

liboqs-cpp

0.1

Generated by Doxygen 1.8.14



# Contents

<b>1</b>	<b>liboqs-cpp</b>	<b>1</b>
<b>2</b>	<b>Namespace Index</b>	<b>3</b>
2.1	Namespace List . . . . .	3
<b>3</b>	<b>Hierarchical 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>File Index</b>	<b>9</b>
5.1	File List . . . . .	9
<b>6</b>	<b>Namespace Documentation</b>	<b>11</b>
6.1	impl_details_ Namespace Reference . . . . .	11
6.1.1	Detailed Description . . . . .	11
6.2	oqs Namespace Reference . . . . .	11
6.2.1	Detailed Description . . . . .	12
6.2.2	Typedef Documentation . . . . .	12
6.2.2.1	byte . . . . .	12
6.2.2.2	bytes . . . . .	12
6.3	oqs::impl_details_ Namespace Reference . . . . .	12
6.4	oqs_literals Namespace Reference . . . . .	12
6.4.1	Function Documentation . . . . .	13
6.4.1.1	operator""_bytes() . . . . .	13

<b>7</b>	<b>Class Documentation</b>	<b>15</b>
7.1	oqs::KeyEncapsulation::alg_details_ Struct Reference	15
7.1.1	Detailed Description	15
7.1.2	Member Data Documentation	15
7.1.2.1	claimed_nist_level	15
7.1.2.2	is_ind_cca	16
7.1.2.3	length_ciphertext	16
7.1.2.4	length_public_key	16
7.1.2.5	length_secret_key	16
7.1.2.6	length_shared_secret	16
7.1.2.7	name	16
7.1.2.8	version	16
7.2	oqs::Signature::alg_details_ Struct Reference	17
7.2.1	Detailed Description	17
7.2.2	Member Data Documentation	17
7.2.2.1	claimed_nist_level	17
7.2.2.2	is_euf_cma	17
7.2.2.3	length_public_key	17
7.2.2.4	length_secret_key	17
7.2.2.5	length_signature	18
7.2.2.6	name	18
7.2.2.7	version	18
7.3	oqs::hex_chop Class Reference	18
7.3.1	Detailed Description	19
7.3.2	Constructor & Destructor Documentation	19
7.3.2.1	hex_chop()	19
7.3.3	Member Function Documentation	19
7.3.3.1	manipulate_ostream_()	19
7.3.4	Friends And Related Function Documentation	20
7.3.4.1	operator<<	20

7.3.5	Member Data Documentation . . . . .	20
7.3.5.1	end_ . . . . .	20
7.3.5.2	start_ . . . . .	20
7.3.5.3	v_ . . . . .	20
7.4	oqs::KEMs Class Reference . . . . .	21
7.4.1	Detailed Description . . . . .	22
7.4.2	Constructor & Destructor Documentation . . . . .	22
7.4.2.1	KEMs() . . . . .	22
7.4.3	Member Function Documentation . . . . .	22
7.4.3.1	get_enabled_KEMs() . . . . .	22
7.4.3.2	get_KEM_name() . . . . .	22
7.4.3.3	get_supported_KEMs() . . . . .	23
7.4.3.4	is_KEM_enabled() . . . . .	23
7.4.3.5	is_KEM_supported() . . . . .	23
7.4.3.6	max_number_KEMs() . . . . .	24
7.4.4	Friends And Related Function Documentation . . . . .	24
7.4.4.1	impl_details_::Singleton< const KEMs > . . . . .	24
7.5	oqs::KeyEncapsulation Class Reference . . . . .	24
7.5.1	Detailed Description . . . . .	25
7.5.2	Constructor & Destructor Documentation . . . . .	25
7.5.2.1	KeyEncapsulation() . . . . .	25
7.5.2.2	~KeyEncapsulation() . . . . .	26
7.5.3	Member Function Documentation . . . . .	26
7.5.3.1	decap_secret() . . . . .	26
7.5.3.2	encap_secret() . . . . .	26
7.5.3.3	export_secret_key() . . . . .	27
7.5.3.4	generate_keypair() . . . . .	27
7.5.3.5	get_details() . . . . .	27
7.5.4	Friends And Related Function Documentation . . . . .	27
7.5.4.1	operator<< [1/2] . . . . .	27

