

liboqs-cpp

0.1

Generated by Doxygen 1.8.14

Contents

1	liboqs-cpp	1
2	Namespace Index	3
2.1	Namespace List	3
3	Hierarchical Index	5
3.1	Class Hierarchy	5
4	Class Index	7
4.1	Class List	7
5	File Index	9
5.1	File List	9
6	Namespace Documentation	11
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

7	Class Documentation	15
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::KEMs Class Reference	18
7.3.1	Detailed Description	19
7.3.2	Constructor & Destructor Documentation	20
7.3.2.1	KEMs()	20
7.3.3	Member Function Documentation	20
7.3.3.1	get_enabled_KEMs()	20
7.3.3.2	get_KEM_name()	20
7.3.3.3	get_supported_KEMs()	21

7.3.3.4	<code>is_KEM_enabled()</code>	21
7.3.3.5	<code>is_KEM_supported()</code>	21
7.3.3.6	<code>max_number_KEMs()</code>	22
7.3.4	Friends And Related Function Documentation	22
7.3.4.1	<code>impl_details_::Singleton< const KEMs ></code>	22
7.4	<code>oqs::KeyEncapsulation</code> Class Reference	22
7.4.1	Detailed Description	23
7.4.2	Constructor & Destructor Documentation	23
7.4.2.1	<code>KeyEncapsulation()</code>	23
7.4.2.2	<code>~KeyEncapsulation()</code>	24
7.4.3	Member Function Documentation	24
7.4.3.1	<code>decap_secret()</code>	24
7.4.3.2	<code>encap_secret()</code>	24
7.4.3.3	<code>export_secret_key()</code>	25
7.4.3.4	<code>generate_keypair()</code>	25
7.4.3.5	<code>get_details()</code>	25
7.4.4	Friends And Related Function Documentation	25
7.4.4.1	<code>operator<< [1/2]</code>	25
7.4.4.2	<code>operator<< [2/2]</code>	26
7.4.5	Member Data Documentation	26
7.4.5.1	<code>alg_name_</code>	26
7.4.5.2	<code>details_</code>	26
7.4.5.3	<code>kem_</code>	27
7.4.5.4	<code>secret_key_</code>	27
7.5	<code>oqs::MechanismNotEnabledError</code> Class Reference	27
7.5.1	Detailed Description	28
7.5.2	Constructor & Destructor Documentation	28
7.5.2.1	<code>MechanismNotEnabledError()</code>	28
7.6	<code>oqs::MechanismNotSupportedError</code> Class Reference	29
7.6.1	Detailed Description	29

7.6.2	Constructor & Destructor Documentation	30
7.6.2.1	MechanismNotSupportedError()	30
7.7	oqs::Signature Class Reference	30
7.7.1	Detailed Description	31
7.7.2	Constructor & Destructor Documentation	31
7.7.2.1	Signature()	31
7.7.2.2	~Signature()	32
7.7.3	Member Function Documentation	32
7.7.3.1	export_secret_key()	32
7.7.3.2	generate_keypair()	32
7.7.3.3	get_details()	32
7.7.3.4	sign()	32
7.7.3.5	verify()	33
7.7.4	Friends And Related Function Documentation	33
7.7.4.1	operator<< [1/2]	33
7.7.4.2	operator<< [2/2]	34
7.7.5	Member Data Documentation	34
7.7.5.1	alg_name_	34
7.7.5.2	details_	34
7.7.5.3	secret_key_	34
7.7.5.4	sig_	35
7.8	oqs::Sigs Class Reference	35
7.8.1	Detailed Description	36
7.8.2	Constructor & Destructor Documentation	36
7.8.2.1	Sigs()	36
7.8.3	Member Function Documentation	36
7.8.3.1	get_enabled_sigs()	37
7.8.3.2	get_sig_name()	37
7.8.3.3	get_supported_sigs()	37
7.8.3.4	is_sig_enabled()	37
7.8.3.5	is_sig_supported()	38
7.8.3.6	max_number_sigs()	38
7.8.4	Friends And Related Function Documentation	38
7.8.4.1	impl_details_::Singleton< const Sigs >	38
7.9	oqs::impl_details_::Singleton< T > Class Template Reference	39
7.9.1	Detailed Description	39
7.9.2	Constructor & Destructor Documentation	40
7.9.2.1	Singleton() [1/2]	40
7.9.2.2	Singleton() [2/2]	40
7.9.2.3	~Singleton()	40
7.9.3	Member Function Documentation	40
7.9.3.1	get_instance()	40
7.9.3.2	operator=()	40

