CryptoGateway Documentation

Adrian Bedard Jonathan Bedard

March 10, 2016

Contents

Part I

CryptoGateway Library

Chapter 1

Introduction

The CryptoGateway library contains classes which handle cryptography. CryptoGateway is designed as an open source library, so much of the cryptography within the library is relatively simple. CryptoGateway is not meant to define cryptography to be used widely, rather, it is meant to provide a series of generalized hooks and interfaces which can be extended to various cryptographic algorithms.

1.1 Namespace

CryptoGateway uses the crypto namespace. The crypto namespace is designed for class, functions and constants related to cryptography. CrytpoGateway depends on many of the tools defined in the os namespace. Additionally, the crypto namespace contains a series of nested namespaces which help to disambiguate constants.

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Namespace Documentation

4.1 crypto Namespace Reference

Classes

- class **actionOnFileClosed**

  File closed error.
- class **actionOnFileError**

  File error.
- class **avlKeyBank**

  AVL key back.
- class **binaryDecryptor**

  Encrypted binary file output.
- class **binaryEncryptor**

  Encrypted binary file output.
- class **bufferLargeError**

  Buffer too large.
- class **bufferSmallError**

  Buffer too small.
- class **checksum_message**
- class **customError**

  *Custom **crypto::error*** (p. **??**).
- class **error**

  Sortable exception.
- class **errorListener**

  crypto::error (p. **??**) *listener*
- class **errorSender**

  *Sends **crypto::error*** (p. **??**).
- class **fileFormatError**

  File format error.
- class **fileOpenError**

File open error.

- class **hash**

  Base hash class.
- class **hashCompareError**

  Hash mis-match.
- class **hashGenerationError**

  Hash generation error.
- class **illegalAlgorithmBind**

  Algorithm bound failure.
- class **insertionFailed**

  ADS Insertion Failed.
- class **integer**

  Integer number definition.
- class **interior_message**
- class **keyBank**

  Key bank interface.
- class **keyMissing**

  Key missing error.
- class **large_integer**
- class **large_number**
- class **masterMismatch**

  Master mis-match.
- class **nodeGroup**

  Node group.
- class **nodeKeyReference**

  Key storage node.
- class **nodeNameReference**

  Name storage node.
- class **NULLDataError**

  NULL data error.
- class **NULLMaster**

  NULL master error.
- class **NULLPublicKey**

  NULL public-key error.
- class **number**

  Basic number definition.
- class **passwordLargeError**

  Symmetric key too big.
- class **passwordSmallError**

  Symmetric key too small.
- class **publicKey**

  Base public-key class.
- class **publicKeyPackage**

- class **publicKeyPackageFrame**
- class **publicKeySizeWrong**

  Public-key size error.
- class **publicKeyTypeBank**
- class **publicRSA**

  RSA public-key encryption.
- class **rc4Hash**

  RC-4 hash class.
- class **RCFour**
- class **RSAKeyGenerator**

  Helper key generation class.
- class **security_gateway**
- class **streamCipher**
- class **streamDecrypter**
- class **streamEncrypter**
- class **streamPackage**
- class **streamPackageFrame**
- class **streamPackageTypeBank**
- class **streamPacket**
- class **unknownErrorType**

  Unknown error.
- class **user**

  Primary user class.
- class **xorHash**

  XOR hash class.

Typedefs

- typedef os::smart_ptr< **error** > **errorPointer**

  *Smart pointer to **crypto::error*** (p. **??**).
- typedef os::smart_ptr< **interior_message** > **smartInteriorMessage**

Functions

- std::ostream & **operator**<< (std::ostream &os, const **hash** &num)

  Output stream operator.
- std::istream & **operator**>> (std::istream &is, **hash** &num)

  Input stream operator.
- template<class hashClass >
  hashClass **hashData** (uint16_t hashType, const unsigned char *data, uint32_t length)

  Hashes data with the specified algorithm.
- std::ostream & **cryptoout_func** ()

  Standard out object for crypto namespace.
- std::ostream & **cryptoerr_func** ()

Standard error object for crypto namespace.

