

# Intel® Cluster Checker Analysis API

Generated by Doxygen 1.6.1

Fri Jan 15 17:28:56 2016



# Contents

<b>1</b>	<b>Disclaimer and Legal Information</b>	<b>1</b>
<b>2</b>	<b>Todo List</b>	<b>3</b>
<b>3</b>	<b>Class Index</b>	<b>5</b>
3.1	Class Hierarchy . . . . .	5
<b>4</b>	<b>Class Index</b>	<b>7</b>
4.1	Class List . . . . .	7
<b>5</b>	<b>Class Documentation</b>	<b>9</b>
5.1	clk::Layer::Config Struct Reference . . . . .	9
5.1.1	Detailed Description . . . . .	10
5.1.2	Constructor & Destructor Documentation . . . . .	10
5.1.2.1	Config . . . . .	10
5.1.2.2	Config . . . . .	10
5.1.3	Member Data Documentation . . . . .	10
5.1.3.1	config_params . . . . .	10
5.1.3.2	db . . . . .	11
5.1.3.3	expiration . . . . .	11
5.1.3.4	extension_mods . . . . .	11
5.1.3.5	extension_path . . . . .	11
5.1.3.6	kb_mods . . . . .	11
5.1.3.7	kb_path . . . . .	11

5.1.3.8	language . . . . .	11
5.1.3.9	node_source . . . . .	11
5.1.3.10	nodes . . . . .	11
5.1.3.11	now . . . . .	12
5.2	clk::Layer::ConfigParam Struct Reference . . . . .	13
5.2.1	Detailed Description . . . . .	13
5.2.2	Member Data Documentation . . . . .	13
5.2.2.1	key . . . . .	13
5.2.2.2	module . . . . .	13
5.2.2.3	values . . . . .	13
5.3	clk::Database Class Reference . . . . .	14
5.3.1	Detailed Description . . . . .	14
5.3.2	Constructor & Destructor Documentation . . . . .	14
5.3.2.1	~Database . . . . .	14
5.4	clk::Diagnosis Class Reference . . . . .	15
5.4.1	Detailed Description . . . . .	15
5.5	clk::Fault Class Reference . . . . .	16
5.5.1	Detailed Description . . . . .	16
5.5.2	Member Data Documentation . . . . .	16
5.5.2.1	confidence . . . . .	16
5.5.2.2	id . . . . .	16
5.5.2.3	msg . . . . .	17
5.5.2.4	nodes . . . . .	17
5.5.2.5	remedy . . . . .	17
5.5.2.6	rowid . . . . .	17
5.5.2.7	severity . . . . .	17
5.5.2.8	suppressed . . . . .	17
5.6	clk::Layer::Filter Struct Reference . . . . .	18
5.6.1	Detailed Description . . . . .	18
5.6.2	Member Data Documentation . . . . .	18
5.6.2.1	confidence . . . . .	18

5.6.2.2	ids	18
5.6.2.3	nodes	18
5.6.2.4	severity	18
5.6.2.5	state	19
5.6.2.6	suppressed	19
5.6.2.7	type	19
5.7	clk::Layer Class Reference	20
5.7.1	Detailed Description	21
5.7.2	Constructor & Destructor Documentation	21
5.7.2.1	Layer	21
5.7.2.2	~Layer	21
5.7.3	Member Function Documentation	21
5.7.3.1	analyze	21
5.7.3.2	collect	21
5.7.3.3	get_faults	22
5.7.3.4	get_messages	22
5.7.3.5	get_nodes	22
5.7.3.6	get_version_number	22
5.7.3.7	progress	23
5.7.4	Member Data Documentation	23
5.7.4.1	message	23
5.8	clk::Layer::Message Struct Reference	24
5.8.1	Detailed Description	24
5.8.2	Member Data Documentation	24
5.8.2.1	level	24
5.8.2.2	msg	24
5.9	clk::Node Class Reference	25
5.9.1	Detailed Description	25
5.9.2	Member Enumeration Documentation	25
5.9.2.1	role_t	25
5.9.3	Member Data Documentation	25

