Polyglot BIN format

The file is made up of two sections: an optional header and then the positions.

Header (16 bytes per line)

Bytes	From	To	Notes	
8	1	8	Key	Fixed: [00] [00] [00] [00] [00] [00] [00] [00]
8	9	16	String	

All of the strings for Keys of zero value are added together giving one long header string. The string is divided into fields using the Line Feed character [0A]. Fields are:

1.	Fixed.	@PG@
2.	Version number as <major>.<minor>	1.0
3.	Number of extra fields	2
4.	(Extra 1) Number of variations supported	1
5.	(Extra 2) One variation name	normal
6.	Free format comments	Description, copyright etc

The free format comments are filled with nulls [00] at the end and do not have a final [0A]. So this example (using "\" to represent Line Feed) would appear as:

```
[00][00][00][00][00][00][00][00]@PG@\1.0
[00][00][00][00][00][00][00][00]\2\1\nor
[00][00][00][00][00][00][00][00]mal\Desc
[00][00][00][00][00][00][00][00]ription,
[00][00][00][00][00][00][00][00] copyrig
[00][00][00][00][00][00][00][00]t etc[00][00][00]
```

Positions (16 bytes per position)

Bytes	From	To	Notes	
8	1	8	Key	Calculated as a Zobrist hash key
2	9	10	Move	
2	11	12	Weight	
4	13	16	Learning data	

The move bytes are encoded as:

0	0	0	0	0	0	1	1
	Promotion			From Rank			Fr.

0	0	0	1	1	1	0	0
File		To Rank			To File		

File:	The file on the board	(0..7 meaning A..H)
Rank:	The rank on the board	(0..7 meaning 1..8)
Promotion:	Piece	(0 None, 1 Knight, 2 Bishop, 3 Rook, 4 Queen)

The example above shows [03] [1C] which is E2-E4.

Note that castling moves are encoded as E1H1, E1A1, E8H8 or E8A8 so need converting to the normal E1G1 etc.

Weight is an integer (in big endian format) but older polyglot programs might fail if its value is zero.

Learning data is used if Polyglot has `BookLearn=True` and is made up of a Count (2 bytes) and Total (2 bytes).

Zobrist hash keys

The Hash key is calculated by combining (using XOR) a series of random numbers. These are selected from a fixed list of 781 64-bit numbers, being made up of:

768 for piece/positions	(12 piece types times 64 squares)
4 for castling available	(white short, white long, black short, black long)
8 for en passant files	(A..H)
1 for White to move	

One number is used for each piece on the board, selected from the possible 768 numbers using the calculation:

(64 * PieceType) + (8 * rank) + file

where rank is 0..7 (representing '1' to '8') and file is 0..7 (representing 'A'..'H') and PieceType is:

black Pawn	0
white Pawn	1
black Knight	2
white Knight	3
black Bishop	4
white Bishop	5
black Rook	6
white Rook	7
black Queen	8
white Queen	9
black King	10
white King	11

So a white Pawn (type 1) on file 'E' (value 4) and rank '2' (value 1) would use random number:

(64 * 1) + (8 * 1) + 4 = 76

This will be XOR'd with all the other random numbers for the other pieces on the board.

The total is then XOR'd with each of the Castling random numbers that apply. These are based on whether each castling move is still available, rather than whether it is actually possible at the current move. So in the opening position all four castling moves are available even though none are actually possible at that time.

If the last move was a pawn move of two squares and that pawn now has an adjacent opposing pawn (or pawns) then the file of the moved pawn is used to select a random number from the 8 listed and this is also XOR'd with the current total.

Finally, if it is now white's turn to move then the additional White to move random number is XOR'd.

The result of all this XOR'ing is a single, hopefully unique, 64-bit number that represents the current board status.