8 File Documentation	41
8.1 oqs_cpp.h File Reference	41
8.1.1 Detailed Description	42
8.1.2 Function Documentation	42
8.1.2.1 operator<<() [1/2]	42
8.1.2.2 operator<<() [2/2]	43
Index	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:

- `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 it, 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:

impl_details_	Implementation details	11
oqs	Main namespace for the liboqs C++ wrapper	11
oqs::impl_details_	12
oqs_literals	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::KeyEncapsulation	22
runtime_error	
oqs::MechanismNotEnabledError	27
oqs::MechanismNotSupportedError	29
oqs::Signature	30
oqs::impl_details_::Singleton< T >	39
oqs::KEMs	18
oqs::impl_details_::Singleton< const KEMs >	39
oqs::impl_details_::Singleton< const Sigs >	39
oqs::Sigs	35

Chapter 4

Class Index

4.1 Class List

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

oqs::KeyEncapsulation::alg_details_	
KEM algorithm details	15
oqs::Signature::alg_details_	
Signature algorithm details	17
oqs::KEMs	
Singleton class, contains details about supported/enabled key exchange mechanisms (KEMs)	18
oqs::KeyEncapsulation	
Key encapsulation mechanisms	22
oqs::MechanismNotEnabledError	
Cryptographic scheme not enabled	27
oqs::MechanismNotSupportedError	
Cryptographic scheme not supported	29
oqs::Signature	
Signature mechanisms	30
oqs::Sigs	
Singleton class, contains details about supported/enabled signature mechanisms	35
oqs::impl_details_::Singleton< T >	
Singleton class using CRTP pattern	39

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

oqs_cpp.h	Main header file for the liboqs C++ wrapper	41
---------------------------	---	----

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 [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:

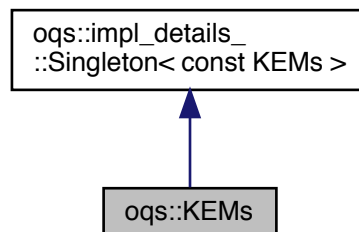
- [oqs_cpp.h](#)

7.3 oqs::KEMs Class Reference

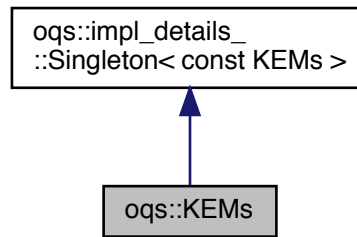
Singleton class, contains details about supported/enabled key exchange mechanisms ([KEMs](#))

```
#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

7.3.1 Detailed Description

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

7.3.2 Constructor & Destructor Documentation

7.3.2.1 KEMs()

```
oqs::KEMs::KEMs ( ) [private], [default]
```

Private default constructor.

Note

Use [oqs::KEMs::get_instance\(\)](#) to create an instance

7.3.3 Member Function Documentation

7.3.3.1 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

7.3.3.2 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 algorithm numerical id
-------------------------	--------------------------------------

Returns

KEM algorithm name

7.3.3.3 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

7.3.3.4 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

7.3.3.5 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

7.3.3.6 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](#)

7.3.4 Friends And Related Function Documentation

7.3.4.1 impl_details_::Singleton< const KEMs >

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

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

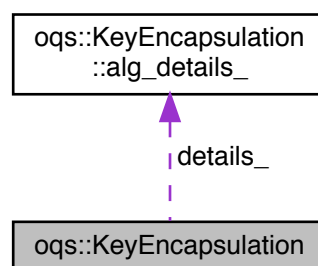
- [oqs_cpp.h](#)

7.4 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](#)

7.4.1 Detailed Description

Key encapsulation mechanisms.

7.4.2 Constructor & Destructor Documentation

7.4.2.1 KeyEncapsulation()

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

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

Parameters

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

7.4.2.2 ~KeyEncapsulation()

```
virtual oqs::KeyEncapsulation::~~KeyEncapsulation ( ) [inline], [virtual]
```

Virtual default destructor.

7.4.3 Member Function Documentation

7.4.3.1 decap_secret()

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

Decapsulate secret.

Parameters

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

Returns

Shared secret

7.4.3.2 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

7.4.3.3 export_secret_key()

