

liboqs-cpp

0.2.0-rc1

Generated by Doxygen 1.8.15

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

7.2 oqs::Signature::alg_details_ Struct Reference	19
7.2.1 Detailed Description	19
7.2.2 Member Data Documentation	19
7.2.2.1 claimed_nist_level	19
7.2.2.2 is_euf_cma	19
7.2.2.3 length_public_key	19
7.2.2.4 length_secret_key	19
7.2.2.5 max_length_signature	20
7.2.2.6 name	20
7.2.2.7 version	20
7.3 oqs::internal::HexChop Class Reference	20
7.3.1 Detailed Description	21
7.3.2 Constructor & Destructor Documentation	21
7.3.2.1 HexChop()	21
7.3.3 Member Function Documentation	21
7.3.3.1 manipulate_ostream_()	21
7.3.4 Friends And Related Function Documentation	22
7.3.4.1 operator<<	22
7.3.5 Member Data Documentation	22
7.3.5.1 end_	22
7.3.5.2 start_	22
7.3.5.3 v_	22
7.4 oqs::KEMs Class Reference	23
7.4.1 Detailed Description	24
7.4.2 Constructor & Destructor Documentation	24
7.4.2.1 KEMs()	24
7.4.3 Member Function Documentation	24
7.4.3.1 get_enabled_KEMs()	24
7.4.3.2 get_KEM_name()	24
7.4.3.3 get_supported_KEMs()	25
7.4.3.4 is_KEM_enabled()	25
7.4.3.5 is_KEM_supported()	25
7.4.3.6 max_number_KEMs()	26
7.4.4 Friends And Related Function Documentation	26
7.4.4.1 internal::Singleton< const KEMs >	26
7.5 oqs::KeyEncapsulation Class Reference	26
7.5.1 Detailed Description	28
7.5.2 Constructor & Destructor Documentation	28
7.5.2.1 KeyEncapsulation() [1/3]	28
7.5.2.2 KeyEncapsulation() [2/3]	28
7.5.2.3 KeyEncapsulation() [3/3]	28
7.5.2.4 ~KeyEncapsulation()	29

7.5.3 Member Function Documentation	29
7.5.3.1 decap_secret()	29
7.5.3.2 encap_secret()	29
7.5.3.3 export_secret_key()	29
7.5.3.4 generate_keypair()	30
7.5.3.5 get_details() [1/2]	30
7.5.3.6 get_details() [2/2]	30
7.5.3.7 operator=() [1/2]	30
7.5.3.8 operator=() [2/2]	30
7.5.4 Friends And Related Function Documentation	31
7.5.4.1 operator<< [1/2]	31
7.5.4.2 operator<< [2/2]	31
7.5.5 Member Data Documentation	32
7.5.5.1 alg_name_	32
7.5.5.2 details_	32
7.5.5.3 kem_	32
7.5.5.4 secret_key_	32
7.6 oqs::MechanismNotEnabledError Class Reference	33
7.6.1 Detailed Description	33
7.6.2 Constructor & Destructor Documentation	34
7.6.2.1 MechanismNotEnabledError()	34
7.7 oqs::MechanismNotSupportedError Class Reference	34
7.7.1 Detailed Description	35
7.7.2 Constructor & Destructor Documentation	35
7.7.2.1 MechanismNotSupportedError()	35
7.8 oqs::Signature Class Reference	36
7.8.1 Detailed Description	37
7.8.2 Constructor & Destructor Documentation	37
7.8.2.1 Signature() [1/3]	37
7.8.2.2 Signature() [2/3]	37
7.8.2.3 Signature() [3/3]	38
7.8.2.4 ~Signature()	38
7.8.3 Member Function Documentation	38
7.8.3.1 export_secret_key()	38
7.8.3.2 generate_keypair()	38
7.8.3.3 get_details() [1/2]	39
7.8.3.4 get_details() [2/2]	39
7.8.3.5 operator=() [1/2]	39
7.8.3.6 operator=() [2/2]	39
7.8.3.7 sign()	40
7.8.3.8 verify()	40
7.8.4 Friends And Related Function Documentation	40

7.8.4.1 operator<< [1/2]	41
7.8.4.2 operator<< [2/2]	41
7.8.5 Member Data Documentation	41
7.8.5.1 alg_name_	41
7.8.5.2 details_	42
7.8.5.3 secret_key_	42
7.8.5.4 sig_	42
7.9 oqs::Sigs Class Reference	42
7.9.1 Detailed Description	43
7.9.2 Constructor & Destructor Documentation	44
7.9.2.1 Sigs()	44
7.9.3 Member Function Documentation	44
7.9.3.1 get_enabled_sigs()	44
7.9.3.2 get_sig_name()	44
7.9.3.3 get_supported_sigs()	45
7.9.3.4 is_sig_enabled()	45
7.9.3.5 is_sig_supported()	45
7.9.3.6 max_number_sigs()	46
7.9.4 Friends And Related Function Documentation	46
7.9.4.1 internal::Singleton< const Sigs >	46
7.10 oqs::internal::Singleton< T > Class Template Reference	46
7.10.1 Detailed Description	47
7.10.2 Constructor & Destructor Documentation	47
7.10.2.1 Singleton() [1/2]	47
7.10.2.2 Singleton() [2/2]	47
7.10.2.3 ~Singleton()	47
7.10.3 Member Function Documentation	48
7.10.3.1 get_instance()	48
7.10.3.2 operator=()	48
7.11 oqs::Timer< T, CLOCK_T > Class Template Reference	48
7.11.1 Detailed Description	49
7.11.2 Constructor & Destructor Documentation	49
7.11.2.1 Timer()	49
7.11.2.2 ~Timer()	49
7.11.3 Member Function Documentation	50
7.11.3.1 get_duration()	50
7.11.3.2 tic()	50
7.11.3.3 tics()	50
7.11.3.4 toc()	51
7.11.4 Friends And Related Function Documentation	51
7.11.4.1 operator<<	51
7.11.5 Member Data Documentation	51