7.5.4.2	<a href="#">operator&lt;&lt; [2/2]</a>	28
7.5.5	<a href="#">Member Data Documentation</a>	28
7.5.5.1	<a href="#">alg_name_</a>	28
7.5.5.2	<a href="#">details_</a>	28
7.5.5.3	<a href="#">kem_</a>	29
7.5.5.4	<a href="#">secret_key_</a>	29
7.6	<a href="#">oqs::MechanismNotEnabledError Class Reference</a>	29
7.6.1	<a href="#">Detailed Description</a>	30
7.6.2	<a href="#">Constructor &amp; Destructor Documentation</a>	30
7.6.2.1	<a href="#">MechanismNotEnabledError()</a>	30
7.7	<a href="#">oqs::MechanismNotSupportedError Class Reference</a>	31
7.7.1	<a href="#">Detailed Description</a>	31
7.7.2	<a href="#">Constructor &amp; Destructor Documentation</a>	32
7.7.2.1	<a href="#">MechanismNotSupportedError()</a>	32
7.8	<a href="#">oqs::Signature Class Reference</a>	32
7.8.1	<a href="#">Detailed Description</a>	33
7.8.2	<a href="#">Constructor &amp; Destructor Documentation</a>	33
7.8.2.1	<a href="#">Signature()</a>	33
7.8.2.2	<a href="#">~Signature()</a>	34
7.8.3	<a href="#">Member Function Documentation</a>	34
7.8.3.1	<a href="#">export_secret_key()</a>	34
7.8.3.2	<a href="#">generate_keypair()</a>	34
7.8.3.3	<a href="#">get_details()</a>	34
7.8.3.4	<a href="#">sign()</a>	34
7.8.3.5	<a href="#">verify()</a>	35
7.8.4	<a href="#">Friends And Related Function Documentation</a>	35
7.8.4.1	<a href="#">operator&lt;&lt; [1/2]</a>	35
7.8.4.2	<a href="#">operator&lt;&lt; [2/2]</a>	36
7.8.5	<a href="#">Member Data Documentation</a>	36
7.8.5.1	<a href="#">alg_name_</a>	36

7.8.5.2	details_ . . . . .	36
7.8.5.3	secret_key_ . . . . .	36
7.8.5.4	sig_ . . . . .	37
7.9	oqs::Sigs Class Reference . . . . .	37
7.9.1	Detailed Description . . . . .	38
7.9.2	Constructor & Destructor Documentation . . . . .	38
7.9.2.1	Sigs() . . . . .	38
7.9.3	Member Function Documentation . . . . .	38
7.9.3.1	get_enabled_sigs() . . . . .	39
7.9.3.2	get_sig_name() . . . . .	39
7.9.3.3	get_supported_sigs() . . . . .	39
7.9.3.4	is_sig_enabled() . . . . .	39
7.9.3.5	is_sig_supported() . . . . .	40
7.9.3.6	max_number_sigs() . . . . .	40
7.9.4	Friends And Related Function Documentation . . . . .	40
7.9.4.1	impl_details_::Singleton< const Sigs > . . . . .	40
7.10	oqs::impl_details_::Singleton< T > Class Template Reference . . . . .	41
7.10.1	Detailed Description . . . . .	41
7.10.2	Constructor & Destructor Documentation . . . . .	42
7.10.2.1	Singleton() [1/2] . . . . .	42
7.10.2.2	Singleton() [2/2] . . . . .	42
7.10.2.3	~Singleton() . . . . .	42
7.10.3	Member Function Documentation . . . . .	42
7.10.3.1	get_instance() . . . . .	42
7.10.3.2	operator=() . . . . .	42
<b>8</b>	<b>File Documentation</b>	<b>43</b>
8.1	oqs_cpp.h File Reference . . . . .	43
8.1.1	Detailed Description . . . . .	44
8.1.2	Function Documentation . . . . .	44
8.1.2.1	operator<<() [1/2] . . . . .	44
8.1.2.2	operator<<() [2/2] . . . . .	45





# Chapter 1

## liboqs-cpp

### C++ bindings for liboqs

#### Build status:

**liboqs-cpp** offers a C++ wrapper for the [Open Quantum Safe liboqs](#) C library. The wrapper is written in standard C++11.

#### Contents

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

- `include/oqs_cpp.h`: the main header file
- `examples/kem.cpp`: key encapsulation example
- `examples/sig.cpp`: signature example
- `unit_tests`: unit tests written using Google Test (included)
- `doc`: Doxygen-generated detailed documentation

#### Usage

To avoid name collisions, liboqs-cpp includes all of its code inside the namespace `oqs`. 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 examples in `examples` file details the wrapper's API.

