

liboqs-cpp

0.2.0

Generated by Doxygen 1.8.16



<b>1 liboqs-cpp: C++ bindings for liboqs</b>	<b>1</b>
<b>2 Namespace Index</b>	<b>5</b>
2.1 Namespace List . . . . .	5
<b>3 Hierarchical Index</b>	<b>7</b>
3.1 Class Hierarchy . . . . .	7
<b>4 Class Index</b>	<b>9</b>
4.1 Class List . . . . .	9
<b>5 File Index</b>	<b>11</b>
5.1 File List . . . . .	11
<b>6 Namespace Documentation</b>	<b>13</b>
6.1 internal Namespace Reference . . . . .	13
6.1.1 Detailed Description . . . . .	13
6.2 oqs Namespace Reference . . . . .	13
6.2.1 Detailed Description . . . . .	14
6.2.2 Typedef Documentation . . . . .	14
6.2.2.1 byte . . . . .	14
6.2.2.2 bytes . . . . .	14
6.2.2.3 OQS_STATUS . . . . .	14
6.2.3 Function Documentation . . . . .	14
6.2.3.1 hex_chop() . . . . .	14
6.3 oqs::C Namespace Reference . . . . .	15
6.3.1 Detailed Description . . . . .	15
6.4 oqs::internal Namespace Reference . . . . .	15
6.5 oqs_literals Namespace Reference . . . . .	15
6.5.1 Function Documentation . . . . .	16
6.5.1.1 operator""_bytes() . . . . .	16
<b>7 Class Documentation</b>	<b>17</b>
7.1 oqs::KeyEncapsulation::alg_details_ Struct Reference . . . . .	17
7.1.1 Detailed Description . . . . .	17
7.1.2 Member Data Documentation . . . . .	17
7.1.2.1 claimed_nist_level . . . . .	17
7.1.2.2 is_ind_cca . . . . .	17
7.1.2.3 length_ciphertext . . . . .	17
7.1.2.4 length_public_key . . . . .	17
7.1.2.5 length_secret_key . . . . .	18
7.1.2.6 length_shared_secret . . . . .	18
7.1.2.7 name . . . . .	18
7.1.2.8 version . . . . .	18

7.2 oqs::Signature::alg_details_ Struct Reference	18
7.2.1 Detailed Description	18
7.2.2 Member Data Documentation	18
7.2.2.1 claimed_nist_level	18
7.2.2.2 is_euf_cma	18
7.2.2.3 length_public_key	18
7.2.2.4 length_secret_key	19
7.2.2.5 max_length_signature	19
7.2.2.6 name	19
7.2.2.7 version	19
7.3 oqs::internal::HexChop Class Reference	19
7.3.1 Detailed Description	19
7.3.2 Constructor & Destructor Documentation	20
7.3.2.1 HexChop()	20
7.3.3 Member Function Documentation	20
7.3.3.1 manipulate_ostream_()	20
7.3.4 Friends And Related Function Documentation	21
7.3.4.1 operator<<	21
7.3.5 Member Data Documentation	22
7.3.5.1 end_	22
7.3.5.2 start_	22
7.3.5.3 v_	22
7.4 oqs::KEMs Class Reference	22
7.4.1 Detailed Description	23
7.4.2 Constructor & Destructor Documentation	23
7.4.2.1 KEMs()	24
7.4.3 Member Function Documentation	24
7.4.3.1 get_enabled_KEMs()	24
7.4.3.2 get_KEM_name()	24
7.4.3.3 get_supported_KEMs()	24
7.4.3.4 is_KEM_enabled()	24
7.4.3.5 is_KEM_supported()	25
7.4.3.6 max_number_KEMs()	25
7.4.4 Friends And Related Function Documentation	25
7.4.4.1 internal::Singleton< const KEMs >	25
7.5 oqs::KeyEncapsulation Class Reference	25
7.5.1 Detailed Description	27
7.5.2 Constructor & Destructor Documentation	27
7.5.2.1 KeyEncapsulation() [1/3]	27
7.5.2.2 KeyEncapsulation() [2/3]	27
7.5.2.3 KeyEncapsulation() [3/3]	27
7.5.2.4 ~KeyEncapsulation()	28

7.5.3 Member Function Documentation	28
7.5.3.1 decap_secret()	28
7.5.3.2 encap_secret()	28
7.5.3.3 export_secret_key()	28
7.5.3.4 generate_keypair()	29
7.5.3.5 get_details() [1/2]	29
7.5.3.6 get_details() [2/2]	29
7.5.3.7 operator=() [1/2]	29
7.5.3.8 operator=() [2/2]	29
7.5.4 Friends And Related Function Documentation	29
7.5.4.1 operator<< [1/2]	30
7.5.4.2 operator<< [2/2]	30
7.5.5 Member Data Documentation	30
7.5.5.1 alg_name_	30
7.5.5.2 details_	30
7.5.5.3 kem_	30
7.5.5.4 secret_key_	31
7.6 oqs::MechanismNotEnabledError Class Reference	31
7.6.1 Detailed Description	32
7.6.2 Constructor & Destructor Documentation	32
7.6.2.1 MechanismNotEnabledError()	32
7.7 oqs::MechanismNotSupportedError Class Reference	32
7.7.1 Detailed Description	33
7.7.2 Constructor & Destructor Documentation	33
7.7.2.1 MechanismNotSupportedError()	33
7.8 oqs::Signature Class Reference	33
7.8.1 Detailed Description	35
7.8.2 Constructor & Destructor Documentation	35
7.8.2.1 Signature() [1/3]	35
7.8.2.2 Signature() [2/3]	35
7.8.2.3 Signature() [3/3]	35
7.8.2.4 ~Signature()	36
7.8.3 Member Function Documentation	36
7.8.3.1 export_secret_key()	36
7.8.3.2 generate_keypair()	36
7.8.3.3 get_details() [1/2]	36
7.8.3.4 get_details() [2/2]	36
7.8.3.5 operator=() [1/2]	37
7.8.3.6 operator=() [2/2]	37
7.8.3.7 sign()	37
7.8.3.8 verify()	37
7.8.4 Friends And Related Function Documentation	38

7.8.4.1 operator<< [1/2]	38
7.8.4.2 operator<< [2/2]	38
7.8.5 Member Data Documentation	38
7.8.5.1 alg_name_	38
7.8.5.2 details_	39
7.8.5.3 secret_key_	39
7.8.5.4 sig_	39
7.9 oqs::Sigs Class Reference	39
7.9.1 Detailed Description	40
7.9.2 Constructor & Destructor Documentation	40
7.9.2.1 Sigs()	41
7.9.3 Member Function Documentation	41
7.9.3.1 get_enabled_sigs()	41
7.9.3.2 get_sig_name()	41
7.9.3.3 get_supported_sigs()	41
7.9.3.4 is_sig_enabled()	41
7.9.3.5 is_sig_supported()	42
7.9.3.6 max_number_sigs()	42
7.9.4 Friends And Related Function Documentation	42
7.9.4.1 internal::Singleton< const Sigs >	42
7.10 oqs::internal::Singleton< T > Class Template Reference	42
7.10.1 Detailed Description	43
7.10.2 Constructor & Destructor Documentation	43
7.10.2.1 Singleton() [1/2]	43
7.10.2.2 Singleton() [2/2]	43
7.10.2.3 ~Singleton()	44
7.10.3 Member Function Documentation	44
7.10.3.1 get_instance()	44
7.10.3.2 operator=()	44
7.11 oqs::Timer< T, CLOCK_T > Class Template Reference	44
7.11.1 Detailed Description	45
7.11.2 Constructor & Destructor Documentation	45
7.11.2.1 Timer()	45
7.11.2.2 ~Timer()	45
7.11.3 Member Function Documentation	45
7.11.3.1 get_duration()	45
7.11.3.2 tic()	46
7.11.3.3 tics()	46
7.11.3.4 toc()	46
7.11.4 Friends And Related Function Documentation	46
7.11.4.1 operator<<	46
7.11.5 Member Data Documentation	46

---

7.11.5.1 end_ . . . . .	47
7.11.5.2 start_ . . . . .	47
<b>8 File Documentation</b>	<b>49</b>
8.1 oqs_cpp.h File Reference . . . . .	49
8.1.1 Detailed Description . . . . .	50
8.1.2 Function Documentation . . . . .	50
8.1.2.1 operator<<() [1/2] . . . . .	50
8.1.2.2 operator<<() [2/2] . . . . .	51
8.2 /Users/vlad/liboqs-cpp/README.md File Reference . . . . .	51
<b>Index</b>	<b>53</b>





# Chapter 1

## liboqs-cpp: C++ bindings for liboqs

**liboqs-cpp** offers a C++ wrapper for the master branch of [Open Quantum Safe liboqs](#) C library, which is a C library for quantum-resistant cryptographic algorithms.

The wrapper is written in standard C++11, hence in the following it is assumed that you have access to a C++11 compliant compiler. liboqs-cpp has been extensively tested on Linux, macOS and Windows systems. Continuous integration is provided via Travis CI and AppVeyor.

### Pre-requisites

liboqs-cpp depends on the [liboqs](#) C library; liboqs master branch must first be compiled as a Linux/macOS/[Windows](#) library, see the specific platform building instructions below.