7.11.5.1 end_	51
7.11.5.2 start_	51
8 File Documentation	53
8.1 oqs_cpp.h File Reference	53
8.1.1 Detailed Description	54
8.1.2 Function Documentation	55
8.1.2.1 operator<<() [1/2]	55
8.1.2.2 operator<<() [2/2]	55
8.2 /Users/vlad/liboqs-cpp/README.md File Reference	55
Index	57

Chapter 1

liboqs-cpp: C++ bindings for liboqs

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

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

Pre-requisites

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

Contents

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

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

Usage

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

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

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

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

The examples in the `examples` folder are self-explanatory and provide more details about the wrapper's API.

Building on POSIX (Linux/UNIX-like) platforms

First, you must build the master branch of liboqs according to the [liboqs building instructions](#), followed (optionally) by a `sudo make install` to ensure that the compiled library is system-wide visible (by default it installs under `/usr/local/include` and `/usr/local/lib` under Linux/macOS).

Next, to use the wrapper, you simply `#include "oqs_cpp.h"` in your program. The wrapper contains a CMake build system for both examples and unit tests. To compile and run the examples, create a `build` folder inside the root folder of the project, change directory to `build`, then type

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

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

```
make kem
```

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

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

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

Building on Windows

We provide CMake support for Visual Studio. We recommend using Visual Studio 2017 or later (preferably Visual Studio 2019). For comprehensive details about using CMake with Visual Studio please read [this page](#).

In addition, a Visual Studio 2017 solution containing both key encapsulation and signature examples from [examples](#) as two separate projects is provided in the `VisualStudio` folder. Building instructions:

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

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

Limitations and security

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

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

We acknowledge that some parties may want to begin deploying post-quantum cryptography prior to the conclusion of the NIST standardization project. We strongly recommend that any attempts to do make use of so-called **hybrid cryptography**, in which post-quantum public-key algorithms are used alongside traditional public key algorithms (like RSA or elliptic curves) so that the solution is at least no less secure than existing traditional cryptography.

Just like liboqs, liboqs-cpp is provided "as is", without warranty of any kind. See [LICENSE](#) for the full disclaimer.

License

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

Team

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

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

Support

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

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

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

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

internal	Internal implementation details	13
oqs	Main namespace for the liboqs C++ wrapper	13
oqs::C	Namespace containing all of the oqs C functions, so they do not pollute the oqs namespace . .	15
oqs::internal	15
oqs_literals	15

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

oqs::KeyEncapsulation::alg_details_	17
oqs::Signature::alg_details_	19
oqs::internal::HexChop	20
oqs::KeyEncapsulation	26
runtime_error	
oqs::MechanismNotEnabledError	33
oqs::MechanismNotSupportedError	34
oqs::Signature	36
oqs::internal::Singleton< T >	46
oqs::KEMs	23
oqs::internal::Singleton< const KEMs >	46
oqs::internal::Singleton< const Sigs >	46
oqs::Sigs	42
oqs::Timer< T, CLOCK_T >	48

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	17
oqs::Signature::alg_details_	
Signature algorithm details	19
oqs::internal::HexChop	
Std::ostream manipulator for long vectors of oqs::byte , use it to display only a small number of elements from the beginning and end of the vector	20
oqs::KEMs	
Singleton class, contains details about supported/enabled key exchange mechanisms (KEMs)	23
oqs::KeyEncapsulation	
Key encapsulation mechanisms	26
oqs::MechanismNotEnabledError	
Cryptographic scheme not enabled	33
oqs::MechanismNotSupportedError	
Cryptographic scheme not supported	34
oqs::Signature	
Signature mechanisms	36
oqs::Sigs	
Singleton class, contains details about supported/enabled signature mechanisms	42
oqs::internal::Singleton< T >	
Singleton class using CRTP pattern	46
oqs::Timer< T, CLOCK_T >	
High resolution timer	48

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	53
---------------------------	---	--------------------

Chapter 6

Namespace Documentation

6.1 internal Namespace Reference

Internal implementation details.

6.1.1 Detailed Description

Internal implementation details.

6.2 oqs Namespace Reference

Main namespace for the liboqs C++ wrapper.

Namespaces

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

Classes

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

Typedefs

- using `byte` = `std::uint8_t`
byte (unsigned)
- using `bytes` = `std::vector< byte >`
vector of bytes (unsigned)
- using `OQS_STATUS` = `C::OQS_STATUS`
bring OQS_STATUS into the oqs namespace

Functions

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

6.2.1 Detailed Description

Main namespace for the liboqs C++ wrapper.