## liboqs installation

liboqs-cpp depends on the [liboqs](#) C library; liboqs must be compiled as a Linux/macOS static library or as a Windows DLL, and be visible to the wrapper, e.g. installed in a system-wide folder.

## Compiling on UNIX-like platforms

To use the wrapper, the user must have access to a C++11 compliant compiler, then simply `#include "oqs_+_cpp.h"` in her/his program. The wrapper contains a CMake build system for both examples and unit tests. To compile and run the examples, create a `build` folder inside the root folder of the project, change directory to `build`, then type

```
cmake ..; make -j;
```

The above commands build `oqs_cpp_kem` and `oqs_cpp_sig` examples, assuming the CMake build system is available on the user's platform.

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

```
cmake ..; make -j;
```

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

liboqs-cpp has been extensively tested on Linux and macOS systems. Continuous integration is provided via Travis CI.

## Compiling on Windows

A Visual Studio solution will be provided soon.

## 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.

## Chapter 2

# Namespace Index

### 2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">impl_details_</a>	Implementation details . . . . .	11
<a href="#">oqs</a>	Main namespace for the liboqs C++ wrapper . . . . .	11
<a href="#">oqs::impl_details_</a>	. . . . .	12
<a href="#">oqs_literals</a>	. . . . .	12



## Chapter 3

# Hierarchical Index

### 3.1 Class Hierarchy

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

oqs::KeyEncapsulation::alg_details_ . . . . .	15
oqs::Signature::alg_details_ . . . . .	17
oqs::hex_chop . . . . .	18
oqs::KeyEncapsulation . . . . .	24
runtime_error	
oqs::MechanismNotEnabledError . . . . .	29
oqs::MechanismNotSupportedError . . . . .	31
oqs::Signature . . . . .	32
oqs::impl_details_::Singleton< T > . . . . .	41
oqs::KEMs . . . . .	21
oqs::impl_details_::Singleton< const KEMs > . . . . .	41
oqs::impl_details_::Singleton< const Sigs > . . . . .	41
oqs::Sigs . . . . .	37



## 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	15
<a href="#">oqs::Signature::alg_details_</a>	
Signature algorithm details	17
<a href="#">oqs::hex_chop</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	18
<a href="#">oqs::KEMs</a>	
Singleton class, contains details about supported/enabled key exchange mechanisms ( <a href="#">KEMs</a> )	21
<a href="#">oqs::KeyEncapsulation</a>	
Key encapsulation mechanisms	24
<a href="#">oqs::MechanismNotEnabledError</a>	
Cryptographic scheme not enabled	29
<a href="#">oqs::MechanismNotSupportedError</a>	
Cryptographic scheme not supported	31
<a href="#">oqs::Signature</a>	
Signature mechanisms	32
<a href="#">oqs::Sigs</a>	
Singleton class, contains details about supported/enabled signature mechanisms	37
<a href="#">oqs::impl_details_::Singleton&lt; T &gt;</a>	
Singleton class using CRTP pattern	41





## 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 . . . . .	<a href="#">43</a>
---------------------------	---	--------------------



## Chapter 6

# Namespace Documentation

### 6.1 impl\_details\_ Namespace Reference

Implementation details.

#### 6.1.1 Detailed Description

Implementation details.

### 6.2 oqs Namespace Reference

Main namespace for the liboqs C++ wrapper.

#### Namespaces

- [impl\\_details\\_](#)

#### Classes

- class [hex\\_chop](#)  
*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 [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.*

## Typedefs

- using `byte` = `std::uint8_t`  
*byte (unsigned)*
- using `bytes` = `std::vector< byte >`  
*vector of bytes (unsigned)*