### Contents

liboqs-cpp is a header-only wrapper. The project contains the following files and directories:

- `**include/oqs_cpp.h`: main header file for the wrapper\*\*
- `examples/kem.cpp`: key encapsulation example
- `examples/sig.cpp`: signature example
- `doc`: Doxygen-generated detailed documentation
- `unit_tests`: unit tests written using Google Test (included)
- `VisualStudio/liboqs-cpp.sln`: Visual Studio 2017 solution

### Usage

To avoid namespace pollution, liboqs-cpp includes all of its code inside the namespace `oqs`. All of the liboqs C API is located in the namespace `oqs::C`, hence to use directly a C API function you must qualify the call with `oqs::C::liboqs_C_function(...)`.

liboqs-cpp defines four main classes: `oqs::KeyEncapsulation` and `oqs::Signature`, providing post-quantum key encapsulation and signature mechanisms, respectively, and `oqs::KEMs` and `oqs::Sigs`, containing only static member functions that provide information related to the available key encapsulation mechanisms or signature mechanism, respectively.

`oqs::KeyEncapsulation` and/or `oqs::Signature` must be instantiated with a string identifying one of mechanisms supported by liboqs; these can be enumerated using the `oqs::KEMs::get_enabled_KEM_mechanisms()` and `oqs::Sigs::get_enabled_sig_mechanisms()` member functions.

The wrapper also defines a high resolution timing class, `oqs::Timer<>`.

The examples in the [examples](#) directory are self-explanatory and provide more details about the wrapper's API.

## Building on POSIX (Linux/UNIX-like) platforms

First, you must build the master branch of liboqs according to the [liboqs building instructions](#), followed (optionally) by a `sudo make install` to ensure that the compiled library is system-wide visible (by default it installs under `/usr/local/include` and `/usr/local/lib` under Linux/macOS). Next, to use the wrapper, you simply `#include "oqs_cpp.h"` in your program. The wrapper contains a CMake build system for both examples and unit tests. To compile and run the examples, create a `build` directory inside the root directory of the project, change directory to `build`, then type

```
cmake .. -DLIBOQS_INCLUDE_DIR=/usr/local/include -DLIBOQS_LIB_DIR=/usr/local/lib
make -j4
```

The above commands build all examples in `examples`, i.e. `examples/kem` and `examples/sig`, assuming the CMake build system is available on your platform. The `-DLIBOQS_INCLUDE_DIR` and `-DLIBOQS_LIB_DIR` flags specify the location to the liboqs headers and compiled library (in this case `/usr/local/include` and `/usr/local/lib`, respectively). You may replace the `-j4` flag with your processor's number of cores, e.g. use `-j8` if your system has 8 cores. To build only a specific example, e.g. `examples/kem`, specify the target as the argument of the `make` command, such as

```
make kem
```

To compile and run the unit tests, first `cd unit_tests`, then create a `build` directory inside `unit_tests`, change directory to it, and finally type

```
cmake .. -DLIBOQS_INCLUDE_DIR=/usr/local/include -DLIBOQS_LIB_DIR=/usr/local/lib
make -j4
```

The above commands build `tests/oqs_cpp_testing` suite of unit tests.

## Building on Windows

We provide CMake support for Visual Studio. We recommend using Visual Studio 2017 or later (preferably Visual Studio 2019). For comprehensive details about using CMake with Visual Studio please read [this page](#). In addition, a Visual Studio 2017 solution containing both key encapsulation and signature examples from [examples](#) as two separate projects is provided in the `VisualStudio` directory. Building instructions:

- First, you must clone/download and build liboqs under Windows, see [liboqs Windows building instructions](#) for more details.
- Next, you must [set the environment variable](#) `LIBOQS_INSTALL_PATH` to point to the location of liboqs, e.g. `C:\liboqs`.
- Only after completing the steps above you may build the liboqs-cpp solution (or each individual projects within the solution). In case you end up with a linker error, make sure that the corresponding liboqs target was built, i.e. if building a `Release` version with an `x64` target, then the corresponding `Release/x64` solution from liboqs should have been built in advance.

In case you get a "Missing Windows SDK" error, right-click on the solution name and choose "Retarget solution" to re-target the projects in the solution to your available Windows SDK.

## Limitations and security

liboqs is designed for prototyping and evaluating quantum-resistant cryptography. Security of proposed quantum-resistant algorithms may rapidly change as research advances, and may ultimately be completely insecure against either classical or quantum computers.

We believe that the NIST Post-Quantum Cryptography standardization project is currently the best avenue to identifying potentially quantum-resistant algorithms. liboqs does not intend to "pick winners", and we strongly recommend that applications and protocols rely on the outcomes of the NIST standardization project when deploying post-quantum cryptography.

We acknowledge that some parties may want to begin deploying post-quantum cryptography prior to the conclusion of the NIST standardization project. We strongly recommend that any attempts to do make use of so-called **hybrid cryptography**, in which post-quantum public-key algorithms are used alongside traditional public key algorithms (like RSA or elliptic curves) so that the solution is at least no less secure than existing traditional cryptography. Just like liboqs, liboqs-cpp is provided "as is", without warranty of any kind. See [LICENSE](#) for the full disclaimer.

## License

liboqs-cpp is licensed under the MIT License; see [LICENSE](#) for details.

## Team

The Open Quantum Safe project is led by [Douglas Stebila](#) and [Michele Mosca](#) at the University of Waterloo.

liboqs-cpp was developed by [Vlad Gheorghiu](#) at evolutionQ and University of Waterloo.

## Support

Financial support for the development of Open Quantum Safe has been provided by Amazon Web Services and the Tutte Institute for Mathematics and Computing.

We'd like to make a special acknowledgement to the companies who have dedicated programmer time to contribute source code to OQS, including Amazon Web Services, evolutionQ, and Microsoft Research.

Research projects which developed specific components of OQS have been supported by various research grants, including funding from the Natural Sciences and Engineering Research Council of Canada (NSERC); see the source papers for funding acknowledgments.



## Chapter 2

# Namespace Index

### 2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">internal</a>	Internal implementation details . . . . .	13
<a href="#">oqs</a>	Main namespace for the liboqs C++ wrapper . . . . .	13
<a href="#">oqs::C</a>	Namespace containing all of the oqs <a href="#">C</a> functions, so they do not pollute the oqs namespace . .	15
<a href="#">oqs::internal</a>	. . . . .	15
<a href="#">oqs_literals</a>	. . . . .	15



## Chapter 3

# Hierarchical Index

### 3.1 Class Hierarchy

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

oqs::KeyEncapsulation::alg_details_ . . . . .	17
oqs::Signature::alg_details_ . . . . .	18
oqs::internal::HexChop . . . . .	19
oqs::KeyEncapsulation . . . . .	25
runtime_error	
oqs::MechanismNotEnabledError . . . . .	31
oqs::MechanismNotSupportedError . . . . .	32
oqs::Signature . . . . .	33
oqs::internal::Singleton< T > . . . . .	42
oqs::KEMs . . . . .	22
oqs::internal::Singleton< const KEMs > . . . . .	42
oqs::internal::Singleton< const Sigs > . . . . .	42
oqs::Sigs . . . . .	39
oqs::Timer< T, CLOCK_T > . . . . .	44





## Chapter 4

# Class Index

### 4.1 Class List

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

<a href="#">oqs::KeyEncapsulation::alg_details_</a>	
KEM algorithm details	17
<a href="#">oqs::Signature::alg_details_</a>	
Signature algorithm details	18
<a href="#">oqs::internal::HexChop</a>	
Std::ostream manipulator for long vectors of <a href="#">oqs::byte</a> , use it to display only a small number of elements from the beginning and end of the vector	19
<a href="#">oqs::KEMs</a>	
Singleton class, contains details about supported/enabled key exchange mechanisms ( <a href="#">KEMs</a> )	22
<a href="#">oqs::KeyEncapsulation</a>	
Key encapsulation mechanisms	25
<a href="#">oqs::MechanismNotEnabledError</a>	
Cryptographic scheme not enabled	31
<a href="#">oqs::MechanismNotSupportedError</a>	
Cryptographic scheme not supported	32
<a href="#">oqs::Signature</a>	
Signature mechanisms	33
<a href="#">oqs::Sigs</a>	
Singleton class, contains details about supported/enabled signature mechanisms	39
<a href="#">oqs::internal::Singleton&lt; T &gt;</a>	
Singleton class using CRTP pattern	42
<a href="#">oqs::Timer&lt; T, CLOCK_T &gt;</a>	
High resolution timer	44



## Chapter 5

# File Index

### 5.1 File List

Here is a list of all files with brief descriptions:

<a href="#">oqs_cpp.h</a>	Main header file for the liboqs C++ wrapper . . . . .	49
---------------------------	---	----



## Chapter 6

# Namespace Documentation

### 6.1 internal Namespace Reference

Internal implementation details.

#### 6.1.1 Detailed Description

Internal implementation details.

### 6.2 oqs Namespace Reference

Main namespace for the liboqs C++ wrapper.