6.2.2 Typedef Documentation

6.2.2.1 byte

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

6.2.2.2 bytes

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

6.2.2.3 OQS_STATUS

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

6.2.3 Function Documentation

6.2.3.1 hex_chop()

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

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

Parameters

<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

Returns

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

6.3 oqs::C Namespace Reference

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

6.3.1 Detailed Description

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

6.4 oqs::internal Namespace Reference

Classes

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

6.5 oqs_literals Namespace Reference

Functions

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

6.5.1 Function Documentation

6.5.1.1 operator""_bytes()

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

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

Note

The null terminator is not included

Parameters

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

Returns

The byte representation of the input C-style string

Chapter 7

Class Documentation

7.1 oqs::KeyEncapsulation::alg_details_ Struct Reference

KEM algorithm details.

Public Attributes

- std::string [name](#)
- std::string [version](#)
- std::size_t [claimed_nist_level](#)
- bool [is_ind_cca](#)
- std::size_t [length_public_key](#)
- std::size_t [length_secret_key](#)
- std::size_t [length_ciphertext](#)
- std::size_t [length_shared_secret](#)

7.1.1 Detailed Description

KEM algorithm details.

7.1.2 Member Data Documentation

7.1.2.1 claimed_nist_level

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

7.1.2.2 is_ind_cca

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

7.1.2.3 length_ciphertext

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

7.1.2.4 length_public_key

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

7.1.2.5 length_secret_key

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

7.1.2.6 length_shared_secret

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

7.1.2.7 name

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

7.1.2.8 version

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

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

- [oqs_cpp.h](#)

7.2 oqs::Signature::alg_details_ Struct Reference

[Signature](#) algorithm details.

Public Attributes

- `std::string` [name](#)
- `std::string` [version](#)
- `std::size_t` [claimed_nist_level](#)
- `bool` [is_euf_cma](#)
- `std::size_t` [length_public_key](#)
- `std::size_t` [length_secret_key](#)
- `std::size_t` [max_length_signature](#)

7.2.1 Detailed Description

[Signature](#) algorithm details.

7.2.2 Member Data Documentation

7.2.2.1 `claimed_nist_level`

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

7.2.2.2 `is_euf_cma`

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

7.2.2.3 `length_public_key`

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

7.2.2.4 `length_secret_key`

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

7.2.2.5 max_length_signature

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

7.2.2.6 name

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

7.2.2.7 version

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

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

- [oqs_cpp.h](#)

7.3 oqs::internal::HexChop Class Reference

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

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

Public Member Functions

- [HexChop](#) (const [oqs::bytes](#) &v, std::size_t start, std::size_t end)
Constructs an instance of [oqs::internal::HexChop](#).

Private Member Functions

- void [manipulate_ostream_](#) (std::ostream &os, std::size_t start, std::size_t end, bool is_short) const
std::ostream manipulator

Private Attributes

- [bytes v_](#)
vector of bytes
- std::size_t [start_](#)
- std::size_t [end_](#)
number of hex bytes taken from the start and from the end

Friends

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

7.3.1 Detailed Description

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

7.3.2 Constructor & Destructor Documentation

7.3.2.1 HexChop()

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

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

Parameters

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

7.3.3 Member Function Documentation

7.3.3.1 manipulate_ostream_()

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

`std::ostream` manipulator

Parameters

<i>os</i>	Output stream
<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 HexChop & rhs ) [friend]
```

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

Parameters

<i>os</i>	Output stream
<i>rhs</i>	oqs::internal::HexChop instance

Returns

Reference to the output stream

7.3.5 Member Data Documentation

7.3.5.1 end_

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

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

7.3.5.2 start_

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

7.3.5.3 v_

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

vector of bytes

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

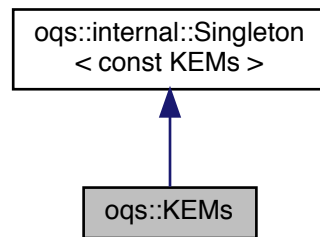
- [oqs_cpp.h](#)

7.4 oqs::KEMs Class Reference

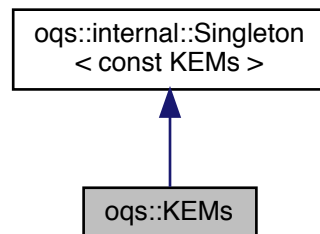
Singleton class, contains details about supported/enabled key exchange mechanisms ([KEMs](#))

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

Inheritance diagram for oqs::KEMs:



Collaboration diagram for oqs::KEMs:



Static Public Member Functions

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

Private Member Functions

- [KEMs](#) ()=default
Private default constructor.

Friends

- class [internal::Singleton](#)< const [KEMs](#) >

Additional Inherited Members

7.4.1 Detailed Description

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

7.4.2 Constructor & Destructor Documentation

7.4.2.1 KEMs()

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

Private default constructor.

Note

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

7.4.3 Member Function Documentation

7.4.3.1 get_enabled_KEMs()