- std::ostream & **operator**<< (std::ostream &os, const **number** &num)

  Output stream operator.
- std::istream & **operator**>> (std::istream &is, **number** &num)

  Input stream operator.
- bool **isHexCharacter** (char c)
- std::string **toHex** (unsigned char i)
- std::string **toHex** (uint32_t i)
- unsigned char **fromHex8** (const std::string &str)
- uint32_t **fromHex32** (const std::string &str)
- static std::vector< std::string > **generateArgumentList** (os::smartXMLNode head)
- static void **recursiveXMLPrinting** (os::smartXMLNode head, os::smart_ptr< **streamCipher** > strm, std::vector< std::string > args, std::ofstream &ofs)
- static os::smartXMLNode **recursiveXMLBuilding** (os::smart_ptr< **streamCipher** > strm, std::vector< std::string > args, std::ifstream &ifs)
- bool **EXML_Output** (std::string path, os::smartXMLNode head, std::string password, os::smart_ptr< **streamPackageFrame** > spf)
- bool **EXML_Output** (std::string path, os::smartXMLNode head, unsigned char *symKey, unsigned int passwordLength, os::smart_ptr< **streamPackageFrame** > spf)
- bool **EXML_Output** (std::string path, os::smartXMLNode head, os::smart_ptr< **publicKey** > pbk, unsigned int lockType, os::smart_ptr< **streamPackageFrame** > spf)
- bool **EXML_Output** (std::string path, os::smartXMLNode head, os::smart_ptr< **number** > **publicKey**, unsigned int pkAlgo, unsigned int pkSize, os::smart_ptr< **streamPackageFrame** > spf)
- os::smartXMLNode **EXML_Input** (std::string path, std::string password)
- os::smartXMLNode **EXML_Input** (std::string path, unsigned char *symKey, unsigned int passwordLength)
- os::smartXMLNode **EXML_Input** (std::string path, os::smart_ptr< **publicKey** > pbk, os::smart_ptr< **keyBank** > kyBank, os::smart_ptr< **nodeGroup** > &author)
- os::smartXMLNode **EXML_Input** (std::string path, os::smart_ptr< **publicKey** > pbk)
- os::smartXMLNode **EXML_Input** (std::string path, os::smart_ptr< **keyBank** > kyBank)
- os::smartXMLNode **EXML_Input** (std::string path, os::smart_ptr< **keyBank** > kyBank, os::smart_ptr< **nodeGroup** > &author)

Variables

- bool **global_logging**

  Deprecated logging flag.
- os::smart_ptr< std::ostream > **cryptoout_ptr**

  Standard out pointer for crypto namespace.
- os::smart_ptr< std::ostream > **cryptoerr_ptr**

  Standard error pointer for crypto namespace.
- const unsigned int **MESSAGE_MAX** =512
- const unsigned int **CHECKSUM_SIZE** =4
- const unsigned int **LARGE_NUMBER_SIZE** =32
- const unsigned int **PRIME_TEST_ITERATION** =10
- static os::smart_ptr< **publicKeyTypeBank** > **_singleton**
- static os::smart_ptr< **streamPackageTypeBank** > **_singleton**

4.1.1 Typedef Documentation

typedef os::smart_ptr<**error**> **crypto::errorPointer**

Smart pointer to **crypto::error** (p. **??**).

typedef os::smart_ptr<**interior_message**> **crypto::smartInteriorMessage**

4.1.2 Function Documentation

std::ostream& crypto::cryptoerr_func ( )

Standard error object for crypto namespace.
#define statements allow the user to call this function with "crypto::cryptoerr." Logging is achieved by using "crypto::cryptoerr" as one would use "std::cerr."

std::ostream& crypto::cryptoout_func ( )

Standard out object for crypto namespace.
#define statements allow the user to call this function with "crypto::cryptoout." Logging is achieved by using "crypto::cryptoout" as one would use "std::cout."

os::smartXMLNode crypto::EXML_Input (std::string path, std::string password)

os::smartXMLNode crypto::EXML_Input (std::string path, unsigned char * symKey, unsigned int passwordLength)

os::smartXMLNode crypto::EXML_Input (std::string path, os::smart_ptr< **publicKey** > pbk, os::smart_ptr< **keyBank** > kyBank, os::smart_ptr< **nodeGroup** > & author)

os::smartXMLNode crypto::EXML_Input (std::string path, os::smart_ptr< **publicKey** > pbk)

os::smartXMLNode crypto::EXML_Input (std::string path, os::smart_ptr< **keyBank** > kyBank)