#### Namespaces

- [C](#)  
*Namespace containing all of the oqs [C](#) functions, so they do not pollute the oqs namespace.*
- [internal](#)

#### Classes

- class [KEMs](#)  
*Singleton class, contains details about supported/enabled key exchange mechanisms ([KEMs](#))*
- class [KeyEncapsulation](#)  
*Key encapsulation mechanisms.*
- class [MechanismNotEnabledError](#)  
*Cryptographic scheme not enabled.*
- class [MechanismNotSupportedError](#)  
*Cryptographic scheme not supported.*
- class [Signature](#)  
*[Signature](#) mechanisms.*
- class [Sigs](#)  
*Singleton class, contains details about supported/enabled signature mechanisms.*
- class [Timer](#)  
*High resolution timer.*

#### Typedefs

- using [byte](#) = std::uint8\_t  
*byte (unsigned)*
- using [bytes](#) = std::vector< [byte](#) >

*vector of bytes (unsigned)*

- using `OQS_STATUS` = `C::OQS_STATUS`  
*bring OQS\_STATUS into the oqs namespace*

## Functions

- `internal::HexChop hex_chop` (const `oqs::bytes` &`v`, `std::size_t start=8`, `std::size_t end=8`)  
*Constructs an instance of `oqs::internal::HexChop`.*

### 6.2.1 Detailed Description

Main namespace for the liboqs C++ wrapper.

### 6.2.2 Typedef Documentation

#### 6.2.2.1 byte

using `oqs::byte` = typedef `std::uint8_t`  
byte (unsigned)

#### 6.2.2.2 bytes

using `oqs::bytes` = typedef `std::vector<byte>`  
vector of bytes (unsigned)

#### 6.2.2.3 OQS\_STATUS

using `oqs::OQS_STATUS` = typedef `C::OQS_STATUS`  
bring OQS\_STATUS into the oqs namespace

### 6.2.3 Function Documentation

#### 6.2.3.1 hex\_chop()

```
internal::HexChop oqs::hex_chop (
    const oqs::bytes & v,
    std::size_t start = 8,
    std::size_t end = 8 ) [inline]
```

Constructs an instance of `oqs::internal::HexChop`.

#### Parameters

<code>v</code>	Vector of bytes
----------------	-----------------------

## Parameters

<i>start</i>	Number of hex characters displayed from the beginning of the vector
<i>end</i>	Number of hex characters displayed from the end of the vector

## Returns

Instance of [oqs::internal::HexChop](#)

## 6.3 oqs::C Namespace Reference

Namespace containing all of the oqs [C](#) functions, so they do not pollute the oqs namespace.

### 6.3.1 Detailed Description

Namespace containing all of the oqs [C](#) functions, so they do not pollute the oqs namespace.

## 6.4 oqs::internal Namespace Reference

### Classes

- class [HexChop](#)  
*std::ostream manipulator for long vectors of [oqs::byte](#), use it to display only a small number of elements from the beginning and end of the vector*
- class [Singleton](#)  
*[Singleton](#) class using CRTP pattern.*

## 6.5 oqs\_literals Namespace Reference

### Functions

- [oqs::bytes operator""\\_bytes](#) (const char \*c\_str, std::size\_t length)  
*User-defined literal operator for converting C-style strings to [oqs::bytes](#).*

## 6.5.1 Function Documentation

### 6.5.1.1 `operator""_bytes()`

```
oqs::bytes oqs_literals::operator""_bytes (
    const char * c_str,
    std::size_t length ) [inline]
```

User-defined literal operator for converting C-style strings to [oqs::bytes](#).

#### Note

The null terminator is not included

#### Parameters

<i>c_str</i>	C-style string
<i>length</i>	C-style string length (deduced automatically by the compiler)

#### Returns

The byte representation of the input C-style string



# Chapter 7

## Class Documentation

### 7.1 oqs::KeyEncapsulation::alg\_details\_ Struct Reference

KEM algorithm details.

#### Public Attributes

- std::string [name](#)
- std::string [version](#)
- std::size\_t [claimed\\_nist\\_level](#)
- bool [is\\_ind\\_cca](#)
- std::size\_t [length\\_public\\_key](#)
- std::size\_t [length\\_secret\\_key](#)
- std::size\_t [length\\_ciphertext](#)
- std::size\_t [length\\_shared\\_secret](#)

#### 7.1.1 Detailed Description

KEM algorithm details.

#### 7.1.2 Member Data Documentation

##### 7.1.2.1 claimed\_nist\_level

```
std::size_t oqs::KeyEncapsulation::alg_details_::claimed_nist_level
```

##### 7.1.2.2 is\_ind\_cca

```
bool oqs::KeyEncapsulation::alg_details_::is_ind_cca
```

##### 7.1.2.3 length\_ciphertext

```
std::size_t oqs::KeyEncapsulation::alg_details_::length_ciphertext
```

##### 7.1.2.4 length\_public\_key

```
std::size_t oqs::KeyEncapsulation::alg_details_::length_public_key
```

#### 7.1.2.5 length\_secret\_key

```
std::size_t oqs::KeyEncapsulation::alg_details_::length_secret_key
```

#### 7.1.2.6 length\_shared\_secret

```
std::size_t oqs::KeyEncapsulation::alg_details_::length_shared_secret
```

#### 7.1.2.7 name

```
std::string oqs::KeyEncapsulation::alg_details_::name
```

#### 7.1.2.8 version

```
std::string oqs::KeyEncapsulation::alg_details_::version
```

The documentation for this struct was generated from the following file:

- [oqs\\_cpp.h](#)

## 7.2 oqs::Signature::alg\_details\_ Struct Reference

[Signature](#) algorithm details.

### Public Attributes

- std::string [name](#)
- std::string [version](#)
- std::size\_t [claimed\\_nist\\_level](#)
- bool [is\\_euf\\_cma](#)
- std::size\_t [length\\_public\\_key](#)
- std::size\_t [length\\_secret\\_key](#)
- std::size\_t [max\\_length\\_signature](#)

### 7.2.1 Detailed Description

[Signature](#) algorithm details.

### 7.2.2 Member Data Documentation

#### 7.2.2.1 claimed\_nist\_level

```
std::size_t oqs::Signature::alg_details_::claimed_nist_level
```

#### 7.2.2.2 is\_euf\_cma

```
bool oqs::Signature::alg_details_::is_euf_cma
```

#### 7.2.2.3 length\_public\_key

```
std::size_t oqs::Signature::alg_details_::length_public_key
```

#### 7.2.2.4 length\_secret\_key

`std::size_t oqs::Signature::alg_details_::length_secret_key`

#### 7.2.2.5 max\_length\_signature

`std::size_t oqs::Signature::alg_details_::max_length_signature`

#### 7.2.2.6 name

`std::string oqs::Signature::alg_details_::name`

#### 7.2.2.7 version

`std::string oqs::Signature::alg_details_::version`

The documentation for this struct was generated from the following file:

- [oqs\\_cpp.h](#)

## 7.3 oqs::internal::HexChop Class Reference

`std::ostream` manipulator for long vectors of [oqs::byte](#), use it to display only a small number of elements from the beginning and end of the vector

```
#include <oqs_cpp.h>
```

### Public Member Functions

- [HexChop](#) (const [oqs::bytes](#) &v, `std::size_t` start, `std::size_t` end)

*Constructs an instance of [oqs::internal::HexChop](#).*

### Private Member Functions

- void [manipulate\\_ostream\\_](#) (`std::ostream` &os, `std::size_t` start, `std::size_t` end, bool is\_short) const

*`std::ostream` manipulator*

### Private Attributes

- [bytes](#) v\_  
*vector of bytes*
- `std::size_t` start\_  
*number of hex bytes taken from the start*
- `std::size_t` end\_  
*number of hex bytes taken from the end*

### Friends

- `std::ostream` & [operator<<](#) (`std::ostream` &os, const [HexChop](#) &rhs)

*`std::ostream` extraction operator for [oqs::internal::HexChop](#)*

### 7.3.1 Detailed Description

`std::ostream` manipulator for long vectors of [oqs::byte](#), use it to display only a small number of elements from the beginning and end of the vector

## 7.3.2 Constructor & Destructor Documentation

### 7.3.2.1 HexChop()

```
oqs::internal::HexChop::HexChop (
    const oqs::bytes & v,
    std::size_t start,
    std::size_t end ) [inline], [explicit]
```

Constructs an instance of `oqs::internal::HexChop`.

#### Parameters

<i>v</i>	Vector of bytes
<i>start</i>	Number of hex characters displayed from the beginning of the vector
<i>end</i>	Number of hex characters displayed from the end of the vector

## 7.3.3 Member Function Documentation

### 7.3.3.1 manipulate\_ostream\_()

```
void oqs::internal::HexChop::manipulate_ostream_ (
    std::ostream & os,
    std::size_t start,
    std::size_t end,
    bool is_short ) const [inline], [private]
```

`std::ostream` manipulator

#### Parameters

<i>os</i>	Output stream
-----------	---------------

## Parameters

<i>start</i>	Number of hex characters displayed from the beginning of the vector
<i>end</i>	Number of hex characters displayed from the end of the vector
<i>is_short</i>	Vector is too short, display all hex characters

## 7.3.4 Friends And Related Function Documentation

## 7.3.4.1 operator&lt;&lt;

```
std::ostream& operator<< (
    std::ostream & os,
    const HexChop & rhs ) [friend]
std::ostream extraction operator for oqs::internal::HexChop
```

## Parameters

<i>os</i>	Output stream
<i>rhs</i>	<a href="#">oqs::internal::HexChop</a> instance