```
static const std::vector<std::string>& oqs::KEMs::get_enabled_KEMs ( ) [inline], [static]
```

Vector of enabled KEM algorithms.

Returns

Vector of enabled KEM algorithms

7.4.3.2 get_KEM_name()

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

KEM algorithm name.

Parameters

<i>alg</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 internal::Singleton< const KEMs >

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

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

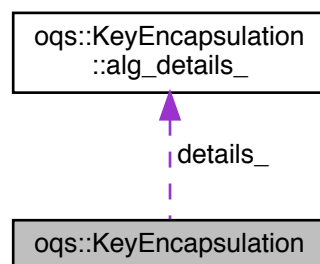
- [oqs_cpp.h](#)

7.5 oqs::KeyEncapsulation Class Reference

Key encapsulation mechanisms.

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

Collaboration diagram for oqs::KeyEncapsulation:



Classes

- struct [alg_details_](#)
KEM algorithm details.

Public Member Functions

- [KeyEncapsulation](#) (const std::string &alg_name, const [bytes](#) &secret_key={})
Constructs an instance of [oqs::KeyEncapsulation](#).
- [KeyEncapsulation](#) (const [KeyEncapsulation](#) &)=default
Default copy constructor.
- [KeyEncapsulation](#) & operator= (const [KeyEncapsulation](#) &)=default
Default copy assignment operator.
- [KeyEncapsulation](#) ([KeyEncapsulation](#) &&rhs)
Move constructor, guarantees that the rvalue secret key is always zeroed.
- [KeyEncapsulation](#) & operator= ([KeyEncapsulation](#) &&rhs)
Move assignment operator, guarantees that the rvalue secret key is always zeroed.
- virtual [~KeyEncapsulation](#) ()
Virtual default destructor.
- const [alg_details_](#) & [get_details](#) () const &
KEM algorithm details, lvalue overload.
- [alg_details_](#) [get_details](#) () const &&
KEM algorithm details, rvalue overload.
- [bytes](#) [generate_keypair](#) ()
Generate public key/secret key pair.
- [bytes](#) [export_secret_key](#) () const
Export secret key.
- std::pair< [bytes](#), [bytes](#) > [encap_secret](#) (const [bytes](#) &public_key) const
Encapsulate secret.
- [bytes](#) [decap_secret](#) (const [bytes](#) &ciphertext) const
Decapsulate secret.

Private Attributes

- std::string [alg_name_](#)
cryptographic algorithm name
- std::shared_ptr< C::OQS_KEM > [kem_](#)
liboqs smart pointer to C::OQS_KEM
- [bytes](#) [secret_key_](#) {}
secret key
- struct [oqs::KeyEncapsulation::alg_details_](#) [details_](#)

Friends

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

7.5.1 Detailed Description

Key encapsulation mechanisms.

7.5.2 Constructor & Destructor Documentation

7.5.2.1 KeyEncapsulation() [1/3]

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

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

Parameters

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

7.5.2.2 KeyEncapsulation() [2/3]

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

Default copy constructor.

7.5.2.3 KeyEncapsulation() [3/3]

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

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

Parameters

<i>rhs</i>	oqs::KeyEncapsulation instance
------------	--

7.5.2.4 ~KeyEncapsulation()

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

Virtual default destructor.

7.5.3 Member Function Documentation

7.5.3.1 decap_secret()

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

Decapsulate secret.

Parameters

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

Returns

Shared secret

7.5.3.2 encap_secret()

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

Encapsulate secret.

Parameters

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

Returns

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

7.5.3.3 export_secret_key()

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

Export secret key.

Returns

Secret key

7.5.3.4 generate_keypair()

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

Generate public key/secret key pair.

Returns

Public key

7.5.3.5 get_details() [1/2]

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

KEM algorithm details, lvalue overload.

Returns

KEM algorithm details

7.5.3.6 get_details() [2/2]

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

KEM algorithm details, rvalue overload.

Returns

KEM algorithm details

7.5.3.7 operator=() [1/2]

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

Default copy assignment operator.

Returns

Reference to the current instance

7.5.3.8 operator=() [2/2]

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

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

Parameters

<i>rhs</i>	oqs::KeyEncapsulation instance
------------	--

Returns

Reference to the current instance

7.5.4 Friends And Related Function Documentation

7.5.4.1 `operator<<` [1/2]

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

std::ostream extraction operator for the KEM algorithm details

Parameters

<i>os</i>	Output stream
<i>rhs</i>	Algorithm details 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>	oqs::KeyEncapsulation instance

Returns

Reference to the output stream

7.5.5 Member Data Documentation

7.5.5.1 alg_name_

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

cryptographic algorithm name

7.5.5.2 details_

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

7.5.5.3 kem_

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

Initial value:

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

liboqs smart pointer to C::OQS_KEM

7.5.5.4 secret_key_

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

secret key

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

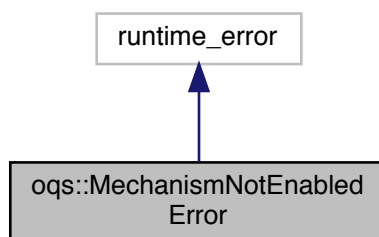
- [oqs_cpp.h](#)

7.6 oqs::MechanismNotEnabledError Class Reference

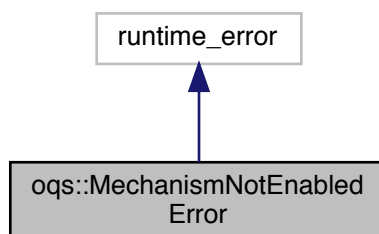
Cryptographic scheme not enabled.

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

Inheritance diagram for oqs::MechanismNotEnabledError:



Collaboration diagram for oqs::MechanismNotEnabledError:



Public Member Functions

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

7.6.1 Detailed Description

Cryptographic scheme not enabled.

7.6.2 Constructor & Destructor Documentation

7.6.2.1 MechanismNotEnabledError()

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

Constructor.

Parameters

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

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

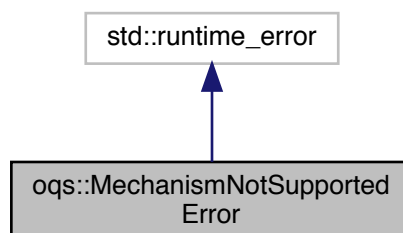
- [oqs_cpp.h](#)

7.7 oqs::MechanismNotSupportedError Class Reference

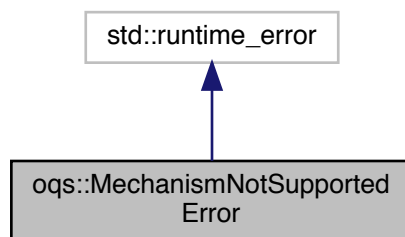
Cryptographic scheme not supported.