### 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)

## 6.3 `oqs::impl_details_` Namespace Reference

### Classes

- class `Singleton`  
*`Singleton` class using CRTP pattern.*

## 6.4 `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`.*

### 6.4.1 Function Documentation

#### 6.4.1.1 `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` [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 length\_signature

```
std::size_t oqs::Signature::alg_details_::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::hex\_chop 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

- [hex\\_chop](#) (const [oqs::bytes](#) &v, std::size\_t start=8, std::size\_t end=8)  
*Constructs an instance of [oqs::hex\\_chop](#).*

### 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\\_](#)
- std::size\_t [end\\_](#)  
*number of hex bytes taken from the start and from the end*

## Friends

- `std::ostream & operator<< (std::ostream &os, const hex_chop &rhs)`  
*std::ostream extraction operator for oqs::hex\_chop*

### 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 hex\_chop()

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

Constructs an instance of `oqs::hex_chop`.

#### 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::hex_chop::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
<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<<

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

std::ostream extraction operator for [oqs::hex\\_chop](#)

##### Parameters

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

##### Returns

Reference to the output stream

### 7.3.5 Member Data Documentation

#### 7.3.5.1 end\_

```
std::size_t oqs::hex_chop::end_ [private]
```

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

#### 7.3.5.2 start\_

```
std::size_t oqs::hex_chop::start_ [private]
```

#### 7.3.5.3 v\_

```
bytes oqs::hex_chop::v_ [private]
```

vector of bytes

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

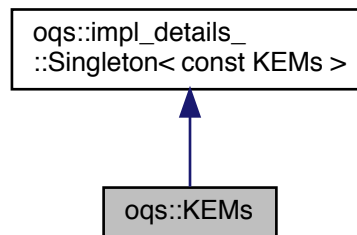
- [oqs\\_cpp.h](#)

## 7.4 oqs::KEMs Class Reference

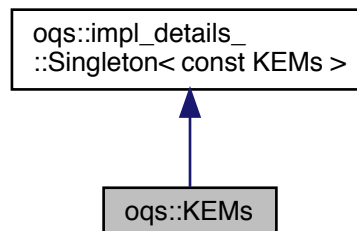
Singleton class, contains details about supported/enabled key exchange mechanisms ([KEMs](#))

```
#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 [impl\\_details\\_::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</i> ↔ _id	Cryptographic algorithm numerical 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 algorithm 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 algorithm 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 impl\_details\_::Singleton&lt; const KEMs &gt;

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

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

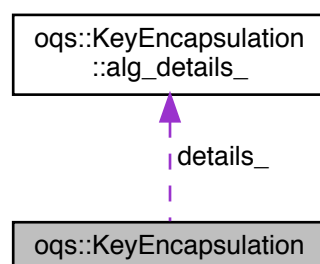
- [oqs\\_cpp.h](#)

## 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](#).*
- virtual [~KeyEncapsulation](#) ()  
*Virtual default destructor.*
- const [alg\\_details\\_](#) & [get\\_details](#) () const  
*KEM algorithm details.*
- [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

- const std::string [alg\\_name\\_](#)  
*cryptographic algorithm name*
- std::shared\_ptr< OQS\_KEM > [kem\\_](#)  
*liboqs smart pointer to 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()

```
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 algorithm name
<i>secret_key</i>	Secret key (optional)

## 7.5.2.2 ~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()**

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

KEM algorithm details.

**Returns**

KEM algorithm details

**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 instance

**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>	Key encapsulation instance

**Returns**

Reference to the output stream

**7.5.5 Member Data Documentation****7.5.5.1 alg\_name\_**

```
const 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<OQS_KEM> oqs::KeyEncapsulation::kem_ [private]
```

**Initial value:**

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

liboqs smart pointer to 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:

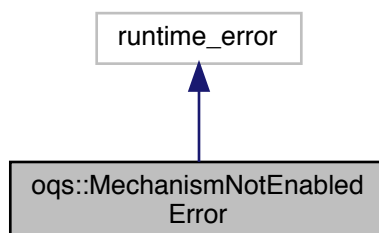
- [oqs\\_cpp.h](#)

## 7.6 oqs::MechanismNotEnabledError Class Reference

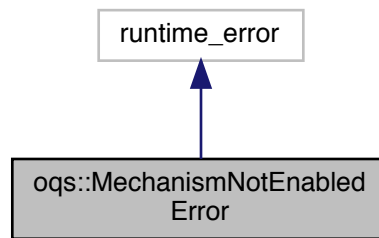
Cryptographic scheme not enabled.