#### Returns

Reference to the output stream

### 7.3.5 Member Data Documentation

#### 7.3.5.1 end\_

`std::size_t oqs::internal::HexChop::end_ [private]`

number of hex bytes taken from the start and from the end

#### 7.3.5.2 start\_

`std::size_t oqs::internal::HexChop::start_ [private]`

#### 7.3.5.3 v\_

`bytes oqs::internal::HexChop::v_ [private]`

vector of bytes

The documentation for this class was generated from the following file:

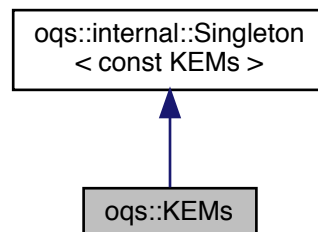
- [oqs\\_cpp.h](#)

## 7.4 oqs::KEMs Class Reference

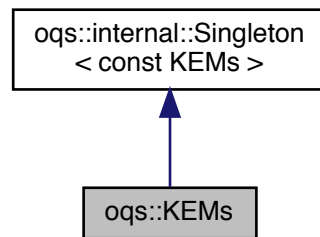
Singleton class, contains details about supported/enabled key exchange mechanisms ([KEMs](#))

`#include <oqs_cpp.h>`

Inheritance diagram for oqs::KEMs:



Collaboration diagram for oqs::KEMs:



### Static Public Member Functions

- static `std::size_t max_number_KEMs ()`  
*Maximum number of supported [KEMs](#).*
- static `bool is_KEM_supported (const std::string &alg_name)`  
*Checks whether the KEM algorithm `alg_name` is supported.*
- static `bool is_KEM_enabled (const std::string &alg_name)`  
*Checks whether the KEM algorithm `alg_name` is enabled.*
- static `std::string get_KEM_name (std::size_t alg_id)`  
*KEM algorithm name.*
- static `const std::vector< std::string > & get_supported_KEMs ()`  
*Vector of supported KEM algorithms.*
- static `const std::vector< std::string > & get_enabled_KEMs ()`  
*Vector of enabled KEM algorithms.*

### Private Member Functions

- `KEMs ()=default`  
*Private default constructor.*

### Friends

- class `internal::Singleton< const KEMs >`

### Additional Inherited Members

#### 7.4.1 Detailed Description

Singleton class, contains details about supported/enabled key exchange mechanisms ([KEMs](#))

#### 7.4.2 Constructor & Destructor Documentation

### 7.4.2.1 KEMs()

`oqs::KEMs::KEMs ( ) [private], [default]`  
Private default constructor.

#### Note

Use `oqs::KEMs::get_instance()` to create an instance

## 7.4.3 Member Function Documentation

### 7.4.3.1 get\_enabled\_KEMs()

`static const std::vector<std::string>& oqs::KEMs::get_enabled_KEMs ( ) [inline], [static]`  
Vector of enabled KEM algorithms.

#### Returns

Vector of enabled KEM algorithms

### 7.4.3.2 get\_KEM\_name()

`static std::string oqs::KEMs::get_KEM_name (`  
`std::size_t alg_id ) [inline], [static]`  
KEM algorithm name.

#### Parameters

<i>alg_id</i>	Cryptographic algo- rithm numer- ical id
---------------	---

#### Returns

KEM algorithm name

### 7.4.3.3 get\_supported\_KEMs()

`static const std::vector<std::string>& oqs::KEMs::get_supported_KEMs ( ) [inline], [static]`  
Vector of supported KEM algorithms.

#### Returns

Vector of supported KEM algorithms

### 7.4.3.4 is\_KEM\_enabled()

`static bool oqs::KEMs::is_KEM_enabled (`  
`const std::string & alg_name ) [inline], [static]`  
Checks whether the KEM algorithm *alg\_name* is enabled.



## Parameters

<i>alg_name</i>	Cryptographic algo- rithm name
-----------------	---

## Returns

True if the KEM algorithm is enabled, false otherwise

**7.4.3.5 is\_KEM\_supported()**

```
static bool oqs::KEMs::is_KEM_supported (
    const std::string & alg_name ) [inline], [static]
```

Checks whether the KEM algorithm *alg\_name* is supported.

## Parameters

<i>alg_name</i>	Cryptographic algo- rithm name
-----------------	---

## Returns

True if the KEM algorithm is supported, false otherwise

**7.4.3.6 max\_number\_KEMs()**

```
static std::size_t oqs::KEMs::max_number_KEMs ( ) [inline], [static]
```

Maximum number of supported [KEMs](#).

## Returns

Maximum number of supported [KEMs](#)

**7.4.4 Friends And Related Function Documentation****7.4.4.1 internal::Singleton< const KEMs >**

```
friend class internal::Singleton< const KEMs > [friend]
```

The documentation for this class was generated from the following file:

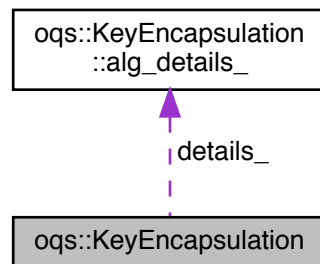
- [oqs\\_cpp.h](#)

**7.5 oqs::KeyEncapsulation Class Reference**

Key encapsulation mechanisms.

```
#include <oqs_cpp.h>
```

Collaboration diagram for `oqs::KeyEncapsulation`:



## Classes

- struct `alg_details_`  
*KEM algorithm details.*

## Public Member Functions

- `KeyEncapsulation` (const std::string &alg\_name, const bytes &secret\_key={})  
*Constructs an instance of `oqs::KeyEncapsulation`.*
- `KeyEncapsulation` (const `KeyEncapsulation` &)=default  
*Default copy constructor.*
- `KeyEncapsulation` & operator= (const `KeyEncapsulation` &)=default  
*Default copy assignment operator.*
- `KeyEncapsulation` (`KeyEncapsulation` &&rhs)  
*Move constructor, guarantees that the rvalue secret key is always zeroed.*
- `KeyEncapsulation` & operator= (`KeyEncapsulation` &&rhs)  
*Move assignment operator, guarantees that the rvalue secret key is always zeroed.*
- virtual `~KeyEncapsulation` ()  
*Virtual default destructor.*
- const `alg_details_` & get\_details () const &  
*KEM algorithm details, lvalue overload.*
- `alg_details_` get\_details () const &&  
*KEM algorithm details, rvalue overload.*
- bytes generate\_keypair ()  
*Generate public key/secret key pair.*
- bytes export\_secret\_key () const  
*Export secret key.*
- std::pair< bytes, bytes > encap\_secret (const bytes &public\_key) const  
*Encapsulate secret.*
- bytes decap\_secret (const bytes &ciphertext) const  
*Decapsulate secret.*

## Private Attributes

- `std::string alg_name_`  
*cryptographic algorithm name*
- `std::shared_ptr< C::OQS_KEM > kem_`  
*liboqs smart pointer to C::OQS\_KEM*
- `bytes secret_key_ {}`  
*secret key*
- `struct oqs::KeyEncapsulation::alg_details_ details_`

## Friends

- `std::ostream & operator<< (std::ostream &os, const alg_details_ &rhs)`  
*std::ostream extraction operator for the KEM algorithm details*
- `std::ostream & operator<< (std::ostream &os, const KeyEncapsulation &rhs)`  
*std::ostream extraction operator for oqs::KeyEncapsulation*

### 7.5.1 Detailed Description

Key encapsulation mechanisms.

### 7.5.2 Constructor & Destructor Documentation

#### 7.5.2.1 KeyEncapsulation() [1/3]

```
oqs::KeyEncapsulation::KeyEncapsulation (
    const std::string & alg_name,
    const bytes & secret_key = {} ) [inline], [explicit]
```

Constructs an instance of `oqs::KeyEncapsulation`.

#### Parameters

<i>alg_name</i>	Cryptographic algo- rithm name
<i>secret_key</i>	Secret key (op- tional)

#### 7.5.2.2 KeyEncapsulation() [2/3]

```
oqs::KeyEncapsulation::KeyEncapsulation (
    const KeyEncapsulation & ) [default]
```

Default copy constructor.

#### 7.5.2.3 KeyEncapsulation() [3/3]

```
oqs::KeyEncapsulation::KeyEncapsulation (
    KeyEncapsulation && rhs ) [inline]
```

Move constructor, guarantees that the rvalue secret key is always zeroed.

## Parameters

<i>rhs</i>	<a href="#">oqs::KeyEncapsulation</a> in- stance
------------	--

**7.5.2.4 ~KeyEncapsulation()**

`virtual oqs::KeyEncapsulation::~~KeyEncapsulation ( ) [inline], [virtual]`  
 Virtual default destructor.

**7.5.3 Member Function Documentation****7.5.3.1 decap\_secret()**

`bytes oqs::KeyEncapsulation::decap_secret (`  
     `const bytes & ciphertext ) const [inline]`

Decapsulate secret.

## Parameters

<i>ciphertext</i>	Ciphertext
-------------------	------------

## Returns

Shared secret

**7.5.3.2 encap\_secret()**

`std::pair<bytes, bytes> oqs::KeyEncapsulation::encap_secret (`  
     `const bytes & public_key ) const [inline]`

Encapsulate secret.

## Parameters

<i>public_key</i>	Public key
-------------------	---------------

## Returns

Pair consisting of 1) ciphertext, and 2) shared secret

**7.5.3.3 export\_secret\_key()**

`bytes oqs::KeyEncapsulation::export_secret_key ( ) const [inline]`  
 Export secret key.