```
#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

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

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

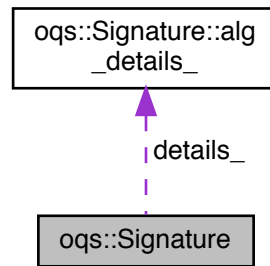
- [oqs_cpp.h](#)

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](#).
- [Signature](#) (const [Signature](#) &)=default
Default copy constructor.
- [Signature](#) & [operator=](#) (const [Signature](#) &)=default
Default copy assignment operator.
- [Signature](#) ([Signature](#) &&rhs)
Move constructor, guarantees that the rvalue secret key is always zeroed.
- [Signature](#) & [operator=](#) ([Signature](#) &&rhs)
Move assignment operator, guarantees that the rvalue secret key is always zeroed.
- virtual [~Signature](#) ()
Virtual default destructor.
- const [alg_details_](#) & [get_details](#) () const &
Signature algorithm details, lvalue overload.
- [alg_details_](#) [get_details](#) () const &&
Signature algorithm details, rvalue overload.
- [bytes](#) [generate_keypair](#) ()
Generate public key/secret key pair.
- [bytes](#) [export_secret_key](#) () const
Export secret key.
- [bytes](#) [sign](#) (const [bytes](#) &message) const
Sign message.
- bool [verify](#) (const [bytes](#) &message, const [bytes](#) &signature, const [bytes](#) &public_key) const
Verify signature.

Private Attributes

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

Friends

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

7.8.1 Detailed Description

[Signature](#) mechanisms.

7.8.2 Constructor & Destructor Documentation

7.8.2.1 Signature() [1/3]

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

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

Parameters

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

7.8.2.2 Signature() [2/3]

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

Default copy constructor.

7.8.2.3 Signature() [3/3]

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

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

Parameters

<i>rhs</i>	oqs::Signature instance
------------	---

7.8.2.4 ~Signature()

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

Virtual default destructor.

7.8.3 Member Function Documentation

7.8.3.1 export_secret_key()

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

Export secret key.

Returns

Secret key

7.8.3.2 generate_keypair()

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

Generate public key/secret key pair.

Returns

Public key

7.8.3.3 `get_details()` [1/2]

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

[Signature](#) algorithm details, lvalue overload.

Returns

[Signature](#) algorithm details

7.8.3.4 `get_details()` [2/2]

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

[Signature](#) algorithm details, rvalue overload.

Returns

[Signature](#) algorithm details

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

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

Default copy assignment operator.

Returns

Reference to the current instance

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

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

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

Parameters

<i>rhs</i>	oqs::Signature instance
------------	---

Returns

Reference to the current instance

7.8.3.7 sign()

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

Sign message.

Parameters

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

Returns

Message signature

7.8.3.8 verify()

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

Verify signature.

Parameters

<i>message</i>	Message
<i>signature</i>	Signature
<i>public_key</i>	Public key

Returns

True if the signature is valid, false otherwise

7.8.4 Friends And Related Function Documentation

7.8.4.1 operator<< [1/2]

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

std::ostream extraction operator for the signature algorithm details

Parameters

<i>os</i>	Output stream
<i>rhs</i>	Algorithm details instance

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>	oqs::Signature instance

Returns

Reference to the output stream

7.8.5 Member Data Documentation

7.8.5.1 alg_name_

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

cryptographic algorithm name

7.8.5.2 details_

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

7.8.5.3 secret_key_

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

secret key

7.8.5.4 sig_

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

Initial value:

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

liboqs smart pointer to C::OQS_SIG

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

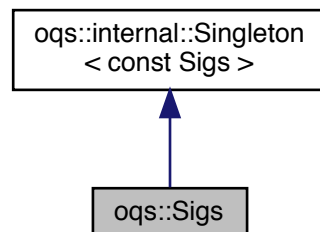
- [oqs_cpp.h](#)

7.9 oqs::Sigs Class Reference

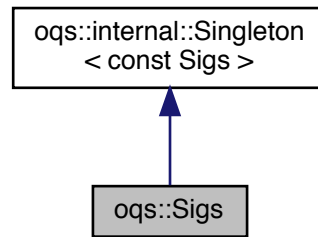
Singleton class, contains details about supported/enabled signature mechanisms.