5.9.3.1	name	25
5.9.3.2	roles	25
5.9.3.3	subcluster	26
5.10	clk::Sign Class Reference	27
5.10.1	Detailed Description	27
5.10.2	Member Data Documentation	27
5.10.2.1	state	27
5.11	clk::Layer::Sorting Struct Reference	28
5.11.1	Detailed Description	28
5.11.2	Constructor & Destructor Documentation	28
5.11.2.1	Sorting	28
5.11.3	Member Data Documentation	28
5.11.3.1	ascending	28
5.11.3.2	field	28
5.12	clk::SQLite Class Reference	29
5.12.1	Detailed Description	29
5.12.2	Constructor & Destructor Documentation	29
5.12.2.1	SQLite	29
5.12.2.2	~SQLite	29
5.12.3	Member Data Documentation	29
5.12.3.1	db_file	29
5.13	clk::Layer::Suppression Struct Reference	30
5.13.1	Detailed Description	30
5.13.2	Member Data Documentation	30
5.13.2.1	confidence	30
5.13.2.2	id	30
5.13.2.3	node	30
5.13.2.4	severity	30

# Chapter 1

## Disclaimer and Legal Information

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

A "Mission Critical Application" is any application in which failure of the Intel Product could result, directly or indirectly, in personal injury or death. SHOULD YOU PURCHASE OR USE INTEL'S PRODUCTS FOR ANY SUCH MISSION CRITICAL APPLICATION, YOU SHALL INDEMNIFY AND HOLD INTEL AND ITS SUBSIDIARIES, SUBCONTRACTORS AND AFFILIATES, AND THE DIRECTORS, OFFICERS, AND EMPLOYEES OF EACH, HARMLESS AGAINST ALL CLAIMS COSTS, DAMAGES, AND EXPENSES AND REASONABLE ATTORNEYS' FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PRODUCT LIABILITY, PERSONAL INJURY, OR DEATH ARISING IN ANY WAY OUT OF SUCH MISSION CRITICAL APPLICATION, WHETHER OR NOT INTEL OR ITS SUBCONTRACTOR WAS NEGLIGENT IN THE DESIGN, MANUFACTURE, OR WARNING OF THE INTEL PRODUCT OR ANY OF ITS PARTS.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising

from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Requires a system with a 64-bit enabled processor, chipset, BIOS and software. Performance will vary depending on the specific hardware and software you use. Consult your PC manufacturer for more information. For more information, visit <http://www.intel.com/info/em64t>

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to: <http://www.intel.com/design/literature.htm>

Intel, the Intel logo, the Intel Inside logo, Xeon, and Xeon Phi are trademarks of Intel Corporation in the U.S. and/or other countries.

Optimization Notice
---------------------

<p>Intel compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.</p>
--

<p>Notice revision #20110804</p>
----------------------------------

\* Other names and brands may be claimed as the property of others.

Copyright ©2016 Intel Corporation. All rights reserved.



## **Chapter 2**

### **Todo List**

**Class `clk::Diagnosis`** A new type field will be added to the Fault class. This class will be removed.

**Member `clk::Layer::collect()`** Placeholder - not implemented

**Class `clk::Layer::Config`** Additional configuration options are likely to be added in the future, as needed.

**Class `clk::Sign`** A new type field will be added to the Fault class. This class will be removed.

## Chapter 3

# Class Index

### 3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

clk::Layer::Config . . . . .	9
clk::Layer::ConfigParam . . . . .	13
clk::Database . . . . .	14
clk::SQLite . . . . .	29
clk::Fault . . . . .	16
clk::Diagnosis . . . . .	15
clk::Sign . . . . .	27
clk::Layer::Filter . . . . .	18
clk::Layer . . . . .	20
clk::Layer::Message . . . . .	24
clk::Node . . . . .	25
clk::Layer::Sorting . . . . .	28
clk::Layer::Suppression . . . . .	30