## Returns

Secret key

#### 7.5.3.4 generate\_keypair()

```
bytes oqs::KeyEncapsulation::generate_keypair ( ) [inline]
```

Generate public key/secret key pair.

##### Returns

Public key

#### 7.5.3.5 get\_details() [1/2]

```
const alg_details_& oqs::KeyEncapsulation::get_details ( ) const & [inline]
```

KEM algorithm details, lvalue overload.

##### Returns

KEM algorithm details

#### 7.5.3.6 get\_details() [2/2]

```
alg_details_ oqs::KeyEncapsulation::get_details ( ) const && [inline]
```

KEM algorithm details, rvalue overload.

##### Returns

KEM algorithm details

#### 7.5.3.7 operator=() [1/2]

```
KeyEncapsulation& oqs::KeyEncapsulation::operator= (
    const KeyEncapsulation & ) [default]
```

Default copy assignment operator.

##### Returns

Reference to the current instance

#### 7.5.3.8 operator=() [2/2]

```
KeyEncapsulation& oqs::KeyEncapsulation::operator= (
    KeyEncapsulation && rhs ) [inline]
```

Move assignment operator, guarantees that the rvalue secret key is always zeroed.

##### Parameters

<i>rhs</i>	<code>oqs::KeyEncapsulation</code> in- stance
------------	---

##### Returns

Reference to the current instance

### 7.5.4 Friends And Related Function Documentation

#### 7.5.4.1 operator<< [1/2]

```
std::ostream& operator<< (
    std::ostream & os,
    const alg_details_ & rhs ) [friend]
```

std::ostream extraction operator for the KEM algorithm details

##### Parameters

<i>os</i>	Output stream
<i>rhs</i>	Algorithm details in-stance

##### Returns

Reference to the output stream

#### 7.5.4.2 operator<< [2/2]

```
std::ostream& operator<< (
    std::ostream & os,
    const KeyEncapsulation & rhs ) [friend]
```

std::ostream extraction operator for [oqs::KeyEncapsulation](#)

##### Parameters

<i>os</i>	Output stream
<i>rhs</i>	<a href="#">oqs::KeyEncapsulation</a> in-stance

##### Returns

Reference to the output stream

### 7.5.5 Member Data Documentation

#### 7.5.5.1 alg\_name\_

```
std::string oqs::KeyEncapsulation::alg_name_ [private]
```

cryptographic algorithm name

#### 7.5.5.2 details\_

```
struct oqs::KeyEncapsulation::alg\_details\_ oqs::KeyEncapsulation::details_ [private]
```

#### 7.5.5.3 kem\_

```
std::shared_ptr<C::OQS_KEM> oqs::KeyEncapsulation::kem_ [private]
```

**Initial value:**

```
{nullptr, [] (C::OQS_KEM* p) {
                                C::OQS_KEM_free(p);
}}
```

liboqs smart pointer to C::OQS\_KEM

**7.5.5.4 secret\_key\_**

```
bytes oqs::KeyEncapsulation::secret_key_ {} [private]
```

secret key

The documentation for this class was generated from the following file:

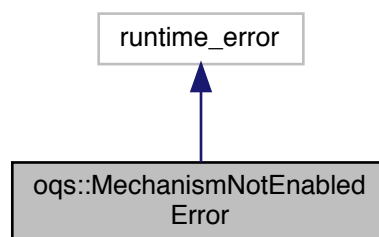
- [oqs\\_cpp.h](#)

**7.6 oqs::MechanismNotEnabledError Class Reference**

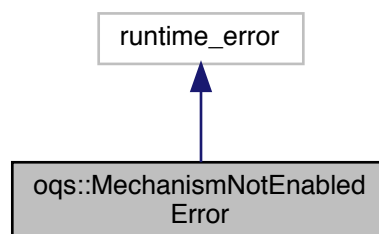
Cryptographic scheme not enabled.

```
#include <oqs_cpp.h>
```

Inheritance diagram for oqs::MechanismNotEnabledError:



Collaboration diagram for oqs::MechanismNotEnabledError:

**Public Member Functions**

- [MechanismNotEnabledError](#) (const std::string &alg\_name)  
*Constructor.*

### 7.6.1 Detailed Description

Cryptographic scheme not enabled.

### 7.6.2 Constructor & Destructor Documentation

#### 7.6.2.1 MechanismNotEnabledError()

```
oqs::MechanismNotEnabledError::MechanismNotEnabledError (
    const std::string & alg_name ) [inline], [explicit]
```

Constructor.

##### Parameters

<i>alg_name</i>	Cryptographic algo- rithm name
-----------------	---

The documentation for this class was generated from the following file:

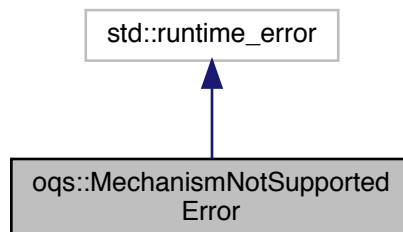
- [oqs\\_cpp.h](#)

## 7.7 oqs::MechanismNotSupportedError Class Reference

Cryptographic scheme not supported.

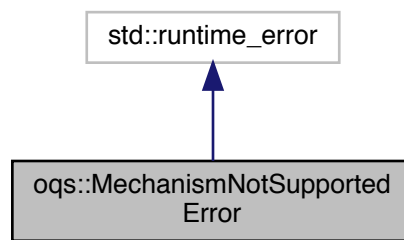
```
#include <oqs_cpp.h>
```

Inheritance diagram for oqs::MechanismNotSupportedError:





Collaboration diagram for oqs::MechanismNotSupportedError:



## Public Member Functions

- [MechanismNotSupportedError](#) (const std::string &alg\_name)  
*Constructor.*

### 7.7.1 Detailed Description

Cryptographic scheme not supported.

### 7.7.2 Constructor & Destructor Documentation

#### 7.7.2.1 MechanismNotSupportedError()

```
oqs::MechanismNotSupportedError::MechanismNotSupportedError (
    const std::string & alg_name ) [inline], [explicit]
```

Constructor.

#### Parameters

<i>alg_name</i>	Cryptographic algo- rithm name
-----------------	---

The documentation for this class was generated from the following file:

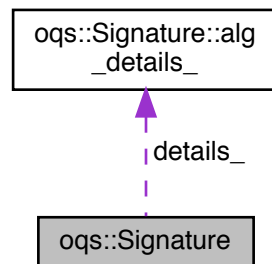
- [oqs\\_cpp.h](#)

## 7.8 oqs::Signature Class Reference

[Signature](#) mechanisms.

```
#include <oqs_cpp.h>
```

Collaboration diagram for `oqs::Signature`:



## Classes

- struct `alg_details_`  
*Signature algorithm details.*

## Public Member Functions

- `Signature` (const std::string &alg\_name, const bytes &secret\_key={})  
*Constructs an instance of `oqs::Signature`.*
- `Signature` (const `Signature` &)=default  
*Default copy constructor.*
- `Signature` & `operator=` (const `Signature` &)=default  
*Default copy assignment operator.*
- `Signature` (`Signature` &&rhs)  
*Move constructor, guarantees that the rvalue secret key is always zeroed.*
- `Signature` & `operator=` (`Signature` &&rhs)  
*Move assignment operator, guarantees that the rvalue secret key is always zeroed.*
- virtual `~Signature` ()  
*Virtual default destructor.*
- const `alg_details_` & `get_details` () const &  
*Signature algorithm details, lvalue overload.*
- `alg_details_` `get_details` () const &&  
*Signature algorithm details, rvalue overload.*
- bytes `generate_keypair` ()  
*Generate public key/secret key pair.*
- bytes `export_secret_key` () const  
*Export secret key.*
- bytes `sign` (const bytes &message) const  
*Sign message.*
- bool `verify` (const bytes &message, const bytes &signature, const bytes &public\_key) const  
*Verify signature.*

## Private Attributes

- `std::string alg_name_`  
*cryptographic algorithm name*
- `std::shared_ptr< C::OQS_SIG > sig_`  
*liboqs smart pointer to C::OQS\_SIG*
- `bytes secret_key_ {}`  
*secret key*
- `struct oqs::Signature::alg_details_ details_`

## Friends

- `std::ostream & operator<< (std::ostream &os, const alg_details_ &rhs)`  
*std::ostream extraction operator for the signature algorithm details*
- `std::ostream & operator<< (std::ostream &os, const Signature &rhs)`  
*std::ostream extraction operator for oqs::Signature*

## 7.8.1 Detailed Description

[Signature](#) mechanisms.

## 7.8.2 Constructor & Destructor Documentation

### 7.8.2.1 Signature() [1/3]

```
oqs::Signature::Signature (
    const std::string & alg_name,
    const bytes & secret_key = {} ) [inline], [explicit]
```

Constructs an instance of [oqs::Signature](#).

#### Parameters

<i>alg_name</i>	Cryptographic algo- rithm name
<i>secret_key</i>	Secret key (op- tional)

### 7.8.2.2 Signature() [2/3]

```
oqs::Signature::Signature (
    const Signature & ) [default]
```

Default copy constructor.

### 7.8.2.3 Signature() [3/3]

```
oqs::Signature::Signature (
    Signature && rhs ) [inline]
```

Move constructor, guarantees that the rvalue secret key is always zeroed.

## Parameters

<i>rhs</i>	<a href="#">oqs::Signature</a> in- stance
------------	---

**7.8.2.4 ~Signature()**

`virtual oqs::Signature::~~Signature ( ) [inline], [virtual]`  
 Virtual default destructor.