```
bytes oqs::KeyEncapsulation::export_secret_key ( ) const [inline]
```

Export secret key.

Returns

Secret key

7.4.3.4 generate_keypair()

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

Generate public key/secret key pair.

Returns

Public key

7.4.3.5 get_details()

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

KEM algorithm details.

Returns

KEM algorithm details

7.4.4 Friends And Related Function Documentation**7.4.4.1 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

7.4.4.2 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

7.4.5 Member Data Documentation**7.4.5.1 alg_name_**

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

cryptographic algorithm name

7.4.5.2 details_

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

7.4.5.3 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

7.4.5.4 secret_key_

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

secret key

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

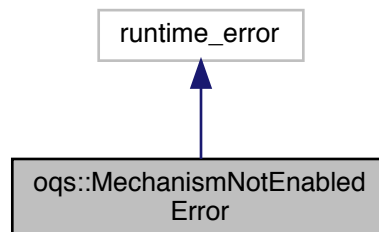
- [oqs_cpp.h](#)

7.5 oqs::MechanismNotEnabledError Class Reference

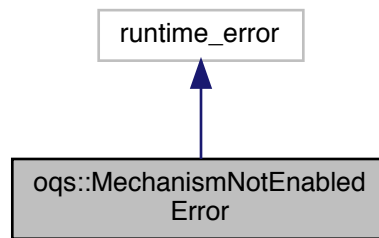
Cryptographic scheme not enabled.

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

Inheritance diagram for oqs::MechanismNotEnabledError:



Collaboration diagram for oqs::MechanismNotEnabledError:



Public Member Functions

- [MechanismNotEnabledError](#) (const std::string &alg_name)
Constructor.

7.5.1 Detailed Description

Cryptographic scheme not enabled.

7.5.2 Constructor & Destructor Documentation

7.5.2.1 MechanismNotEnabledError()

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

Constructor.

Parameters

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

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

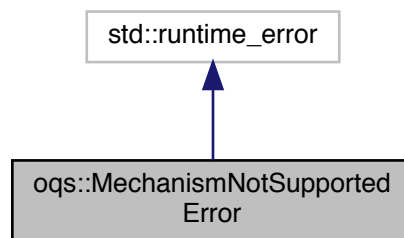
- [oqs_cpp.h](#)

7.6 oqs::MechanismNotSupportedError Class Reference

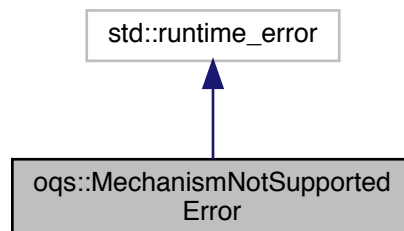
Cryptographic scheme not supported.

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

Inheritance diagram for oqs::MechanismNotSupportedError:



Collaboration diagram for oqs::MechanismNotSupportedError:



Public Member Functions

- [MechanismNotSupportedError](#) (const std::string &alg_name)
Constructor.

7.6.1 Detailed Description

Cryptographic scheme not supported.

7.6.2 Constructor & Destructor Documentation

7.6.2.1 MechanismNotSupportedError()

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

Constructor.

Parameters

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

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

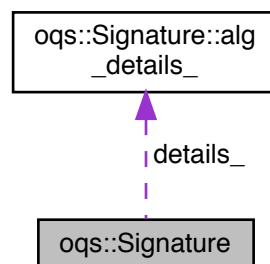
- [oqs_cpp.h](#)

7.7 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](#)

7.7.1 Detailed Description

[Signature](#) mechanisms.

7.7.2 Constructor & Destructor Documentation

7.7.2.1 Signature()

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

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

Parameters

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

7.7.2.2 ~Signature()

```
virtual oqs::Signature::~~Signature ( ) [inline], [virtual]
```

Virtual default destructor.

7.7.3 Member Function Documentation

7.7.3.1 export_secret_key()

