

# Record Layer Design

This file provides some guidance on the thinking behind the design of the record layer code to aid future maintenance.

The record layer is divided into a number of components. At the time of writing there are four: SSL3\_RECORD, SSL3\_BUFFER, DTLS1\_BITMAP and RECORD\_LAYER. Each of these components is defined by:

1. A struct definition of the same name as the component
2. A set of source files that define the functions for that component
3. A set of accessor macros

All struct definitions are in record.h. The functions and macros are either defined in record.h or record\_local.h dependent on whether they are intended to be private to the record layer, or whether they form part of the API to the rest of libssl.

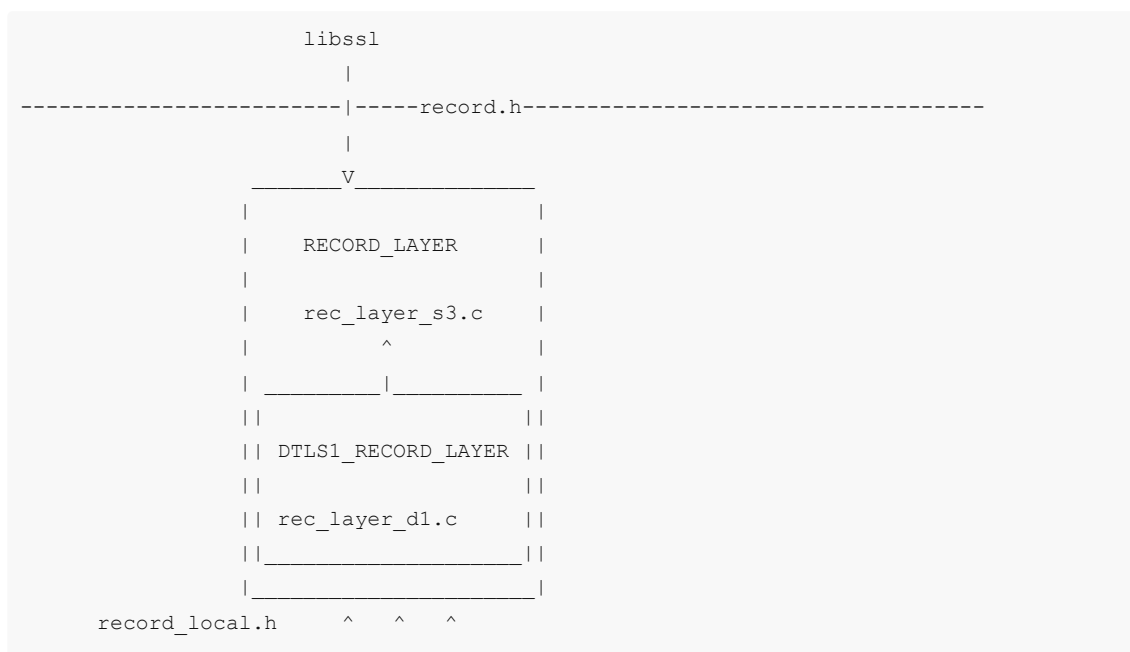
The source files map to components as follows:

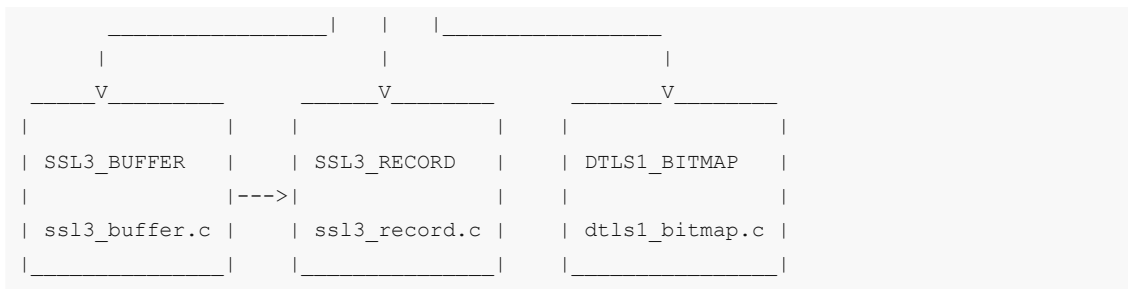
```
dtls1_bitmap.c          -> DTLS1_BITMAP component
ssl3_buffer.c           -> SSL3_BUFFER component
ssl3_record.c           -> SSL3_RECORD component
rec_layer_s3.c, rec_layer_d1.c -> RECORD_LAYER component
```

The RECORD\_LAYER component is a facade pattern, i.e. it provides a simplified interface to the record layer for the rest of libssl. The other 3 components are entirely private to the record layer and therefore should never be accessed directly by libssl.

Any component can directly access its own members - they are private to that component, e.g. ssl3\_buffer.c can access members of the SSL3\_BUFFER struct without using a macro. No component can directly access the members of another component, e.g. ssl3\_buffer cannot reach inside the RECORD\_LAYER component to directly access its members. Instead components use accessor macros, so if code in ssl3\_buffer.c wants to access the members of the RECORD\_LAYER it uses the RECORD\_LAYER\_\* macros.

Conceptually it looks like this:





The two RECORD\_LAYER source files build on each other, i.e. the main one is `rec_layer_s3.c` which provides the core SSL/TLS layer. The second one is `rec_layer_d1.c` which builds off of the SSL/TLS code to provide DTLS specific capabilities. It uses some DTLS specific RECORD\_LAYER component members which should only be accessed from `rec_layer_d1.c`. These are held in the `DTLS1_RECORD_LAYER` struct.