```
#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

<code>alg_name</code>	Cryptographic algorithm name
-----------------------	------------------------------

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

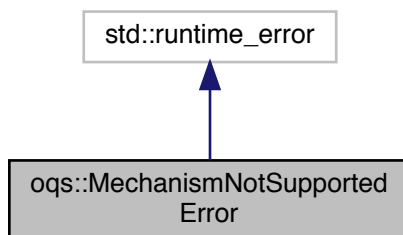
- [oqs\\_cpp.h](#)

## 7.7 oqs::MechanismNotSupportedError Class Reference

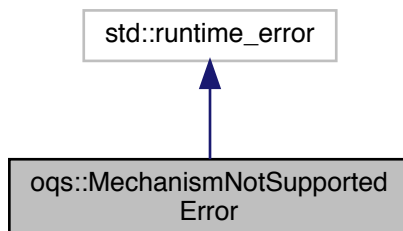
Cryptographic scheme not supported.

```
#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 algorithm name
-----------------	------------------------------

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

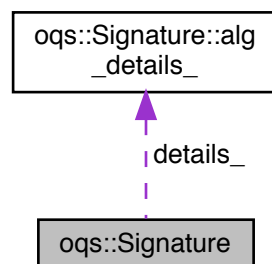
- [oqs\\_cpp.h](#)

## 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](#).*
- virtual [~Signature](#) ()  
*Virtual default destructor.*
- const [alg\\_details\\_](#) & [get\\_details](#) () const  
*[Signature](#) algorithm details.*
- [bytes](#) [generate\\_keypair](#) ()  
*Generate public key/secret key pair.*
- [bytes](#) [export\\_secret\\_key](#) () const  
*Export secret key.*
- [bytes](#) [sign](#) (const [bytes](#) &message)  
*Sign message.*
- bool [verify](#) (const [bytes](#) &message, const [bytes](#) &signature, const [bytes](#) &public\_key)  
*Verify signature.*

## Private Attributes

- const std::string [alg\\_name\\_](#)  
*cryptographic algorithm name*
- std::shared\_ptr< OQS\_SIG > [sig\\_](#)  
*liboqs smart pointer to 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()

```
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 algorithm name
<i>secret_key</i>	Secret key (optional)

## 7.8.2.2 ~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()

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

[Signature](#) algorithm details.

## Returns

[Signature](#) algorithm details

## 7.8.3.4 sign()

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

Sign message.

## Parameters

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

## Returns

Message signature

## 7.8.3.5 verify()

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

**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="#">Signature</a> instance

**Returns**

Reference to the output stream

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

```
const 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<OQS_SIG> oqs::Signature::sig_ [private]
```

**Initial value:**

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

liboqs smart pointer to OQS\_SIG

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

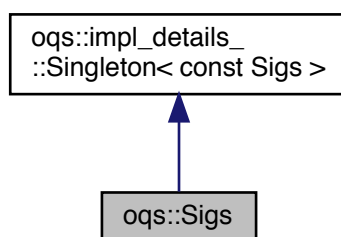
- [oqs\\_cpp.h](#)

## 7.9 oqs::Sigs Class Reference

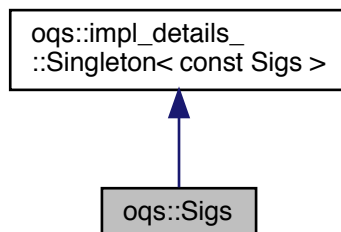
Singleton class, contains details about supported/enabled signature mechanisms.

```
#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 `impl_details_::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</i> ↔ _id	Cryptographic algorithm numerical 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 algorithm 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 algorithm 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 impl\_details::Singleton< const Sigs >**

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

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

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

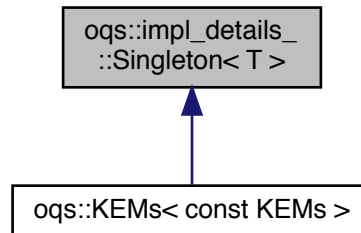


## 7.10 oqs::impl\_details\_::Singleton< T > Class Template Reference

[Singleton](#) class using CRTP pattern.

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

Inheritance diagram for oqs::impl\_details\_::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::impl_details_::Singleton< T >
```

[Singleton](#) class using CRTP pattern.

#### 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::impl_details_::Singleton< T >::Singleton ( ) [protected], [default], [noexcept]
```

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

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

### 7.10.2.3 ~Singleton()

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

## 7.10.3 Member Function Documentation

### 7.10.3.1 get\_instance()

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

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

#### Note

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

#### Returns

[Singleton](#) instance

### 7.10.3.2 operator=()

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

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

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

## Chapter 8

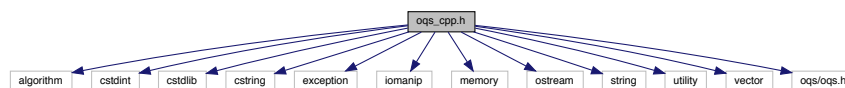
# File Documentation

### 8.1 oqs\_cpp.h File Reference

Main header file for the liboqs C++ wrapper.