## Chapter 4

# Class Index

### 4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">clk::Layer::Config</a> ( <a href="#">Layer</a> configuration options ) . . . . .	9
<a href="#">clk::Layer::ConfigParam</a> (Data analysis configuration parameter ) . . . . .	13
<a href="#">clk::Database</a> (Base class for database configuration ) . . . . .	14
<a href="#">clk::Diagnosis</a> (A diagnosis is the root cause of an issue. <a href="#">Diagnosis</a> is derived from the <a href="#">Fault</a> class ) . . . . .	15
<a href="#">clk::Fault</a> (A fault is the basic analysis unit. A fault is either a sign (i.e., observation) or a diagnosis (i.e., root cause) ) . . . . .	16
<a href="#">clk::Layer::Filter</a> ( <a href="#">Filter</a> for the list of faults returned by <a href="#">get_faults()</a> ) . . . .	18
<a href="#">clk::Layer</a> (The presentation layer ) . . . . .	20
<a href="#">clk::Layer::Message</a> (Internal <a href="#">Layer</a> messages for the caller to handle ) . . .	24
<a href="#">clk::Node</a> (The node container ) . . . . .	25
<a href="#">clk::Sign</a> (A sign is an observation of an issue. <a href="#">Sign</a> is derived from the <a href="#">Fault</a> class ) . . . . .	27
<a href="#">clk::Layer::Sorting</a> (Sort order for the list of faults returned by <a href="#">get_faults()</a> )	28
<a href="#">clk::SQLite</a> ( <a href="#">SQLite</a> configuration. Derived from <a href="#">Database</a> ) . . . . .	29
<a href="#">clk::Layer::Suppression</a> (Suppress faults matching the specified values ) . .	30



## Chapter 5

# Class Documentation

### 5.1 clk::Layer::Config Struct Reference

[Layer](#) configuration options.

```
#include <clk.h>
```

#### Public Types

- enum { DATABASE, NODELIST, INTERSECTION, UNION }

#### Public Member Functions

- [Config](#) ()
- [Config](#) (std::shared\_ptr< [Database](#) > db, const std::vector< std::string > &extension\_mods, const std::vector< std::string > &kb\_mods, const std::string &extension\_path="", const std::string &kb\_path="")

#### Public Attributes

- std::shared\_ptr< [Database](#) > db
- time\_t [expiration](#) = 0
- std::vector< std::string > [extension\\_mods](#)
- std::string [extension\\_path](#)
- std::vector< std::string > [kb\\_mods](#)
- std::string [kb\\_path](#)
- std::string [language](#)

- `std::vector< Node > nodes`
- `enum clk::Layer::Config:: { ... } node\_source`
- `time_t now = time(NULL)`
- `std::vector< ConfigParam > config\_params`

### 5.1.1 Detailed Description

[Layer](#) configuration options.

#### [Todo](#)

Additional configuration options are likely to be added in the future, as needed.

### 5.1.2 Constructor & Destructor Documentation

#### 5.1.2.1 `clk::Layer::Config::Config ()`

Default [Config](#) constructor

**5.1.2.2** `clk::Layer::Config::Config (std::shared_ptr< Database > db, const std::vector< std::string > & extension_mods, const std::vector< std::string > & kb_mods, const std::string & extension_path = "", const std::string & kb_path = "")`

[Config](#) constructor

#### Parameters:

*db* Instance of class derived from [Database](#)

*extension\_mods* List of connector extensions to be loaded

*kb\_mods* List of knowledge base files to be loaded

*extension\_path* Absolute path to the connector extension directory

*kb\_path* Absolute path to the knowledge base directory

### 5.1.3 Member Data Documentation

#### 5.1.3.1 `std::vector<ConfigParam> clk::Layer::Config::config_params`

Data analysis configuration parameters



### 5.1.3.2 `std::shared_ptr<Database> clk::Layer::Config::db`

Instance of class derived from [Database](#)

### 5.1.3.3 `time_t clk::Layer::Config::expiration = 0`

Maximum allowable age of data, relative to the value of now. Data older than now minus expiration will be ignored. A value of 0 means to use all data, i.e., there is no expiration.

### 5.1.3.4 `std::vector<std::string> clk::Layer::Config::extension_mods`

List of connector extensions to be loaded

### 5.1.3.5 `std::string clk::Layer::Config::extension_path`

Absolute path to the connector extension directory

### 5.1.3.6 `std::vector<std::string> clk::Layer::Config::kb_mods`

List of knowledge base files to be loaded

### 5.1.3.7 `std::string clk::Layer::Config::kb_path`

Absolute path to the knowledge base directory

### 5.1.3.8 `std::string clk::Layer::Config::language`

String representing the message catalog language

### 5.1.3.9 `enum { ... } clk::Layer::Config::node_source`

Select the source for the list of nodes. Options are to use the database (exclusively), the input [nodes](#) vector (exclusively), or the intersection or union of the two.