**7.8.3 Member Function Documentation****7.8.3.1 export\_secret\_key()**

`bytes oqs::Signature::export_secret_key ( ) const [inline]`  
 Export secret key.

## Returns

Secret key

**7.8.3.2 generate\_keypair()**

`bytes oqs::Signature::generate_keypair ( ) [inline]`  
 Generate public key/secret key pair.

## Returns

Public key

**7.8.3.3 get\_details() [1/2]**

`const alg_details_& oqs::Signature::get_details ( ) const & [inline]`  
[Signature](#) algorithm details, lvalue overload.

## Returns

[Signature](#) algorithm details

**7.8.3.4 get\_details() [2/2]**

`alg_details_ oqs::Signature::get_details ( ) const && [inline]`  
[Signature](#) algorithm details, rvalue overload.

## Returns

[Signature](#) algorithm details

**7.8.3.5 operator=()** [1/2]

```
Signature& oqs::Signature::operator= (
    const Signature & ) [default]
```

Default copy assignment operator.

**Returns**

Reference to the current instance

**7.8.3.6 operator=()** [2/2]

```
Signature& oqs::Signature::operator= (
    Signature && rhs ) [inline]
```

Move assignment operator, guarantees that the rvalue secret key is always zeroed.

**Parameters**

<i>rhs</i>	<a href="#">oqs::Signature</a> in- stance
------------	---

**Returns**

Reference to the current instance

**7.8.3.7 sign()**

```
bytes oqs::Signature::sign (
    const bytes & message ) const [inline]
```

Sign message.

**Parameters**

<i>message</i>	Message
----------------	---------

**Returns**

Message signature

**7.8.3.8 verify()**

```
bool oqs::Signature::verify (
    const bytes & message,
    const bytes & signature,
    const bytes & public_key ) const [inline]
```

Verify signature.

**Parameters**

<i>message</i>	Message
<i>signature</i>	<a href="#">Signature</a>
<i>public_key</i>	Public key

**Returns**

True if the signature is valid, false otherwise

**7.8.4 Friends And Related Function Documentation****7.8.4.1 operator<< [1/2]**

```
std::ostream& operator<< (
    std::ostream & os,
    const alg_details_ & rhs ) [friend]
```

std::ostream extraction operator for the signature algorithm details

**Parameters**

<i>os</i>	Output stream
<i>rhs</i>	Algorithm details in-stance

**Returns**

Reference to the output stream

**7.8.4.2 operator<< [2/2]**

```
std::ostream& operator<< (
    std::ostream & os,
    const Signature & rhs ) [friend]
```

std::ostream extraction operator for [oqs::Signature](#)

**Parameters**

<i>os</i>	Output stream
<i>rhs</i>	<a href="#">oqs::Signature</a> in-stance

**Returns**

Reference to the output stream

**7.8.5 Member Data Documentation****7.8.5.1 alg\_name\_**

```
std::string oqs::Signature::alg_name_ [private]
```

cryptographic algorithm name

## 7.8.5.2 details\_

```
struct oqs::Signature::alg_details_ oqs::Signature::details_ [private]
```

## 7.8.5.3 secret\_key\_

```
bytes oqs::Signature::secret_key_ {} [private]
```

secret key

## 7.8.5.4 sig\_

```
std::shared_ptr<C::OQS_SIG> oqs::Signature::sig_ [private]
```

**Initial value:**

```
{nullptr, [] (C::OQS_SIG* p) {
                                C::OQS_SIG_free(p);
                                }}
```

liboqs smart pointer to C::OQS\_SIG

The documentation for this class was generated from the following file:

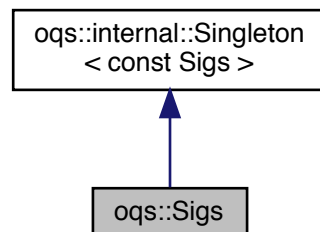
- [oqs\\_cpp.h](#)

## 7.9 oqs::Sigs Class Reference

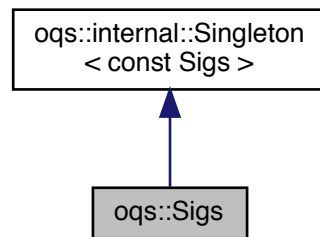
Singleton class, contains details about supported/enabled signature mechanisms.

```
#include <oqs_cpp.h>
```

Inheritance diagram for oqs::Sigs:



Collaboration diagram for `oqs::Sigs`:



## Static Public Member Functions

- static `std::size_t max_number_sigs ()`  
*Maximum number of supported signatures.*
- static `bool is_sig_supported (const std::string &alg_name)`  
*Checks whether the signature algorithm `alg_name` is supported.*
- static `bool is_sig_enabled (const std::string &alg_name)`  
*Checks whether the signature algorithm `alg_name` is enabled.*
- static `std::string get_sig_name (std::size_t alg_id)`  
*Signature algorithm name.*
- static `const std::vector< std::string > & get_supported_sigs ()`  
*Vector of supported signature algorithms.*
- static `const std::vector< std::string > & get_enabled_sigs ()`  
*Vector of enabled signature algorithms.*

## Private Member Functions

- `Sigs ()=default`  
*Private default constructor.*

## Friends

- class `internal::Singleton< const Sigs >`

## Additional Inherited Members

### 7.9.1 Detailed Description

Singleton class, contains details about supported/enabled signature mechanisms.

### 7.9.2 Constructor & Destructor Documentation



### 7.9.2.1 Sigs()

`oqs::Sigs::Sigs ( ) [private], [default]`  
Private default constructor.

#### Note

Use [oqs::Sigs::get\\_instance\(\)](#) to create an instance

## 7.9.3 Member Function Documentation

### 7.9.3.1 get\_enabled\_sigs()

`static const std::vector<std::string>& oqs::Sigs::get_enabled_sigs ( ) [inline], [static]`  
Vector of enabled signature algorithms.

#### Returns

Vector of enabled signature algorithms

### 7.9.3.2 get\_sig\_name()

`static std::string oqs::Sigs::get_sig_name (`  
`std::size_t alg_id ) [inline], [static]`

[Signature](#) algorithm name.

#### Parameters

<i>alg_id</i>	Cryptographic algo- rithm numer- ical id
---------------	---

#### Returns

[Signature](#) algorithm name

### 7.9.3.3 get\_supported\_sigs()

`static const std::vector<std::string>& oqs::Sigs::get_supported_sigs ( ) [inline], [static]`  
Vector of supported signature algorithms.

#### Returns

Vector of supported signature algorithms

### 7.9.3.4 is\_sig\_enabled()

`static bool oqs::Sigs::is_sig_enabled (`  
`const std::string & alg_name ) [inline], [static]`

Checks whether the signature algorithm *alg\_name* is enabled.

**Parameters**

<i>alg_name</i>	Cryptographic algo- rithm name
-----------------	---

**Returns**

True if the signature algorithm is enabled, false otherwise

**7.9.3.5 is\_sig\_supported()**

```
static bool oqs::Sigs::is_sig_supported (
    const std::string & alg_name ) [inline], [static]
```

Checks whether the signature algorithm *alg\_name* is supported.

**Parameters**

<i>alg_name</i>	Cryptographic algo- rithm name
-----------------	---

**Returns**

True if the signature algorithm is supported, false otherwise

**7.9.3.6 max\_number\_sigs()**

```
static std::size_t oqs::Sigs::max_number_sigs ( ) [inline], [static]
```

Maximum number of supported signatures.

**Returns**

Maximum number of supported signatures

**7.9.4 Friends And Related Function Documentation****7.9.4.1 internal::Singleton< const Sigs >**

```
friend class internal::Singleton< const Sigs > [friend]
```

The documentation for this class was generated from the following file:

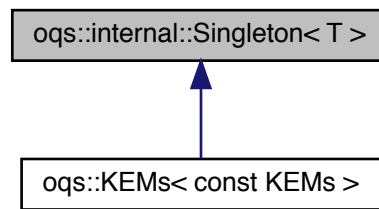
- [oqs\\_cpp.h](#)

**7.10 oqs::internal::Singleton< T > Class Template Reference**

[Singleton](#) class using CRTP pattern.

```
#include <oqs_cpp.h>
```

Inheritance diagram for oqs::internal::Singleton< T >:



### Static Public Member Functions

- static T & [get\\_instance](#) () noexcept(std::is\_nothrow\_constructible< T >::value)  
*Singleton instance (thread-safe) via CRTP pattern.*

### Protected Member Functions

- [Singleton](#) () noexcept=default
- [Singleton](#) (const [Singleton](#) &)=delete
- [Singleton](#) & [operator=](#) (const [Singleton](#) &)=delete
- virtual [~Singleton](#) ()=default

#### 7.10.1 Detailed Description

```
template<typename T>
class oqs::internal::Singleton< T >
```

[Singleton](#) class using CRTP pattern.

