**This is the link to the Jaeger- cpp client**

<https://github.com/isaachier/jaeger-client-cpp>

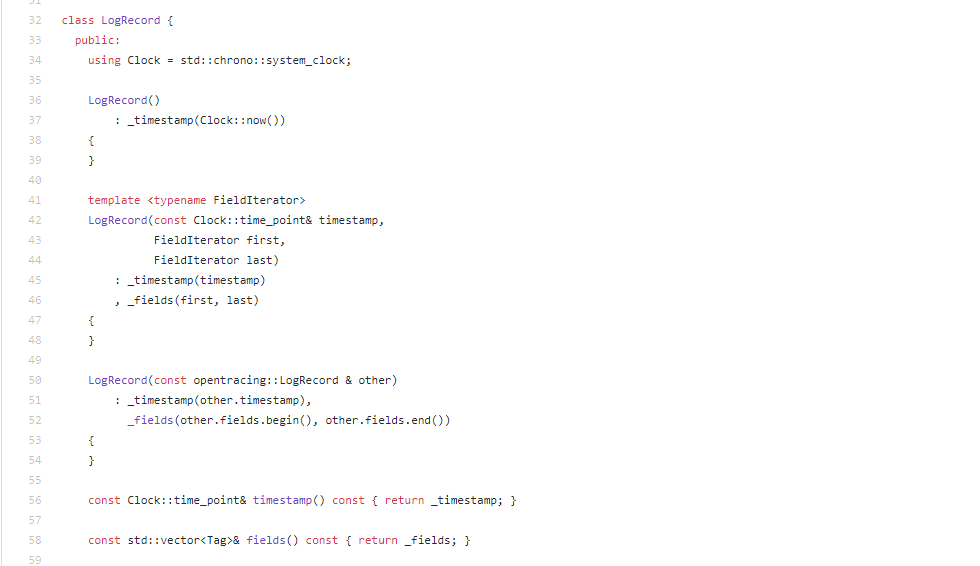
**Understanding and explaining functions**

***1 - Config.h***

#ifdef - directives to the preprocessor (not code) pre-compilation

YAML - not a markup language and is used in configuration file. Also used where data is stored and transmitted.

***2 - LogRecord.h***



Purpose of Log is to hold trace and span id (span needs to be sampled)

This piece of code defines different constructors. Different because they are being used for different purposes.

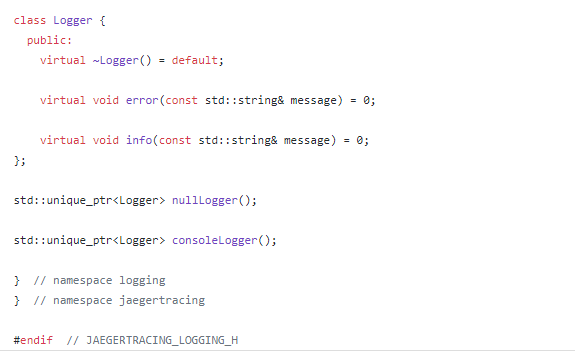
First Logrecord() - simply gives the current timestamp (since no parameter passed)

Second Logrecord() - used for a different purpose (since parameter passed)

|  |  |
| --- | --- |
| const (Clock::time\_point& timestamp, |  |
| FieldIterator first, |  |
| FieldIterator last)  This means, everything after scope operator belongs to class Clock. Therefore, there are multiple versions of logrecord in the code. |  |

This means that another thrift exists but this thrift is specific to this class. I think it has something to do with the logrecord timestamp.

***2 - Logging.h***



First step is a destructor which tells what events should happen when a variable of a class is being deleted

Next two lines mean, when creating variable of that class, we are setting them to zero.\

Next two lines after the class is defining the functions. Can use them later as nullLogger.h

***3 - Reference.h***



Defines a Type. Then defines a constructor.

Next, spancontext called without any parameter simply returns the spancontext.

Followed by switch case depending on whether it precedes or follows the reference.

Else, it throws an error.

***4 - Span.h***

the explicit keyword to the constructor prevents the compiler from using that constructor for implicit conversions



Defining and storing variables of class span

***5 - SpanContext.h***



**static\_cast**<type> (object); The type parameter must be a data type to which object can be converted via a known method, whether it be a builtin or a cast. The type can be a reference or an enumerator. All types of conversions that are well-defined and allowed by the compiler are performed using **static\_cast**

Swap is used to display zipkin UI.

Equivalent to setinfo in blkin. I.e it sets data

Key value annotations are used to define applications specific info in traces.