### 5.1.3.10 `std::vector<Node> clk::Layer::Config::nodes`

List of nodes to be analyzed (see [node\\_source](#)).

**5.1.3.11    `time_t clk::Layer::Config::now = time(NULL)`**

Reference time to be used as the current time for analysis

## 5.2 clk::Layer::ConfigParam Struct Reference

Data analysis configuration parameter.

```
#include <clk.h>
```

### Public Attributes

- std::string [module](#)
- std::string [key](#)
- std::vector< std::string > [values](#)

#### 5.2.1 Detailed Description

Data analysis configuration parameter.

#### 5.2.2 Member Data Documentation

##### 5.2.2.1 std::string clk::Layer::ConfigParam::key

Parameter name

##### 5.2.2.2 std::string clk::Layer::ConfigParam::module

Module name in the knowledge base

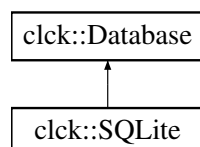
##### 5.2.2.3 std::vector<std::string> clk::Layer::ConfigParam::values

Parameter value(s)

## 5.3 clk::Database Class Reference

Base class for database configuration.

#include <clk.h>Inheritance diagram for clk::Database::



### Public Member Functions

- virtual [~Database](#) ()=0

#### 5.3.1 Detailed Description

Base class for database configuration.

#### 5.3.2 Constructor & Destructor Documentation

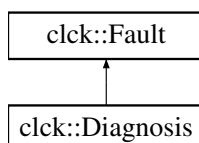
##### 5.3.2.1 virtual clk::Database::~~Database () [pure virtual]

[Database](#) destructor

## 5.4 clk::Diagnosis Class Reference

A diagnosis is the root cause of an issue. [Diagnosis](#) is derived from the [Fault](#) class.

`#include <clk.h>`Inheritance diagram for `clk::Diagnosis`:



### 5.4.1 Detailed Description

A diagnosis is the root cause of an issue. [Diagnosis](#) is derived from the [Fault](#) class.

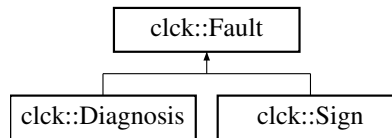
#### [Todo](#)

A new type field will be added to the [Fault](#) class. This class will be removed.

## 5.5 clk::Fault Class Reference

A fault is the basic analysis unit. A fault is either a sign (i.e., observation) or a diagnosis (i.e., root cause).

#include <clk.h> Inheritance diagram for clk::Fault:



### Public Attributes

- int [confidence](#) = 0
- std::string [id](#)
- std::string [msg](#)
- std::vector< std::string > [nodes](#)
- std::string [remedy](#)
- int [severity](#) = 0
- bool [suppressed](#) = false
- std::set< int > [rowid](#)

### 5.5.1 Detailed Description

A fault is the basic analysis unit. A fault is either a sign (i.e., observation) or a diagnosis (i.e., root cause).

### 5.5.2 Member Data Documentation

#### 5.5.2.1 int clk::Fault::confidence = 0

Confidence percentage (0 - 100)

#### 5.5.2.2 std::string clk::Fault::id

Message catalog id

**5.5.2.3 std::string clk::Fault::msg**

Expanded message string

**5.5.2.4 std::vector<std::string> clk::Fault::nodes**

List of nodes

**5.5.2.5 std::string clk::Fault::remedy**

Expanded remedy string

**5.5.2.6 std::set<int> clk::Fault::rowid**

DB rows that provide the raw data upon which the sign/diagnosis is based

**5.5.2.7 int clk::Fault::severity = 0**

Severity percentage (0 - 100)

**5.5.2.8 bool clk::Fault::suppressed = false**

True if the diagnosis / sign is suppressed, false otherwise

## 5.6 clk::Layer::Filter Struct Reference

[Filter](#) for the list of faults returned by [get\\_faults\(\)](#).

```
#include <clk.h>
```

### Public Attributes

- int [confidence](#) = 0
- std::vector< std::string > [ids](#)
- std::vector< std::string > [nodes](#)
- int [severity](#) = 0
- std::bitset< 2 > [state](#) = CLK\_FAULT\_STATE\_DIAGNOSED | CLK\_FAULT\_STATE\_OBSERVED
- std::bitset< 2 > [suppressed](#) = CLK\_FAULT\_SUPPRESSED\_FALSE
- std::bitset< 2 > [type](#) = CLK\_FAULT\_TYPE\_DIAGNOSIS | CLK\_FAULT\_TYPE\_SIGN

### 5.6.1 Detailed Description

[Filter](#) for the list of faults returned by [get\\_faults\(\)](#).

