aws-encryption-sdk-python-cli

Release 1.1.7

Jun 05, 2020

Contents

This command line tool can be used to encrypt and decrypt files and directories using the AWS Encryption SDK.

The latest full documentation can be found at Read the Docs.

Find us on GitHub.

CHAPTER 1

Getting Started

1.1 Required Prerequisites

- Python 2.7+ or 3.4+
- aws-encryption-sdk >= 1.3.2

1.2 Installation

Note: If you have not already installed cryptography, you might need to install additional prerequisites as detailed in the cryptography installation guide for your operating system.

```
$ pip install aws-encryption-sdk-cli
```

CHAPTER 2

Usage

2.1 Input and Output

For the most part, the behavior of `aws-encryption-cli` in handling files is based on that of GNU CLIs such as `cp`. A qualifier to this is that when encrypting a file, if a directory is provided as the destination, rather than creating the source filename in the destination directory, a suffix is appended to the destination filename. By default the suffix is `.encrypted` when encrypting and `.decrypted` when decrypting, but a custom suffix can be provided by the caller if desired.

If a destination file already exists, the contents will be overwritten.

Allowed input/output pairings		**output**		
		stdout	**file**	**directory**
input	**stdin**	Y	Y	
	single file	Y	Y	Y
	pattern match			Y
	directory			Y

If the source includes a directory and the `--recursive` flag is set, the entire tree of the source directory is replicated in the target directory.

2.1.1 Parameter Values

Some arguments accept additional parameter values. These values must be provided in the form of `key=value` as demonstrated below.

```
--encryption-context key1=value1 key2=value2 "key 3=value with spaces"
--master-keys provider=aws-kms key=$KEY_ID_1 key=$KEY_ID_2
--caching capacity=3 max_age=80.0
```

2.1.2 Encryption Context

Encrypt

The encryption context is an optional, but recommended, set of key-value pairs that contain arbitrary nonsecret data. The encryption context can contain any data you choose, but it typically consists of data that is useful in logging and tracking, such as data about the file type, purpose, or ownership.

Parameters may be provided using *Parameter Values*.

```
--encryption-context key1=value1 key2=value2 "key 3=value with spaces"
```

Decrypt

If an encryption context is provided on decrypt, it is instead used to require that the message being decrypted was encrypted using an encryption context that matches the specified requirements.

If `key=value` elements are provided, the decryption will only continue if the encryption context found in the encrypted message contains matching pairs.

```
--encryption-context required_key=required_value classification=secret
```

If bare `key` elements are provided, the decryption will continue if those keys are found, regardless of the values. `key` and `key=value` elements can be mixed.

```
--encryption-context required_key classification=secret
```

Warning: If encryption context requirements are not satisfied by the ciphertext message, the message will not be decrypted. One side effect of this is that if you chose to write the plaintext output to a file and that file already exists, it will be deleted when we stop the decryption.

2.1.3 Output Metadata

In addition to the actual output of the operation, there is metadata about the operation that can be useful. This metadata includes some information about the operation as well as the complete header data from the ciphertext message.

The metadata for each operation is written to the specified file as a single line containing formatted JSON, so if a single command performs multiple file operations, a separate line will be written for each operation. There are three operating modes:

- `--metadata-output FILE` : Writes the metadata output to `FILE` (can be `-` for stdout as long as main output is not stdout). Default behavior is to append the metadata entry to the end of `FILE`.
- `--overwrite-metadata` : Force overwriting the contents of `FILE` with the new metadata.
- `-S/--suppress-metadata` : Output metadata is suppressed.

Metadata Contents

The metadata JSON contains the following fields:

- `"mode"` : `"encrypt"/"decrypt"`
- `"input"` : Full path to input file (or `"<stdin>"` if stdin)

- `"output"` : Full path to output file (or `"<stdout>"` if stdout)
- `"header"` : JSON representation of message header data
- `"header_auth"` : JSON representation of message header authentication data (only on decrypt)

Skipped Files

If encryption context checks fail when attempting to decrypt a file, the metadata contains additional fields:

- `skipped: true`
- `reason: "Missing encryption context key or value"`
- `missing_encryption_context_keys` : List of required encryption context keys that were missing from the message.
- `missing_encryption_context_pairs` : List of required encryption context key-value pairs missing from the message.

2.1.4 Master Key Provider

Information for configuring a master key provider must be provided.

Parameters may be provided using *Parameter Values*.

Required parameters:

- **provider** *(default: aws-encryption-sdk-cli::aws-kms)* : Indicator of the master key provider to use.
  - See *Advanced Configuration* for more information on using other master key providers.
- **key** *(at least one required, many allowed)* : Identifier for a master key to be used. Must be an identifier understood by the specified master key provider.
  - If using `aws-kms` to decrypt, you must not specify a key.