```
bytes oqs::Signature::export_secret_key ( ) const [inline]
```

Export secret key.

Returns

Secret key

7.7.3.2 generate_keypair()

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

Generate public key/secret key pair.

Returns

Public key

7.7.3.3 get_details()

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

[Signature](#) algorithm details.

Returns

[Signature](#) algorithm details

7.7.3.4 sign()

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

Sign message.

Parameters

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

Returns

Message signature

7.7.3.5 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>	Signature
<i>public_key</i>	Public key

Returns

True if the signature is valid, false otherwise

7.7.4 Friends And Related Function Documentation

7.7.4.1 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

Returns

Reference to the output stream

7.7.4.2 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>	Signature instance

Returns

Reference to the output stream

7.7.5 Member Data Documentation**7.7.5.1 alg_name_**

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

cryptographic algorithm name

7.7.5.2 details_

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

7.7.5.3 secret_key_

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

secret key

7.7.5.4 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:

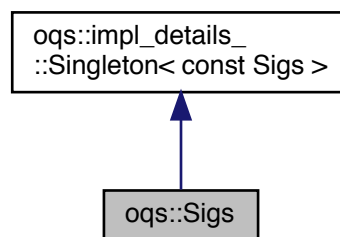
- [oqs_cpp.h](#)

7.8 oqs::Sigs Class Reference

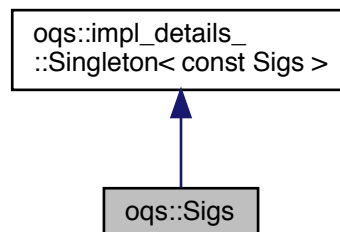
Singleton class, contains details about supported/enabled signature mechanisms.

```
#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

7.8.1 Detailed Description

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

7.8.2 Constructor & Destructor Documentation

7.8.2.1 Sigs()

```
oqs::Sigs::Sigs ( ) [private], [default]
```

Private default constructor.

Note

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

7.8.3 Member Function Documentation

7.8.3.1 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

7.8.3.2 get_sig_name()

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

[Signature](#) algorithm name.

Parameters

$alg \leftrightarrow$ _id	Cryptographic algorithm numerical id
------------------------------	--------------------------------------

Returns

[Signature](#) algorithm name

7.8.3.3 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

7.8.3.4 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

7.8.3.5 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

7.8.3.6 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

7.8.4 Friends And Related Function Documentation**7.8.4.1 impl_details::Singleton< const Sigs >**

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

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

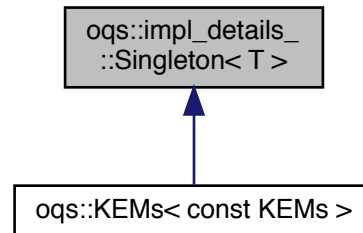
- [oqs_cpp.h](#)

7.9 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

7.9.1 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 Singleton
----------	--

7.9.2 Constructor & Destructor Documentation

7.9.2.1 Singleton() [1/2]

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

7.9.2.2 Singleton() [2/2]

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

7.9.2.3 ~Singleton()

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

7.9.3 Member Function Documentation

7.9.3.1 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

7.9.3.2 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

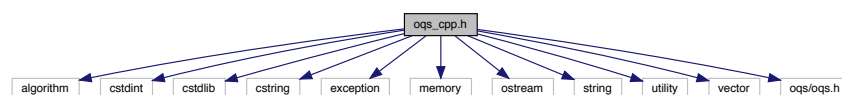
File Documentation

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 <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.