### 5.6.2 Member Data Documentation

#### 5.6.2.1 int clk::Layer::Filter::confidence = 0

Select faults with a greater than or equal to confidence value.

#### 5.6.2.2 std::vector<std::string> clk::Layer::Filter::ids

Select faults corresponding to at least one of the ids. If empty, does not filter on id.

#### 5.6.2.3 std::vector<std::string> clk::Layer::Filter::nodes

Selects faults corresponding to at least one of the nodes. If empty, does not filter on node.

#### 5.6.2.4 int clk::Layer::Filter::severity = 0

Select faults with a greater than or equal to severity value.



**5.6.2.5 std::bitset<2> clk::Layer::Filter::state = CLCK\_FAULT\_STATE\_-  
DIAGNOSED | CLCK\_FAULT\_STATE\_OBSERVED**

Select faults with a matching state. Only applies to signs, not diagnoses.

**5.6.2.6 std::bitset<2> clk::Layer::Filter::suppressed =  
CLCK\_FAULT\_SUPPRESSED\_FALSE**

Select faults with a matching suppression value.

**5.6.2.7 std::bitset<2> clk::Layer::Filter::type =  
CLCK\_FAULT\_TYPE\_DIAGNOSIS | CLCK\_FAULT\_TYPE\_SIGN**

Select faults with a matching type.

## 5.7 clk::Layer Class Reference

The presentation layer.

```
#include <clk.h>
```

### Classes

- struct [Config](#)  
*Layer configuration options.*
- struct [ConfigParam](#)  
*Data analysis configuration parameter.*
- struct [Filter](#)  
*Filter for the list of faults returned by [get\\_faults\(\)](#).*
- struct [Message](#)  
*Internal [Layer](#) messages for the caller to handle.*
- struct [Sorting](#)  
*Sort order for the list of faults returned by [get\\_faults\(\)](#).*
- struct [Suppression](#)  
*Suppress faults matching the specified values.*

### Public Member Functions

- [Layer](#) (const [Config](#) &config)
- [~Layer](#) ()
- bool [analyze](#) (const std::vector< [Suppression](#) > &suppressions)
- bool [collect](#) ()
- std::vector< std::shared\_ptr< [Fault](#) > > [get\\_faults](#) (const [Filter](#) &filter, const std::vector< [Sorting](#) > &sorting)
- std::vector< [Message](#) > [get\\_messages](#) ()
- std::vector< [Node](#) > [get\\_nodes](#) ()
- int [get\\_version\\_number](#) ()
- bool [progress](#) (unsigned long &remaining, unsigned long &completed)

### Public Attributes

- std::condition\_variable [message](#)

### 5.7.1 Detailed Description

The presentation layer.

### 5.7.2 Constructor & Destructor Documentation

#### 5.7.2.1 clk::Layer::Layer (const Config & *config*)

[Layer](#) constructor

#### 5.7.2.2 clk::Layer::~~Layer ()

[Layer](#) destructor

### 5.7.3 Member Function Documentation

#### 5.7.3.1 bool clk::Layer::analyze (const std::vector< Suppression > & *suppressions*)

Start the analysis. Note: behavior is undefined if invoked more than once per [Layer](#) instance.

**Returns:**

True if the analysis was completed successfully, false otherwise. Note: this does NOT reflect the state of the cluster.

#### 5.7.3.2 bool clk::Layer::collect () **[inline]**

Collect data.

**Todo**

Placeholder - not implemented

**Returns:**

True if the data collection was completed successfully, false otherwise.

### 5.7.3.3 `std::vector<std::shared_ptr<Fault> > clk::Layer::get_faults (const Filter &filter, const std::vector< Sorting > &sorting)`

Returns a list of faults. May be called multiple times. While [analyze\(\)](#) is active, presumably in another thread, the behavior is undefined. Use [progress\(\)](#) to determine current analysis status. If called before [analyze\(\)](#), returns an empty list.

