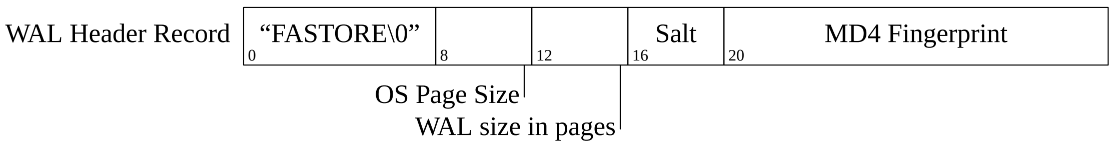
WAL Data Structures

File Header

Each WAL file is organized as succession of revision records. It starts with a header record to distinguish it from other files.



Recovery consists of reading the header, then reading records sequentially until the EOF marker.

The header record and sets of data records contain an MD4 checksum, giving the reader great protection against file corruption.

The first 8 bytes contain the **magic** string *FASTORE* with a trailing NULL. These serve to identify the file as a Fastore log file, and to denote the version of the file. For consistency with future versions the first four bytes should be considered constant (to identify the file type). The last four bytes are a version number; the final bit is zero for little-endian and one for big-endian.

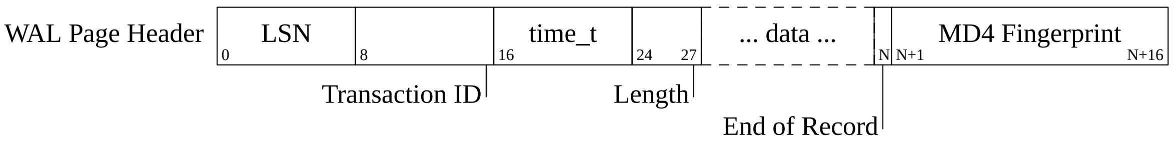
The **OS Page Size** serves tells the reader where to look for the other copy of the header record. It is also a factor in computing the *physical* file size, allowing **WAL size in pages** to describe a file up to 4 "gigapages", or 1 TB using a page size of 1 KB.

Salt is a random integer generated when the file is initialized.

The **MD4 fingerprint** is computed over the whole header. The random number helps to prevent any mixing of a header with the wrong file.

Record Header

Records are variable length without regard to OS page boundaries. A record may span two or more pages, or several records may appear on one page.



Each WAL record begins with a **Transaction ID**, followed by `time_t` as returned by `time(3)`, and a data **Length** in bytes. After the data is an **EOR** marker, with one of these values:

- 0x00, end of record, no MD4 follows
- 0x01, MD4 follows next
- 0xFF, End of File

To ensure data integrity, an **MD4 Fingerprint** is added

- to multipage transactions,
- once per page, for batches of small transactions, and

- after the last record.

This is done as follows:

- Write record, write MD4
- Note offset within page at end of MD4
- Compute size of next record
- If *offset + next record size ≤ OS page size* then set write pointer to overwrite last MD4
- Repeat until EOF

This algorithm ensures an MD4 is computed over most of a page or, for large records, over the entire record (and perhaps some small ones immediately preceding it). It avoids recording an MD4 for every tiny record, and avoids interpersing MD4s on page boundaries within records.[1]

The MD4 Fingerprint is computed over the bytes since the preceding MD4, even if that MD4 was in another WAL file (in which case it includes the new file's File Header Record).

[1] An alternative design would be to have one MD4 per page, and interrupt a record with an MD4 when writing across a page boundary. Such a design requires page-reading logic to splice together individually fingerprinted pages, whereas the chosen design permits the reader to simply read sequentially and compute an MD4 whenever the EOR indicates. The number of MD4 computations would be the same, because the MD4 would have to be updated whenever the page is updated.