```
#include <algorithm>
#include <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::impl_details::Singleton< T >`  
*Singleton class using CRTP pattern.*
- 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.*
- class [oqs::hex\\_chop](#)  
*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*

## Namespaces

- [oqs](#)  
*Main namespace for the liboqs C++ wrapper.*
- [impl\\_details\\_](#)  
*Implementation details.*
- [oqs::impl\\_details\\_](#)
- [oqs\\_literals](#)

## Typedefs

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

## Functions

- 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



# Index

- ~KeyEncapsulation
  - oqs::KeyEncapsulation, [26](#)
- ~Signature
  - oqs::Signature, [34](#)
- ~Singleton
  - oqs::impl\_details\_::Singleton, [42](#)
- alg\_name\_
  - oqs::KeyEncapsulation, [28](#)
  - oqs::Signature, [36](#)
- byte
  - oqs, [12](#)
- bytes
  - oqs, [12](#)
- claimed\_nist\_level
  - oqs::KeyEncapsulation::alg\_details\_, [15](#)
  - oqs::Signature::alg\_details\_, [17](#)
- decap\_secret
  - oqs::KeyEncapsulation, [26](#)
- details\_
  - oqs::KeyEncapsulation, [28](#)
  - oqs::Signature, [36](#)
- encap\_secret
  - oqs::KeyEncapsulation, [26](#)
- end\_
  - oqs::hex\_chop, [20](#)
- export\_secret\_key
  - oqs::KeyEncapsulation, [27](#)
  - oqs::Signature, [34](#)
- generate\_keypair
  - oqs::KeyEncapsulation, [27](#)
  - oqs::Signature, [34](#)
- get\_KEM\_name
  - oqs::KEMs, [22](#)
- get\_details\_
  - oqs::KeyEncapsulation, [27](#)
  - oqs::Signature, [34](#)
- get\_enabled\_KEMs
  - oqs::KEMs, [22](#)
- get\_enabled\_sigs
  - oqs::Sigs, [38](#)
- get\_instance
  - oqs::impl\_details\_::Singleton, [42](#)
- get\_sig\_name
  - oqs::Sigs, [39](#)
- get\_supported\_KEMs
  - oqs::KEMs, [23](#)
- get\_supported\_sigs
  - oqs::Sigs, [39](#)
- hex\_chop
  - oqs::hex\_chop, [19](#)
- impl\_details\_, [11](#)
- impl\_details\_::Singleton< const KEMs >
  - oqs::KEMs, [24](#)
- impl\_details\_::Singleton< const Sigs >
  - oqs::Sigs, [40](#)
- is\_KEM\_enabled
  - oqs::KEMs, [23](#)
- is\_KEM\_supported
  - oqs::KEMs, [23](#)
- is\_euf\_cma
  - oqs::Signature::alg\_details\_, [17](#)
- is\_ind\_cca
  - oqs::KeyEncapsulation::alg\_details\_, [15](#)
- is\_sig\_enabled
  - oqs::Sigs, [39](#)
- is\_sig\_supported
  - oqs::Sigs, [40](#)
- KEMs
  - oqs::KEMs, [22](#)
- kem\_
  - oqs::KeyEncapsulation, [28](#)
- KeyEncapsulation
  - oqs::KeyEncapsulation, [25](#)
- length\_ciphertext
  - oqs::KeyEncapsulation::alg\_details\_, [16](#)
- length\_public\_key
  - oqs::KeyEncapsulation::alg\_details\_, [16](#)
  - oqs::Signature::alg\_details\_, [17](#)
- length\_secret\_key
  - oqs::KeyEncapsulation::alg\_details\_, [16](#)
  - oqs::Signature::alg\_details\_, [17](#)
- length\_shared\_secret
  - oqs::KeyEncapsulation::alg\_details\_, [16](#)
- length\_signature
  - oqs::Signature::alg\_details\_, [17](#)
- manipulate\_ostream\_
  - oqs::hex\_chop, [19](#)
- max\_number\_KEMs
  - oqs::KEMs, [24](#)
- max\_number\_sigs
  - oqs::Sigs, [40](#)