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

Inheritance diagram for oqs::Sigs:



Collaboration diagram for oqs::Sigs:



Static Public Member Functions

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

Private Member Functions

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

Friends

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

Additional Inherited Members

7.9.1 Detailed Description

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

7.9.2 Constructor & Destructor Documentation

7.9.2.1 Sigs()

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

Private default constructor.

Note

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

7.9.3 Member Function Documentation

7.9.3.1 get_enabled_sigs()

```
static const std::vector<std::string>& oqs::Sigs::get_enabled_sigs ( ) [inline], [static]
```

Vector of enabled signature algorithms.

Returns

Vector of enabled signature algorithms

7.9.3.2 get_sig_name()

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

[Signature](#) algorithm name.

Parameters

<i>alg_{id}</i>	Cryptographic 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 internal::Singleton< const Sigs >

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

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

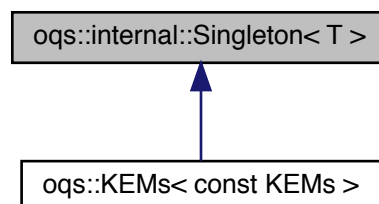
- [oqs_cpp.h](#)

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

[Singleton](#) class using CRTP pattern.

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

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



Static Public Member Functions

- static T & [get_instance](#) () noexcept(std::is_nothrow_constructible< T >::value)
[Singleton](#) instance (thread-safe) via CRTP pattern.

Protected Member Functions

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

7.10.1 Detailed Description

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

[Singleton](#) class using CRTP pattern.

Note

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

Template Parameters

<i>T</i>	Class type of which instance will become a Singleton
----------	--

7.10.2 Constructor & Destructor Documentation

7.10.2.1 Singleton() [1/2]

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

7.10.2.2 Singleton() [2/2]

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

7.10.2.3 ~Singleton()

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

7.10.3 Member Function Documentation

7.10.3.1 `get_instance()`

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

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

Returns

[Singleton](#) instance

7.10.3.2 `operator=()`

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

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

- [oqs_cpp.h](#)

7.11 `oqs::Timer< T, CLOCK_T >` Class Template Reference

High resolution timer.

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

Public Member Functions

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

Protected Attributes

- CLOCK_T::time_point [start_](#)
- CLOCK_T::time_point [end_](#)

Friends

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

7.11.1 Detailed Description

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

High resolution timer.

Note

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

Template Parameters

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

7.11.2 Constructor & Destructor Documentation

7.11.2.1 Timer()

```
template<typename T = std::chrono::duration<double>, typename CLOCK_T = std::chrono::steady_clock>
oqs::Timer< T, CLOCK_T >::Timer ( ) [inline], [noexcept]
```

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

7.11.2.2 ~Timer()

```
template<typename T = std::chrono::duration<double>, typename CLOCK_T = std::chrono::steady_clock>
virtual oqs::Timer< T, CLOCK_T >::~~Timer ( ) [virtual], [default]
```

Default virtual destructor.

7.11.3 Member Function Documentation

7.11.3.1 `get_duration()`

```
template<typename T = std::chrono::duration<double>, typename CLOCK_T = std::chrono::steady↵_clock>
template<typename U = T>
U oqs::Timer< T, CLOCK_T >::get_duration ( ) const [inline], [noexcept]
```

Duration specified by U.

Template Parameters

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

Returns

Duration that passed between the instantiation/reset and invocation of `oqs::Timer::toc()`

7.11.3.2 `tic()`

```
template<typename T = std::chrono::duration<double>, typename CLOCK_T = std::chrono::steady↵_clock>
void oqs::Timer< T, CLOCK_T >::tic ( ) [inline], [noexcept]
```

Resets the chronometer.

Resets the start/end point to the current time

7.11.3.3 `tics()`

```
template<typename T = std::chrono::duration<double>, typename CLOCK_T = std::chrono::steady↵_clock>
double oqs::Timer< T, CLOCK_T >::tics ( ) const [inline], [noexcept]
```

Time passed in the duration specified by T.

Returns

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

7.11.3.4 toc()

```
template<typename T = std::chrono::duration<double>, typename CLOCK_T = std::chrono::steady↵
_clock>
const Timer& oqs::Timer< T, CLOCK_T >::toc ( ) & [inline], [noexcept]
```

Stops the chronometer.

Set the current time as the end point

Returns

Reference to the current instance

7.11.4 Friends And Related Function Documentation

7.11.4.1 operator<<

```
template<typename T = std::chrono::duration<double>, typename CLOCK_T = std::chrono::steady↵
_clock>
std::ostream& operator<< (
    std::ostream & os,
    const Timer< T, CLOCK_T > & rhs ) [friend]
```

7.11.5 Member Data Documentation

7.11.5.1 end_

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

7.11.5.2 start_

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

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

- [oqs_cpp.h](#)

Chapter 8

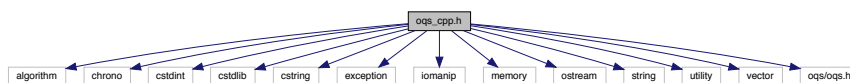
File Documentation

8.1 oqs_cpp.h File Reference

Main header file for the liboqs C++ wrapper.