#### Parameters:

*filter* Return only faults that match the filter

*sorting* List of sorting criteria. The first element is the primary sorting criterion, the second element the secondary sorting criterion, etc. If empty, the faults are returned unsorted.

#### Returns:

A list of faults

### 5.7.3.4 `std::vector<Message> clk::Layer::get_messages ()`

Returns a list of messages generated internal to [Layer](#) for the caller to handle.

#### Returns:

A list of messages

### 5.7.3.5 `std::vector<Node> clk::Layer::get_nodes ()`

Returns the list of nodes to be analyzed.

#### Returns:

A list of nodes.

### 5.7.3.6 `int clk::Layer::get_version_number ()`

Returns the version of the [Layer](#) API.

#### Returns:

The version number. Version X.Y.Z is represented as (X\*1000000 + Y\*1000 + Z).

### 5.7.3.7 bool clk::Layer::progress (unsigned long & *remaining*, unsigned long & *completed*)

While [analyze\(\)](#) is active, presumably in another thread, returns the number of rules remaining to be fired and the number of rules already run. If called before [analyze\(\)](#), both values will be 0.

#### Parameters:

*remaining* The number of rules remaining to be fired. Not guaranteed to be monotonic. Returned by reference.

*completed* The number of rules that have been fired. Will be monotonic. Returned by reference.

#### Returns:

False if [analyze\(\)](#) has not yet been called, true otherwise.

## 5.7.4 Member Data Documentation

### 5.7.4.1 std::condition\_variable clk::Layer::message

Notify when a new internal [Layer](#) message is available

## 5.8 clk::Layer::Message Struct Reference

Internal [Layer](#) messages for the caller to handle.

```
#include <clk.h>
```

### Public Attributes

- int [level](#)
- std::string [msg](#)

#### 5.8.1 Detailed Description

Internal [Layer](#) messages for the caller to handle.

#### 5.8.2 Member Data Documentation

##### 5.8.2.1 int clk::Layer::Message::level

[Message](#) level (priority). Inherits logging levels from syslog.

##### 5.8.2.2 std::string clk::Layer::Message::msg

[Message](#) string

## 5.9 clk::Node Class Reference

The node container.

```
#include <clk.h>
```

### Public Types

- enum [role\\_t](#) {  
    ROLE\_BOOT,   ROLE\_COMPUTE,   ROLE\_ENHANCED,   ROLE\_-  
    EXTERNAL,  
    ROLE\_HEAD,   ROLE\_JOB\_SCHEDULE,   ROLE\_LOGIN,   ROLE\_-  
    NETWORK\_ADDRESS,  
    ROLE\_STORAGE }

### Public Attributes

- std::string [subcluster](#)
- std::string [name](#)
- std::vector< [role\\_t](#) > [roles](#)

### 5.9.1 Detailed Description

The node container.

### 5.9.2 Member Enumeration Documentation

#### 5.9.2.1 enum clk::Node::role\_t

Roles than a node can fulfill.

### 5.9.3 Member Data Documentation

#### 5.9.3.1 std::string clk::Node::name

An unique node identifier, i.e., the hostname.

#### 5.9.3.2 std::vector<role\_t> clk::Node::roles

A list of roles that the node fulfills.

### 5.9.3.3 `std::string clk::Node::subcluster`

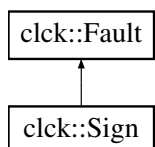
A subcluster to which the node belongs.



## 5.10 clk::Sign Class Reference

A sign is an observation of an issue. [Sign](#) is derived from the [Fault](#) class.

`#include <clk.h>`Inheritance diagram for `clk::Sign`:



### Public Types

- enum { **DIAGNOSED**, **OBSERVED** }

### Public Attributes

- enum `clk::Sign::` { ... } [state](#)

#### 5.10.1 Detailed Description

A sign is an observation of an issue. [Sign](#) is derived from the [Fault](#) class.

##### [Todo](#)

A new type field will be added to the [Fault](#) class. This class will be removed.