- MechanismNotEnabledError
  - oqs::MechanismNotEnabledError, 30
- MechanismNotSupportedError
  - oqs::MechanismNotSupportedError, 32
- name
  - oqs::KeyEncapsulation::alg\_details\_, 16
  - oqs::Signature::alg\_details\_, 18
- operator<<
  - oqs::KeyEncapsulation, 27, 28
  - oqs::Signature, 35, 36
  - oqs::hex\_chop, 20
  - oqs\_cpp.h, 44, 45
- operator=
  - oqs::impl\_details\_::Singleton, 42
- operator""\_bytes
  - oqs\_literals, 13
- oqs, 11
  - byte, 12
  - bytes, 12
- oqs::KEMs, 21
  - get\_KEM\_name, 22
  - get\_enabled\_KEMs, 22
  - get\_supported\_KEMs, 23
  - impl\_details\_::Singleton< const KEMs >, 24
  - is\_KEM\_enabled, 23
  - is\_KEM\_supported, 23
  - KEMs, 22
  - max\_number\_KEMs, 24
- oqs::KeyEncapsulation, 24
  - ~KeyEncapsulation, 26
  - alg\_name\_, 28
  - decap\_secret, 26
  - details\_, 28
  - encap\_secret, 26
  - export\_secret\_key, 27
  - generate\_keypair, 27
  - get\_details, 27
  - kem\_, 28
  - KeyEncapsulation, 25
  - operator<<, 27, 28
  - secret\_key\_, 29
- oqs::KeyEncapsulation::alg\_details\_, 15
  - claimed\_nist\_level, 15
  - is\_ind\_cca, 15
  - length\_ciphertext, 16
  - length\_public\_key, 16
  - length\_secret\_key, 16
  - length\_shared\_secret, 16
  - name, 16
  - version, 16
- oqs::MechanismNotEnabledError, 29
  - MechanismNotEnabledError, 30
- oqs::MechanismNotSupportedError, 31
  - MechanismNotSupportedError, 32
- oqs::Signature, 32
  - ~Signature, 34
  - alg\_name\_, 36
  - details\_, 36
  - export\_secret\_key, 34
  - generate\_keypair, 34
  - get\_details, 34
  - operator<<, 35, 36
  - secret\_key\_, 36
  - sig\_, 36
  - sign, 34
  - Signature, 33
  - verify, 35
- oqs::Signature::alg\_details\_, 17
  - claimed\_nist\_level, 17
  - is\_euf\_cma, 17
  - length\_public\_key, 17
  - length\_secret\_key, 17
  - length\_signature, 17
  - name, 18
  - version, 18
- oqs::Sigs, 37
  - get\_enabled\_sigs, 38
  - get\_sig\_name, 39
  - get\_supported\_sigs, 39
  - impl\_details\_::Singleton< const Sigs >, 40
  - is\_sig\_enabled, 39
  - is\_sig\_supported, 40
  - max\_number\_sigs, 40
  - Sigs, 38
- oqs::hex\_chop, 18
  - end\_, 20
  - hex\_chop, 19
  - manipulate\_ostream\_, 19
  - operator<<, 20
  - start\_, 20
  - v\_, 20
- oqs::impl\_details\_, 12
- oqs::impl\_details\_::Singleton
  - ~Singleton, 42
  - get\_instance, 42
  - operator=, 42
  - Singleton, 42
- oqs::impl\_details\_::Singleton< T >, 41
- oqs\_cpp.h, 43
  - operator<<, 44, 45
- oqs\_literals, 12
  - operator""\_bytes, 13
- secret\_key\_
  - oqs::KeyEncapsulation, 29
  - oqs::Signature, 36
- sig\_
  - oqs::Signature, 36
- sign
  - oqs::Signature, 34
- Signature
  - oqs::Signature, 33
- Sigs
  - oqs::Sigs, 38
- Singleton
  - oqs::impl\_details\_::Singleton, 42



---

start\_  
    oqs::hex\_chop, [20](#)

v\_  
    oqs::hex\_chop, [20](#)

verify  
    oqs::Signature, [35](#)

version  
    oqs::KeyEncapsulation::alg\_details\_, [16](#)  
    oqs::Signature::alg\_details\_, [18](#)