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

Include dependency graph for oqs_cpp.h:



Classes

- class `oqs::internal::Singleton< T >`
Singleton class using CRTP pattern.
- class `oqs::internal::HexChop`
std::ostream manipulator for long vectors of `oqs::byte`, use it to display only a small number of elements from the beginning and end of the vector
- class `oqs::Timer< T, CLOCK_T >`
High resolution timer.
- class `oqs::MechanismNotSupportedError`
Cryptographic scheme not supported.

- class [oqs::MechanismNotEnabledError](#)
Cryptographic scheme not enabled.
- class [oqs::KEMs](#)
Singleton class, contains details about supported/enabled key exchange mechanisms ([KEMs](#))
- class [oqs::KeyEncapsulation](#)
Key encapsulation mechanisms.
- struct [oqs::KeyEncapsulation::alg_details_](#)
KEM algorithm details.
- class [oqs::Sigs](#)
Singleton class, contains details about supported/enabled signature mechanisms.
- class [oqs::Signature](#)
Signature mechanisms.
- struct [oqs::Signature::alg_details_](#)
Signature algorithm details.

Namespaces

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

Typedefs

- using [oqs::byte](#) = std::uint8_t
byte (unsigned)
- using [oqs::bytes](#) = std::vector< byte >
vector of bytes (unsigned)
- using [oqs::OQS_STATUS](#) = C::OQS_STATUS
bring OQS_STATUS into the oqs namespace

Functions

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

8.1.1 Detailed Description

Main header file for the liboqs C++ wrapper.

8.1.2 Function Documentation

8.1.2.1 `operator<<()` [1/2]

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

std::ostream extraction operator for `oqs::bytes`

Parameters

<i>os</i>	Output stream
<i>rhs</i>	Vector of <code>oqs::byte</code>

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 <code>std::string</code>

Returns

Reference to the output stream

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

Index

/Users/vlad/liboqs-cpp/README.md, [55](#)

~KeyEncapsulation

oqs::KeyEncapsulation, [28](#)

~Signature

oqs::Signature, [38](#)

~Singleton

oqs::internal::Singleton< T >, [47](#)

~Timer

oqs::Timer< T, CLOCK_T >, [49](#)

alg_name_

oqs::KeyEncapsulation, [32](#)

oqs::Signature, [41](#)

byte

oqs, [14](#)

bytes

oqs, [14](#)

claimed_nist_level

oqs::KeyEncapsulation::alg_details_, [17](#)

oqs::Signature::alg_details_, [19](#)

decap_secret

oqs::KeyEncapsulation, [29](#)

details_

oqs::KeyEncapsulation, [32](#)

oqs::Signature, [41](#)

encap_secret

oqs::KeyEncapsulation, [29](#)

end_

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

oqs::Timer< T, CLOCK_T >, [51](#)

export_secret_key

oqs::KeyEncapsulation, [29](#)

oqs::Signature, [38](#)

generate_keypair

oqs::KeyEncapsulation, [30](#)

oqs::Signature, [38](#)

get_details

oqs::KeyEncapsulation, [30](#)

oqs::Signature, [38](#), [39](#)

get_duration

oqs::Timer< T, CLOCK_T >, [50](#)

get_enabled_KEMs

oqs::KEMs, [24](#)

get_enabled_sigs

oqs::Sigs, [44](#)

get_instance

oqs::internal::Singleton< T >, [48](#)

get_KEM_name

oqs::KEMs, [24](#)

get_sig_name

oqs::Sigs, [44](#)

get_supported_KEMs

oqs::KEMs, [25](#)

get_supported_sigs

oqs::Sigs, [44](#)

hex_chop

oqs, [14](#)

HexChop

oqs::internal::HexChop, [21](#)

internal, [13](#)

internal::Singleton< const KEMs >

oqs::KEMs, [26](#)

internal::Singleton< const Sigs >

oqs::Sigs, [46](#)

is_euf_cma

oqs::Signature::alg_details_, [19](#)

is_ind_cca

oqs::KeyEncapsulation::alg_details_, [17](#)

is_KEM_enabled

oqs::KEMs, [25](#)

is_KEM_supported

oqs::KEMs, [25](#)

is_sig_enabled

oqs::Sigs, [45](#)

is_sig_supported

oqs::Sigs, [45](#)

kem_

oqs::KeyEncapsulation, [32](#)

KEMs

oqs::KEMs, [24](#)

KeyEncapsulation

oqs::KeyEncapsulation, [28](#)

length_ciphertext

oqs::KeyEncapsulation::alg_details_, [18](#)

length_public_key

oqs::KeyEncapsulation::alg_details_, [18](#)

oqs::Signature::alg_details_, [19](#)

length_secret_key

oqs::KeyEncapsulation::alg_details_, [18](#)

oqs::Signature::alg_details_, [19](#)

length_shared_secret

oqs::KeyEncapsulation::alg_details_, [18](#)