##### Note

Code from <https://github.com/vsoftco/qpp/blob/master/include/internal/classes/singleton.h>

##### Template Parameters

<i>T</i>	Class type of which instance will become a <a href="#">Singleton</a>
----------	--

#### 7.10.2 Constructor & Destructor Documentation

##### 7.10.2.1 Singleton() [1/2]

```
template<typename T>
oqs::internal::Singleton< T >::Singleton ( ) [protected], [default], [noexcept]
```

##### 7.10.2.2 Singleton() [2/2]

```
template<typename T>
```

```
oqs::internal::Singleton< T >::Singleton (
    const Singleton< T > & ) [protected], [delete]
```

### 7.10.2.3 ~Singleton()

```
template<typename T>
virtual oqs::internal::Singleton< T >::~~Singleton ( ) [protected], [virtual], [default]
```

## 7.10.3 Member Function Documentation

### 7.10.3.1 get\_instance()

```
template<typename T>
static T& oqs::internal::Singleton< T >::get_instance ( ) [inline], [static], [noexcept]
```

[Singleton](#) instance (thread-safe) via CRTP pattern.

#### Returns

[Singleton](#) instance

### 7.10.3.2 operator=()

```
template<typename T>
Singleton& oqs::internal::Singleton< T >::operator= (
    const Singleton< T > & ) [protected], [delete]
```

The documentation for this class was generated from the following file:

- [oqs\\_cpp.h](#)

## 7.11 oqs::Timer< T, CLOCK\_T > Class Template Reference

High resolution timer.

```
#include <oqs_cpp.h>
```

### Public Member Functions

- [Timer](#) () noexcept  
*Constructs an instance with the current time as the start point.*
- [Timer](#) & [tic](#) () noexcept  
*Resets the chronometer.*
- [Timer](#) & [toc](#) () &noexcept  
*Stops the chronometer.*
- double [tics](#) () const noexcept  
*Time passed in the duration specified by T.*
- template<typename U = T>  
U [get\\_duration](#) () const noexcept  
*Duration specified by U.*
- virtual [~Timer](#) ()=default  
*Default virtual destructor.*

### Protected Attributes

- CLOCK\_T::time\_point [start\\_](#)
- CLOCK\_T::time\_point [end\\_](#)

## Friends

- std::ostream & [operator<<](#) (std::ostream &os, const [Timer](#) &rhs)

### 7.11.1 Detailed Description

```
template<typename T = std::chrono::duration<double>, typename CLOCK_T = std::chrono::steady_clock>
class oqs::Timer< T, CLOCK_T >
```

High resolution timer.

#### Note

Code from <https://github.com/vsoftco/qpp/blob/master/include/classes/timer.h>

#### Template Parameters

<i>T</i>	Tics duration, default is std::chrono::duration<double>, i.e. seconds in double precision
<i>CLOCK_T</i>	Clock's type, default is std::chrono::steady_clock, not affected by wall clock changes during runtime

### 7.11.2 Constructor & Destructor Documentation

#### 7.11.2.1 Timer()

```
template<typename T = std::chrono::duration<double>, typename CLOCK_T = std::chrono::steady_clock>
```

```
oqs::Timer< T, CLOCK_T >::Timer ( ) [inline], [noexcept]
```

Constructs an instance with the current time as the start point.

#### 7.11.2.2 ~Timer()

```
template<typename T = std::chrono::duration<double>, typename CLOCK_T = std::chrono::steady_clock>
```

```
virtual oqs::Timer< T, CLOCK_T >::~~Timer ( ) [virtual], [default]
```

Default virtual destructor.

### 7.11.3 Member Function Documentation

#### 7.11.3.1 get\_duration()

```
template<typename T = std::chrono::duration<double>, typename CLOCK_T = std::chrono::steady_clock>
```

```
template<typename U = T>
```

```
U oqs::Timer< T, CLOCK_T >::get_duration ( ) const [inline], [noexcept]
```

Duration specified by U.

#### Template Parameters

<i>U</i>	Duration, default is T, which defaults to std::chrono::duration<double>, i.e. seconds in double precision
----------	---

**Returns**

Duration that passed between the instantiation/reset and invocation of [oqs::Timer::toc\(\)](#)

**7.11.3.2 tic()**

```
template<typename T = std::chrono::duration<double>, typename CLOCK_T = std::chrono::steady↵_clock>
```

```
Timer& oqs::Timer< T, CLOCK_T >::tic ( ) [inline], [noexcept]
```

Resets the chronometer.

Resets the start/end point to the current time

**Returns**

Reference to the current instance

**7.11.3.3 tics()**

```
template<typename T = std::chrono::duration<double>, typename CLOCK_T = std::chrono::steady↵_clock>
```

```
double oqs::Timer< T, CLOCK_T >::tics ( ) const [inline], [noexcept]
```

Time passed in the duration specified by T.

**Returns**

Number of tics (specified by T) that passed between the instantiation/reset and invocation of [oqs::Timer::toc\(\)](#)

**7.11.3.4 toc()**

```
template<typename T = std::chrono::duration<double>, typename CLOCK_T = std::chrono::steady↵_clock>
```

```
Timer& oqs::Timer< T, CLOCK_T >::toc ( ) & [inline], [noexcept]
```

Stops the chronometer.

Set the current time as the end point

**Returns**

Reference to the current instance

**7.11.4 Friends And Related Function Documentation****7.11.4.1 operator<<**

```
template<typename T = std::chrono::duration<double>, typename CLOCK_T = std::chrono::steady↵_clock>
```

```
std::ostream& operator<< (
    std::ostream & os,
    const Timer< T, CLOCK_T > & rhs ) [friend]
```

**7.11.5 Member Data Documentation**

### 7.11.5.1 end\_

```
template<typename T = std::chrono::duration<double>, typename CLOCK_T = std::chrono::steady↵_clock>
CLOCK_T::time_point oqs::Timer< T, CLOCK_T >::end_ [protected]
```

### 7.11.5.2 start\_

```
template<typename T = std::chrono::duration<double>, typename CLOCK_T = std::chrono::steady↵_clock>
CLOCK_T::time_point oqs::Timer< T, CLOCK_T >::start_ [protected]
```

The documentation for this class was generated from the following file:

- [oqs\\_cpp.h](#)





## Chapter 8

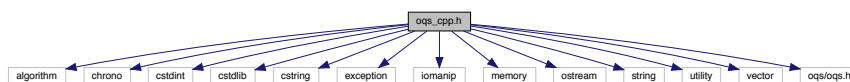
# File Documentation

### 8.1 oqs\_cpp.h File Reference

Main header file for the liboqs C++ wrapper.

```
#include <algorithm>
#include <chrono>
#include <cstdint>
#include <cstdlib>
#include <cstring>
#include <exception>
#include <iomanip>
#include <memory>
#include <ostream>
#include <string>
#include <utility>
#include <vector>
#include <oqs/oqs.h>
```

Include dependency graph for oqs\_cpp.h:



### Classes

- class [oqs::internal::Singleton< T >](#)  
*Singleton class using CRTP pattern.*
- class [oqs::internal::HexChop](#)  
*std::ostream manipulator for long vectors of [oqs::byte](#), use it to display only a small number of elements from the beginning and end of the vector*
- class [oqs::Timer< T, CLOCK\\_T >](#)  
*High resolution timer.*
- class [oqs::MechanismNotSupportedError](#)  
*Cryptographic scheme not supported.*
- class [oqs::MechanismNotEnabledError](#)  
*Cryptographic scheme not enabled.*
- class [oqs::KEMs](#)  
*Singleton class, contains details about supported/enabled key exchange mechanisms ([KEMs](#))*
- class [oqs::KeyEncapsulation](#)

- *Key encapsulation mechanisms.*
- struct [oqs::KeyEncapsulation::alg\\_details\\_](#)  
*KEM algorithm details.*
- class [oqs::Sigs](#)  
*Singleton class, contains details about supported/enabled signature mechanisms.*
- class [oqs::Signature](#)  
*Signature mechanisms.*
- struct [oqs::Signature::alg\\_details\\_](#)  
*Signature algorithm details.*

## Namespaces

- [oqs](#)  
*Main namespace for the liboqs C++ wrapper.*
- [oqs::C](#)  
*Namespace containing all of the oqs C functions, so they do not pollute the oqs namespace.*
- [internal](#)  
*Internal implementation details.*
- [oqs::internal](#)
- [oqs\\_literals](#)

## Typedefs

- using [oqs::byte](#) = std::uint8\_t  
*byte (unsigned)*
- using [oqs::bytes](#) = std::vector< byte >  
*vector of bytes (unsigned)*
- using [oqs::OQS\\_STATUS](#) = C::OQS\_STATUS  
*bring OQS\_STATUS into the oqs namespace*

## Functions

- [internal::HexChop](#) [oqs::hex\\_chop](#) (const [oqs::bytes](#) &v, std::size\_t start=8, std::size\_t end=8)  
*Constructs an instance of [oqs::internal::HexChop](#).*
- std::ostream & [operator<<](#) (std::ostream &os, const [oqs::bytes](#) &rhs)  
*std::ostream extraction operator for [oqs::bytes](#)*
- std::ostream & [operator<<](#) (std::ostream &os, const std::vector< std::string > &rhs)  
*std::ostream extraction operator for vectors of strings*
- [oqs::bytes](#) [oqs\\_literals::operator""\\_bytes](#) (const char \*c\_str, std::size\_t length)  
*User-defined literal operator for converting C-style strings to [oqs::bytes](#).*

### 8.1.1 Detailed Description

Main header file for the liboqs C++ wrapper.