#### 5.10.2 Member Data Documentation

##### 5.10.2.1 enum { ... } `clk::Sign::state`

Either diagnosed (meaning used to make a diagnosis) or observed (undiagnosed).

## 5.11 clk::Layer::Sorting Struct Reference

Sort order for the list of faults returned by [get\\_faults\(\)](#).

```
#include <clk.h>
```

### Public Types

- enum { **CONFIDENCE**, **ID**, **NODE**, **SEVERITY** }

### Public Member Functions

- [Sorting](#) (bool [ascending](#), decltype([field](#)) [field](#))

### Public Attributes

- bool [ascending](#) = true
- enum clk::Layer::Sorting:: { ... } [field](#)

#### 5.11.1 Detailed Description

Sort order for the list of faults returned by [get\\_faults\(\)](#).

#### 5.11.2 Constructor & Destructor Documentation

##### 5.11.2.1 clk::Layer::Sorting::Sorting (bool *ascending*, decltype(*field*) *field*)

[Sorting](#) constructor

#### 5.11.3 Member Data Documentation

##### 5.11.3.1 bool clk::Layer::Sorting::ascending = true

If true, sort in ascending order, otherwise sort in descending order.

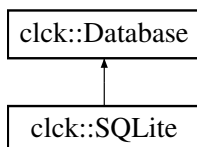
##### 5.11.3.2 enum { ... } clk::Layer::Sorting::field

[Fault](#) field to sort on

## 5.12 clk::SQLite Class Reference

[SQLite](#) configuration. Derived from [Database](#).

`#include <clk.h>`Inheritance diagram for `clk::SQLite`:



### Public Member Functions

- [SQLite](#) (const std::string &[db\\_file](#))
- [~SQLite](#) ()

### Public Attributes

- std::string [db\\_file](#)

#### 5.12.1 Detailed Description

[SQLite](#) configuration. Derived from [Database](#).

#### 5.12.2 Constructor & Destructor Documentation

##### 5.12.2.1 `clk::SQLite::SQLite (const std::string & db_file)`

[SQLite](#) constructor

##### 5.12.2.2 `clk::SQLite::~~SQLite ()`

[SQLite](#) destructor

#### 5.12.3 Member Data Documentation

##### 5.12.3.1 `std::string clk::SQLite::db_file`

Absolute path to the database file

## 5.13 clk::Layer::Suppression Struct Reference

Suppress faults matching the specified values.

```
#include <clk.h>
```

### Public Attributes

- int [confidence](#) = 0
- std::string [id](#)
- std::string [node](#)
- int [severity](#) = 0

#### 5.13.1 Detailed Description

Suppress faults matching the specified values.

#### 5.13.2 Member Data Documentation

##### 5.13.2.1 int clk::Layer::Suppression::confidence = 0

Suppress all faults with a value less than the confidence value.

##### 5.13.2.2 std::string clk::Layer::Suppression::id

Suppress all messages with a matching id. If empty, does not suppress on id.

##### 5.13.2.3 std::string clk::Layer::Suppression::node

Suppress all messages containing the node value. If empty, does not suppress on node.

##### 5.13.2.4 int clk::Layer::Suppression::severity = 0

Suppress all faults with a value less than the severity value.

