atom data contains one or more individual atoms. The amount each atom takes is determined by a header for each. This header is 4 bytes long and is described as follows:

```
unsigned short m_flags;
unsigned short m_keyIndex;
```

m_flags



Atom Type

The atom type is 2 bits in size and indicates what type of atom it is. Understanding some of the other fields depends on the atom type. Following is a list of valid values:

00 - Unknown
01 - Aggregate
10 - List
11 - Scalar

Has Name

Each atom may have a name string in addition to the key. If this atom has a name this field will be one and an additional 2 bytes will be following this header for the name.

Data Type

The data type is used for the scalar and list types to determine the type of data the atom has. Following is a list of valid values:

000 - Unknown
001 - String
010 - Float
011 - Int
100 - Binary (only with a scalar atom)

Length Size

The length size field indicates how many bytes are used to store the length following the header. This is used for list and scalar atom types.   If a list of ints was storing 150 integers, only one byte would be needed to store the length. If a binary scalar was storing 2000 bytes, two bytes would needed to store the length. Following is a list of valid values:

00 - 1 byte
01 - 2 bytes
10 - 3 bytes
11 - 4 bytes

Extra

The extra field is used for multiple purposes. In a binary scalar it is used to indicate if the atom has a file name hint string. This field is also used for packing integers within the file. This is used for int scalars as well as int lists. When an int is packed, a single byte can hold a value from -128 to 127.Following are valid values for the integer packing size:

00 - 1 byte
01 - 2 bytes
10 - 3 bytes
11 - 4 bytes

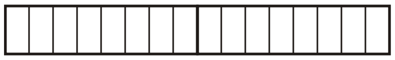
Parent Information

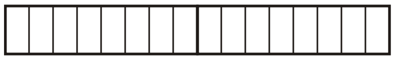
The parent information is used to place the atom within the atom hierarchy. Each aggregate is placed into a list of parents. The parent information is how deep the parent list is for the atom. The aggregate at this index is the parent for the atom. Any additional aggregates after this index are dropped. This will allow for nesting of aggregates to be at the most 63 levels deep since there are 6 bits used to represent this information.

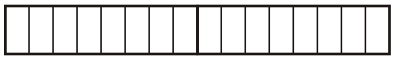
m_keyIndex

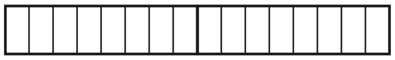
These 2 bytes of data indicates which string from the string table to use. 0 represents a null string. There is a limit of 65535 unique strings within the string table.

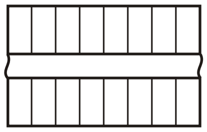
Atom Data Fields (What gets added based on what type of atom it is)

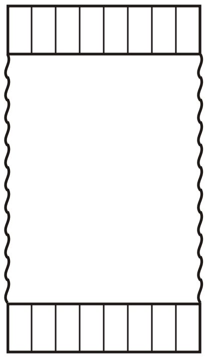

m_flags (all atoms)


m_keyIndex (all atoms)


Name Index (if has name is set)


Binary File Name Hint Index (if binary scalar and extra set)


Length 1-4 bytes (lists as well as string & binary scalars)
Number of bytes determined by Length Size field within m_flags


Additional data (added by atom type as well as data type)

String List Atoms add Length bytes
Int List Atoms add length * packed size bytes
Float List Atoms add length * 4 bytes
Int Scalars add packed size bytes
Float Scalars add 4 bytes
String Scalars add length bytes
Binary Scalars add length bytes