### 8.1.2 Function Documentation

#### 8.1.2.1 [operator<<\(\)](#) [1/2]

```
std::ostream& operator<< (
    std::ostream & os,
    const oqs::bytes & rhs ) [inline]
std::ostream extraction operator for oqs::bytes
```

**Parameters**

<i>os</i>	Output stream
<i>rhs</i>	Vector of <a href="#">oqs::byte</a>

**Returns**

Reference to the output stream

**8.1.2.2 operator<<() [2/2]**

```
std::ostream& operator<< (
    std::ostream & os,
    const std::vector< std::string > & rhs ) [inline]
```

std::ostream extraction operator for vectors of strings

**Parameters**

<i>os</i>	Output stream
<i>rhs</i>	Vector of std↔ ::string

**Returns**

Reference to the output stream

**8.2 /Users/vlad/liboqs-cpp/README.md File Reference**



# Index

/Users/vlad/liboqs-cpp/README.md, [51](#)

~KeyEncapsulation

oqs::KeyEncapsulation, [28](#)

~Signature

oqs::Signature, [36](#)

~Singleton

oqs::internal::Singleton< T >, [44](#)

~Timer

oqs::Timer< T, CLOCK\_T >, [45](#)

alg\_name\_

oqs::KeyEncapsulation, [30](#)

oqs::Signature, [38](#)

byte

oqs, [14](#)

bytes

oqs, [14](#)

claimed\_nist\_level

oqs::KeyEncapsulation::alg\_details\_, [17](#)

oqs::Signature::alg\_details\_, [18](#)

decap\_secret

oqs::KeyEncapsulation, [28](#)

details\_

oqs::KeyEncapsulation, [30](#)

oqs::Signature, [38](#)

encap\_secret

oqs::KeyEncapsulation, [28](#)

end\_

oqs::internal::HexChop, [22](#)

oqs::Timer< T, CLOCK\_T >, [46](#)

export\_secret\_key

oqs::KeyEncapsulation, [28](#)

oqs::Signature, [36](#)

generate\_keypair

oqs::KeyEncapsulation, [28](#)

oqs::Signature, [36](#)

get\_details

oqs::KeyEncapsulation, [29](#)

oqs::Signature, [36](#)

get\_duration

oqs::Timer< T, CLOCK\_T >, [45](#)

get\_enabled\_KEMs

oqs::KEMs, [24](#)

get\_enabled\_sigs

oqs::Sigs, [41](#)

get\_instance

oqs::internal::Singleton< T >, [44](#)

get\_KEM\_name

oqs::KEMs, [24](#)

get\_sig\_name

oqs::Sigs, [41](#)

get\_supported\_KEMs

oqs::KEMs, [24](#)

get\_supported\_sigs

oqs::Sigs, [41](#)

hex\_chop

oqs, [14](#)

HexChop

oqs::internal::HexChop, [20](#)

internal, [13](#)

internal::Singleton< const KEMs >

oqs::KEMs, [25](#)

internal::Singleton< const Sigs >

oqs::Sigs, [42](#)

is\_euf\_cma

oqs::Signature::alg\_details\_, [18](#)

is\_ind\_cca

oqs::KeyEncapsulation::alg\_details\_, [17](#)

is\_KEM\_enabled

oqs::KEMs, [24](#)

is\_KEM\_supported

oqs::KEMs, [25](#)

is\_sig\_enabled

oqs::Sigs, [41](#)

is\_sig\_supported

oqs::Sigs, [42](#)

kem\_

oqs::KeyEncapsulation, [30](#)

KEMs

oqs::KEMs, [23](#)

KeyEncapsulation

oqs::KeyEncapsulation, [27](#)

length\_ciphertext

oqs::KeyEncapsulation::alg\_details\_, [17](#)

length\_public\_key

oqs::KeyEncapsulation::alg\_details\_, [17](#)

oqs::Signature::alg\_details\_, [18](#)

length\_secret\_key

oqs::KeyEncapsulation::alg\_details\_, [17](#)

oqs::Signature::alg\_details\_, [18](#)

length\_shared\_secret

oqs::KeyEncapsulation::alg\_details\_, [18](#)

- manipulate\_ostream\_
  - oqs::internal::HexChop, 20
- max\_length\_signature
  - oqs::Signature::alg\_details\_, 19
- max\_number\_KEMs
  - oqs::KEMs, 25
- max\_number\_sigs
  - oqs::Sigs, 42
- MechanismNotEnabledError
  - oqs::MechanismNotEnabledError, 32
- MechanismNotSupportedError
  - oqs::MechanismNotSupportedError, 33
- name
  - oqs::KeyEncapsulation::alg\_details\_, 18
  - oqs::Signature::alg\_details\_, 19
- operator<<
  - oqs::internal::HexChop, 21
  - oqs::KeyEncapsulation, 29, 30
  - oqs::Signature, 38
  - oqs::Timer< T, CLOCK\_T >, 46
  - oqs\_cpp.h, 50, 51
- operator=
  - oqs::internal::Singleton< T >, 44
  - oqs::KeyEncapsulation, 29
  - oqs::Signature, 36, 37
- operator""\_bytes
  - oqs\_literals, 16
- oqs, 13
  - byte, 14
  - bytes, 14
  - hex\_chop, 14
  - OQS\_STATUS, 14
- oqs::C, 15
- oqs::internal, 15
- oqs::internal::HexChop, 19
  - end\_, 22
  - HexChop, 20
  - manipulate\_ostream\_, 20
  - operator<<, 21
  - start\_, 22
  - v\_, 22
- oqs::internal::Singleton< T >, 42
  - ~Singleton, 44
  - get\_instance, 44
  - operator=, 44
  - Singleton, 43
- oqs::KEMs, 22
  - get\_enabled\_KEMs, 24
  - get\_KEM\_name, 24
  - get\_supported\_KEMs, 24
  - internal::Singleton< const KEMs >, 25
  - is\_KEM\_enabled, 24
  - is\_KEM\_supported, 25
  - KEMs, 23
  - max\_number\_KEMs, 25
- oqs::KeyEncapsulation, 25
  - ~KeyEncapsulation, 28
  - alg\_name\_, 30
  - decap\_secret, 28
  - details\_, 30
  - encap\_secret, 28
  - export\_secret\_key, 28
  - generate\_keypair, 28
  - get\_details, 29
  - kem\_, 30
  - KeyEncapsulation, 27
  - operator<<, 29, 30
  - operator=, 29
  - secret\_key\_, 31
- oqs::KeyEncapsulation::alg\_details\_, 17
  - claimed\_nist\_level, 17
  - is\_ind\_cca, 17
  - length\_ciphertext, 17
  - length\_public\_key, 17
  - length\_secret\_key, 17
  - length\_shared\_secret, 18
  - name, 18
  - version, 18
- oqs::MechanismNotEnabledError, 31
  - MechanismNotEnabledError, 32
- oqs::MechanismNotSupportedError, 32
  - MechanismNotSupportedError, 33
- oqs::Signature, 33
  - ~Signature, 36
  - alg\_name\_, 38
  - details\_, 38
  - export\_secret\_key, 36
  - generate\_keypair, 36
  - get\_details, 36
  - operator<<, 38
  - operator=, 36, 37
  - secret\_key\_, 39
  - sig\_, 39
  - sign, 37
  - Signature, 35
  - verify, 37
- oqs::Signature::alg\_details\_, 18
  - claimed\_nist\_level, 18
  - is\_euf\_cma, 18
  - length\_public\_key, 18
  - length\_secret\_key, 18
  - max\_length\_signature, 19
  - name, 19
  - version, 19
- oqs::Sigs, 39
  - get\_enabled\_sigs, 41
  - get\_sig\_name, 41
  - get\_supported\_sigs, 41
  - internal::Singleton< const Sigs >, 42
  - is\_sig\_enabled, 41
  - is\_sig\_supported, 42
  - max\_number\_sigs, 42
  - Sigs, 40
- oqs::Timer< T, CLOCK\_T >, 44
  - ~Timer, 45

- end\_, [46](#)
- get\_duration, [45](#)
- operator<<, [46](#)
- start\_, [47](#)
- tic, [46](#)
- tics, [46](#)
- Timer, [45](#)
- toc, [46](#)
- oqs\_cpp.h, [49](#)
  - operator<<, [50](#), [51](#)
- oqs\_literals, [15](#)
  - operator""\_bytes, [16](#)
- OQS\_STATUS
  - oqs, [14](#)
- secret\_key\_
  - oqs::KeyEncapsulation, [31](#)
  - oqs::Signature, [39](#)
- sig\_
  - oqs::Signature, [39](#)
- sign
  - oqs::Signature, [37](#)
- Signature
  - oqs::Signature, [35](#)
- Sigs
  - oqs::Sigs, [40](#)
- Singleton
  - oqs::internal::Singleton< T >, [43](#)
- start\_
  - oqs::internal::HexChop, [22](#)
  - oqs::Timer< T, CLOCK\_T >, [47](#)
- tic
  - oqs::Timer< T, CLOCK\_T >, [46](#)
- tics
  - oqs::Timer< T, CLOCK\_T >, [46](#)
- Timer
  - oqs::Timer< T, CLOCK\_T >, [45](#)
- toc
  - oqs::Timer< T, CLOCK\_T >, [46](#)
- v\_
  - oqs::internal::HexChop, [22](#)
- verify
  - oqs::Signature, [37](#)
- version
  - oqs::KeyEncapsulation::alg\_details\_, [18](#)
  - oqs::Signature::alg\_details\_, [19](#)