# Index

- ~Database
  - clk::Database, [14](#)
- ~Layer
  - clk::Layer, [21](#)
- ~SQLite
  - clk::SQLite, [29](#)
- analyze
  - clk::Layer, [21](#)
- ascending
  - clk::Layer::Sorting, [28](#)
- clk::Database, [14](#)
  - ~Database, [14](#)
- clk::Diagnosis, [15](#)
- clk::Fault, [16](#)
  - confidence, [16](#)
  - id, [16](#)
  - msg, [16](#)
  - nodes, [17](#)
  - remedy, [17](#)
  - rowid, [17](#)
  - severity, [17](#)
  - suppressed, [17](#)
- clk::Layer, [20](#)
  - ~Layer, [21](#)
  - analyze, [21](#)
  - collect, [21](#)
  - get\_faults, [21](#)
  - get\_messages, [22](#)
  - get\_nodes, [22](#)
  - get\_version\_number, [22](#)
  - Layer, [21](#)
  - message, [23](#)
  - progress, [22](#)
- clk::Layer::Config, [9](#)
  - Config, [10](#)
  - config\_params, [10](#)
  - db, [10](#)
  - expiration, [11](#)
  - extension\_mods, [11](#)
  - extension\_path, [11](#)
  - kb\_mods, [11](#)
  - kb\_path, [11](#)
  - language, [11](#)
  - node\_source, [11](#)
  - nodes, [11](#)
  - now, [11](#)
- clk::Layer::ConfigParam, [13](#)
  - key, [13](#)
  - module, [13](#)
  - values, [13](#)
- clk::Layer::Filter, [18](#)
  - confidence, [18](#)
  - ids, [18](#)
  - nodes, [18](#)
  - severity, [18](#)
  - state, [18](#)
  - suppressed, [19](#)
  - type, [19](#)
- clk::Layer::Message, [24](#)
  - level, [24](#)
  - msg, [24](#)
- clk::Layer::Sorting, [28](#)
  - ascending, [28](#)
  - field, [28](#)
  - Sorting, [28](#)
- clk::Layer::Suppression, [30](#)
  - confidence, [30](#)
  - id, [30](#)
  - node, [30](#)
  - severity, [30](#)
- clk::Node, [25](#)
  - name, [25](#)

- role\_t, [25](#)
  - roles, [25](#)
  - subcluster, [25](#)
- clk::Sign, [27](#)
- state, [27](#)
- clk::SQLite, [29](#)
- ~SQLite, [29](#)
  - db\_file, [29](#)
  - SQLite, [29](#)
- collect
  - clk::Layer, [21](#)
- confidence
  - clk::Fault, [16](#)
  - clk::Layer::Filter, [18](#)
  - clk::Layer::Suppression, [30](#)
- Config
  - clk::Layer::Config, [10](#)
- config\_params
  - clk::Layer::Config, [10](#)
- db
  - clk::Layer::Config, [10](#)
- db\_file
  - clk::SQLite, [29](#)
- expiration
  - clk::Layer::Config, [11](#)
- extension\_mods
  - clk::Layer::Config, [11](#)
- extension\_path
  - clk::Layer::Config, [11](#)
- field
  - clk::Layer::Sorting, [28](#)
- get\_faults
  - clk::Layer, [21](#)
- get\_messages
  - clk::Layer, [22](#)
- get\_nodes
  - clk::Layer, [22](#)
- get\_version\_number
  - clk::Layer, [22](#)
- id
  - clk::Fault, [16](#)
  - clk::Layer::Suppression, [30](#)
- ids
  - clk::Layer::Filter, [18](#)
- kb\_mods
  - clk::Layer::Config, [11](#)
- kb\_path
  - clk::Layer::Config, [11](#)
- key
  - clk::Layer::ConfigParam, [13](#)
- language
  - clk::Layer::Config, [11](#)
- Layer
  - clk::Layer, [21](#)
- level
  - clk::Layer::Message, [24](#)
- message
  - clk::Layer, [23](#)
- module
  - clk::Layer::ConfigParam, [13](#)
- msg
  - clk::Fault, [16](#)
  - clk::Layer::Message, [24](#)
- name
  - clk::Node, [25](#)
- node
  - clk::Layer::Suppression, [30](#)
- node\_source
  - clk::Layer::Config, [11](#)
- nodes
  - clk::Fault, [17](#)
  - clk::Layer::Config, [11](#)
  - clk::Layer::Filter, [18](#)
- now
  - clk::Layer::Config, [11](#)
- progress
  - clk::Layer, [22](#)
- remedy
  - clk::Fault, [17](#)
- role\_t
  - clk::Node, [25](#)
- roles
  - clk::Node, [25](#)

---

- rowid
  - clk::Fault, [17](#)
- severity
  - clk::Fault, [17](#)
  - clk::Layer::Filter, [18](#)
  - clk::Layer::Suppression, [30](#)
- Sorting
  - clk::Layer::Sorting, [28](#)
- SQLite
  - clk::SQLite, [29](#)
- state
  - clk::Layer::Filter, [18](#)
  - clk::Sign, [27](#)
- subcluster
  - clk::Node, [25](#)
- suppressed
  - clk::Fault, [17](#)
  - clk::Layer::Filter, [19](#)
- type
  - clk::Layer::Filter, [19](#)
- values
  - clk::Layer::ConfigParam, [13](#)