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 oqs::byte

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, [24](#)
- ~Signature
 - oqs::Signature, [32](#)
- ~Singleton
 - oqs::impl_details_::Singleton, [40](#)
- alg_name_
 - oqs::KeyEncapsulation, [26](#)
 - oqs::Signature, [34](#)
- byte
 - oqs, [12](#)
- bytes
 - oqs, [12](#)
- claimed_nist_level
 - oqs::KeyEncapsulation::alg_details_, [15](#)
 - oqs::Signature::alg_details_, [17](#)
- decap_secret
 - oqs::KeyEncapsulation, [24](#)
- details_
 - oqs::KeyEncapsulation, [26](#)
 - oqs::Signature, [34](#)
- encap_secret
 - oqs::KeyEncapsulation, [24](#)
- export_secret_key
 - oqs::KeyEncapsulation, [25](#)
 - oqs::Signature, [32](#)
- generate_keypair
 - oqs::KeyEncapsulation, [25](#)
 - oqs::Signature, [32](#)
- get_KEM_name
 - oqs::KEMs, [20](#)
- get_details_
 - oqs::KeyEncapsulation, [25](#)
 - oqs::Signature, [32](#)
- get_enabled_KEMs
 - oqs::KEMs, [20](#)
- get_enabled_sigs
 - oqs::Sigs, [36](#)
- get_instance
 - oqs::impl_details_::Singleton, [40](#)
- get_sig_name
 - oqs::Sigs, [37](#)
- get_supported_KEMs
 - oqs::KEMs, [20](#)
- get_supported_sigs
 - oqs::Sigs, [37](#)
- impl_details_, [11](#)
- impl_details_::Singleton< const KEMs >
 - oqs::KEMs, [22](#)
- impl_details_::Singleton< const Sigs >
 - oqs::Sigs, [38](#)
- is_KEM_enabled
 - oqs::KEMs, [21](#)
- is_KEM_supported
 - oqs::KEMs, [21](#)
- is_euf_cma
 - oqs::Signature::alg_details_, [17](#)
- is_ind_cca
 - oqs::KeyEncapsulation::alg_details_, [15](#)
- is_sig_enabled
 - oqs::Sigs, [37](#)
- is_sig_supported
 - oqs::Sigs, [38](#)
- KEMs
 - oqs::KEMs, [20](#)
- kem_
 - oqs::KeyEncapsulation, [26](#)
- KeyEncapsulation
 - oqs::KeyEncapsulation, [23](#)
- 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](#)
- max_number_KEMs
 - oqs::KEMs, [21](#)
- max_number_sigs
 - oqs::Sigs, [38](#)
- MechanismNotEnabledError
 - oqs::MechanismNotEnabledError, [28](#)
- MechanismNotSupportedError
 - oqs::MechanismNotSupportedError, [30](#)
- name
 - oqs::KeyEncapsulation::alg_details_, [16](#)

- oqs::Signature::alg_details_, 18
- operator<<
 - oqs::KeyEncapsulation, 25, 26
 - oqs::Signature, 33, 34
 - oqs_cpp.h, 42, 43
- operator=
 - oqs::impl_details_::Singleton, 40
- operator""_bytes
 - oqs_literals, 13
- oqs, 11
 - byte, 12
 - bytes, 12
- oqs::KEMs, 18
 - get_KEM_name, 20
 - get_enabled_KEMs, 20
 - get_supported_KEMs, 20
 - impl_details_::Singleton< const KEMs >, 22
 - is_KEM_enabled, 21
 - is_KEM_supported, 21
 - KEMs, 20
 - max_number_KEMs, 21
- oqs::KeyEncapsulation, 22
 - ~KeyEncapsulation, 24
 - alg_name_, 26
 - decap_secret, 24
 - details_, 26
 - encap_secret, 24
 - export_secret_key, 25
 - generate_keypair, 25
 - get_details, 25
 - kem_, 26
 - KeyEncapsulation, 23
 - operator<<, 25, 26
 - secret_key_, 27
- 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, 27
 - MechanismNotEnabledError, 28
- oqs::MechanismNotSupportedError, 29
 - MechanismNotSupportedError, 30
- oqs::Signature, 30
 - ~Signature, 32
 - alg_name_, 34
 - details_, 34
 - export_secret_key, 32
 - generate_keypair, 32
 - get_details, 32
 - operator<<, 33, 34
 - secret_key_, 34
 - sig_, 34
 - sign, 32
- Signature, 31
 - verify, 33
- 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, 35
 - get_enabled_sigs, 36
 - get_sig_name, 37
 - get_supported_sigs, 37
 - impl_details_::Singleton< const Sigs >, 38
 - is_sig_enabled, 37
 - is_sig_supported, 38
 - max_number_sigs, 38
 - Sigs, 36
- oqs::impl_details_, 12
- oqs::impl_details_::Singleton
 - ~Singleton, 40
 - get_instance, 40
 - operator=, 40
 - Singleton, 40
- oqs::impl_details_::Singleton< T >, 39
- oqs_cpp.h, 41
 - operator<<, 42, 43
- oqs_literals, 12
 - operator""_bytes, 13
- secret_key_
 - oqs::KeyEncapsulation, 27
 - oqs::Signature, 34
- sig_
 - oqs::Signature, 34
- sign
 - oqs::Signature, 32
- Signature
 - oqs::Signature, 31
- Sigs
 - oqs::Sigs, 36
- Singleton
 - oqs::impl_details_::Singleton, 40
- verify
 - oqs::Signature, 33
- version
 - oqs::KeyEncapsulation::alg_details_, 16
 - oqs::Signature::alg_details_, 18