- manipulate_ostream_
 - oqs::internal::HexChop, 21
- max_length_signature
 - oqs::Signature::alg_details_, 19
- max_number_KEMs
 - oqs::KEMs, 26
- max_number_sigs
 - oqs::Sigs, 45
- MechanismNotEnabledError
 - oqs::MechanismNotEnabledError, 34
- MechanismNotSupportedError
 - oqs::MechanismNotSupportedError, 35
- name
 - oqs::KeyEncapsulation::alg_details_, 18
 - oqs::Signature::alg_details_, 20
- operator<<
 - oqs::internal::HexChop, 22
 - oqs::KeyEncapsulation, 31
 - oqs::Signature, 40, 41
 - oqs::Timer< T, CLOCK_T >, 51
 - oqs_cpp.h, 55
- operator=
 - oqs::internal::Singleton< T >, 48
 - oqs::KeyEncapsulation, 30
 - oqs::Signature, 39
- operator""_bytes
 - oqs_literals, 15
- oqs, 13
 - byte, 14
 - bytes, 14
 - hex_chop, 14
 - OQS_STATUS, 14
- oqs::C, 15
- oqs::internal, 15
- oqs::internal::HexChop, 20
 - end_, 22
 - HexChop, 21
 - manipulate_ostream_, 21
 - operator<<, 22
 - start_, 22
 - v_, 22
- oqs::internal::Singleton< T >, 46
 - ~Singleton, 47
 - get_instance, 48
 - operator=, 48
 - Singleton, 47
- oqs::KEMs, 23
 - get_enabled_KEMs, 24
 - get_KEM_name, 24
 - get_supported_KEMs, 25
 - internal::Singleton< const KEMs >, 26
 - is_KEM_enabled, 25
 - is_KEM_supported, 25
 - KEMs, 24
 - max_number_KEMs, 26
- oqs::KeyEncapsulation, 26
 - ~KeyEncapsulation, 28
 - alg_name_, 32
 - decap_secret, 29
 - details_, 32
 - encap_secret, 29
 - export_secret_key, 29
 - generate_keypair, 30
 - get_details, 30
 - kem_, 32
 - KeyEncapsulation, 28
 - operator<<, 31
 - operator=, 30
 - secret_key_, 32
- oqs::KeyEncapsulation::alg_details_, 17
 - claimed_nist_level, 17
 - is_ind_cca, 17
 - length_ciphertext, 18
 - length_public_key, 18
 - length_secret_key, 18
 - length_shared_secret, 18
 - name, 18
 - version, 18
- oqs::MechanismNotEnabledError, 33
 - MechanismNotEnabledError, 34
- oqs::MechanismNotSupportedError, 34
 - MechanismNotSupportedError, 35
- oqs::Signature, 36
 - ~Signature, 38
 - alg_name_, 41
 - details_, 41
 - export_secret_key, 38
 - generate_keypair, 38
 - get_details, 38, 39
 - operator<<, 40, 41
 - operator=, 39
 - secret_key_, 42
 - sig_, 42
 - sign, 40
 - Signature, 37
 - verify, 40
- oqs::Signature::alg_details_, 19
 - claimed_nist_level, 19
 - is_euf_cma, 19
 - length_public_key, 19
 - length_secret_key, 19
 - max_length_signature, 19
 - name, 20
 - version, 20
- oqs::Sigs, 42
 - get_enabled_sigs, 44
 - get_sig_name, 44
 - get_supported_sigs, 44
 - internal::Singleton< const Sigs >, 46
 - is_sig_enabled, 45
 - is_sig_supported, 45
 - max_number_sigs, 45
 - Sigs, 44
- oqs::Timer< T, CLOCK_T >, 48
 - ~Timer, 49

- end_, [51](#)
- get_duration, [50](#)
- operator<<, [51](#)
- start_, [51](#)
- tic, [50](#)
- tics, [50](#)
- Timer, [49](#)
- toc, [50](#)
- oqs_cpp.h, [53](#)
 - operator<<, [55](#)
- oqs_literals, [15](#)
 - operator""_bytes, [15](#)
- OQS_STATUS
 - oqs, [14](#)
- secret_key_
 - oqs::KeyEncapsulation, [32](#)
 - oqs::Signature, [42](#)
- sig_
 - oqs::Signature, [42](#)
- sign
 - oqs::Signature, [40](#)
- Signature
 - oqs::Signature, [37](#)
- Sigs
 - oqs::Sigs, [44](#)
- Singleton
 - oqs::internal::Singleton< T >, [47](#)
- start_
 - oqs::internal::HexChop, [22](#)
 - oqs::Timer< T, CLOCK_T >, [51](#)
- tic
 - oqs::Timer< T, CLOCK_T >, [50](#)
- tics
 - oqs::Timer< T, CLOCK_T >, [50](#)
- Timer
 - oqs::Timer< T, CLOCK_T >, [49](#)
- toc
 - oqs::Timer< T, CLOCK_T >, [50](#)
- v_
 - oqs::internal::HexChop, [22](#)
- verify
 - oqs::Signature, [40](#)
- version
 - oqs::KeyEncapsulation::alg_details_, [18](#)
 - oqs::Signature::alg_details_, [20](#)