os::smartXMLNode crypto::EXML_Input (std::string path, os::smart_ptr< **keyBank** > kyBank, os::smart_ptr< **nodeGroup** > & author)

bool crypto::EXML_Output (std::string path, os::smartXMLNode head, std::string password, os::smart_ptr< **streamPackageFrame** > spf)

bool crypto::EXML_Output (std::string path, os::smartXMLNode head, unsigned char * symKey, unsigned int passwordLength, os::smart_ptr< **streamPackageFrame** > spf)

bool crypto::EXML_Output (std::string path, os::smartXMLNode head, os::smart_ptr< **publicKey** > pbk, unsigned int lockType, os::smart_ptr< **streamPackageFrame** > spf)

bool crypto::EXML_Output (std::string path, os::smartXMLNode head, os::smart_ptr< **number** > publicKey, unsigned int pkAlgo, unsigned int pkSize, os::smart_ptr< **streamPackageFrame** > spf)

uint32_t crypto::fromHex32 (const std::string & str)

unsigned char crypto::fromHex8 (const std::string & str)

static std::vector<std::string> crypto::generateArgumentList (os::smartXMLNode head) [static]

template<class hashClass > hashClass crypto::hashData (uint16_t hashType, const unsigned char * data, uint32_t length)

Hashes data with the specified algorithm.

Hashes the provided data array returning a hash of the specified algorithm. This is a template function, which calls the static hash function for the specified algorithm.

Parameters

in	*hashType*	Size of hash
in	*data*	Data array to be hashed
in	*length*	Length of data to be hashed

Returns

Hash for data array

bool crypto::isHexCharacter (char c)

std::ostream& crypto::operator<< (std::ostream & os, const **number** & num)

Output stream operator.

Parameters

	[in/out]	os Output stream
in	*num*	Number to be output

Returns

reference to std::ostream& os

std::ostream& crypto::operator<< (std::ostream & os, const **hash** & num)

Output stream operator.

Outputs a hex version of the hash to the provided output stream. This output will look identical for two hashes which are equal but have different algorithms.

Parameters

	[in/out]	os Output stream
in	*num*	Hash to be printed return Reference to output stream

std::istream& crypto::operator>> (std::istream & is, **number** & num)

Input stream operator.

Parameters

	[in/out]	is Input stream
`in`	*num*	Number to set with the string

Returns

reference to std::istream& is

std::istream& crypto::operator>> (std::istream & is, **hash** & num)

Input stream operator.

Inputs a hex version of the hash from the provided output stream. This function must receive a constructed hash, although it will rebuild the provided hash with the stream data.

Parameters

	[in/out]	is Input stream
`in`	*num*	Hash to be created return Reference to input stream

static os::smartXMLNode crypto::recursiveXMLBuilding (os::smart_ptr< **streamCipher** > strm, std::vector< std::string > args, std::ifstream & ifs) `[static]`

static void crypto::recursiveXMLPrinting (os::smartXMLNode head, os::smart_ptr< **streamCipher** > strm, std::vector< std::string > args, std::ofstream & ofs) `[static]`

std::string crypto::toHex (unsigned char i)

std::string crypto::toHex (uint32_t i)

4.1.3 Variable Documentation

os::smart_ptr<**publicKeyTypeBank**> crypto::_singleton `[static]`

os::smart_ptr<**streamPackageTypeBank**> crypto::_singleton `[static]`

const unsigned int crypto::CHECKSUM_SIZE =4

os::smart_ptr<std::ostream> crypto::cryptoerr_ptr

Standard error pointer for crypto namespace.

This std::ostream is used as standard error for the crypto namespace. This pointer can be swapped out to programmatically redirect standard error for the crypto namespace.

os::smart_ptr<std::ostream> crypto::cryptoout_ptr

Standard out pointer for crypto namespace.

This std::ostream is used as standard out for the crypto namespace. This pointer can be swapped out to programmatically redirect standard out for the crypto namespace.

bool crypto::global_logging

Deprecated logging flag.

Old logging flag. Deprecated in the new CryptoGateway files. This has been replaced by the logging system outlined in this file.

const unsigned int crypto::LARGE_NUMBER_SIZE =32

const unsigned int crypto::MESSAGE_MAX =512

const unsigned int crypto::PRIME_TEST_ITERATION =10

Chapter 5

Class Documentation

Chapter 6

File Documentation