Any additional parameters supplied are collected into lists by parameter name and passed to the master key provider class when it is instantiated. Custom master key providers must accept all arguments as prepared. See *Advanced Configuration* for more information.

Multiple master keys can be defined using multiple instances of the `key` argument.

Multiple master key providers can be defined using multiple `--master-keys` groups.

If multiple master key providers are defined, the first one is treated as the primary.

If multiple master keys are defined in the primary master key provider, the first one is treated as the primary. The primary master key is used to generate the data key.

The below logic is used to construct all master key providers. We use `KMSMasterKeyProvider` as an example.

```
# With parameters:
--master-keys provider=aws-kms key=$KEY_1 key=$KEY_2

# KMSMasterKeyProvider is called as:
key_provider = KMSMasterKeyProvider()
key_provider.add_master_key($KEY_1)
key_provider.add_master_key($KEY_2)
```

```
# Single KMS CMK
--master-keys provider=aws-kms key=$KEY_ARN_1

# Two KMS CMKs
--master-keys provider=aws-kms key=$KEY_ARN_1 key=$KEY_ARN_2

# KMS Alias by name in default region
--master-keys provider=aws-kms key=$ALIAS_NAME

# KMS Alias by name in two specific regions
--master-keys provider=aws-kms key=$ALIAS_NAME region=us-west-2
--master-keys provider=aws-kms key=$ALIAS_NAME region=eu-central-1
```

AWS KMS

If you want to use the `aws-kms` master key provider, you can either specify that as the provider or simply not specify a provider and allow the default value to be used.

There are some configuration options which are unique to the `aws-kms` master key provider:

- **profile** : Providing this configuration value will use the specified named profile credentials.
- **region** : This allows you to specify the target region.

The logic for determining which region to use is shown in the pseudocode below:

```
if key ID is an ARN:
   use region identified in ARN
else:
   if region is specified:
      use region
   else if profile is specified and profile has a defined region:
      use region defined in profile
   else:
      use system default region
```

Advanced Configuration

If you want to use a different master key provider, that provider must register a setuptools entry point. You can find an example of registering this entry point in the `setup.py` for this package.

When a provider name is specifed in a call to `aws-encryption-cli`, the appropriate entry point for that name is used.

Handling Multiple Entry Points

If multiple entry points are registered for a given name, you will need to specify the package that registered the entry point you want to use.

In order to specify the package name, use the format: `PACKAGE_NAME::ENTRY_POINT`.

- `provider=aws-kms`
- `provider=aws-encryption-sdk-cli::aws-kms`

If you supply only an entry point name and there is only one entry point registered for that name, that entry point will be used.

If you supply only an entry point name and there is more than one entry point registered for that name, an error will be raised showing you all of the packages that have an entry point registered for that name.

If you supply both a package and an entry point name, that exact entry point will be used. If it is not accessible, an error will be raised showing you all of the packages that have an entry point registered for that name.

External Master Key Providers

The entry point name use must not contain the string `::`. This is used as a namespace separator as descibed in *Handling Multiple Entry Points*.

When called, these entry points must return an instance of a master key provider. They must accept the parameters prepared by the CLI as described in *Master Key Provider*.

These entry points must be registered in the `aws_encryption_sdk_cli.master_key_providers` group.

If the entry point raises a `aws_encryption_sdk_cli.exceptions.BadUserArgumentError`, the CLI will present the raised error message to the user to indicate bad user input.

2.1.5 Data Key Caching

Data key caching is optional, but if used then the parameters noted as required must be provided. For detailed information about using data key caching with the AWS Encryption SDK, see the data key caching documentation.

Parameters may be provided using *Parameter Values*.

Allowed parameters:

- **capacity** *(required)* : Number of entries that the cache will hold.
- **max_age** *(required)* : Determines how long each entry can remain in the cache, beginning when it was added.
- **max_messages_encrypted** : Determines how long each entry can remain in the cache, beginning when it was added.
- **max_bytes_encrypted** : Specifies the maximum number of bytes that a cached data key can encrypt.

2.1.6 Logging and Verbosity

The `-v` argument allows you to tune the verbosity of the built-in logging to your desired level. In short, the more `-v` arguments you supply, the more verbose the output gets.

- unset : `aws-encryption-cli` logs all warnings, all dependencies only log critical messages
- `-v` : `aws-encryption-cli` performs moderate logging, all dependencies only log critical messages
- `-vv` : `aws-encryption-cli` performs detailed logging, all dependencies only log critical messages
- `-vvv` : `aws-encryption-cli` performs detailed logging, all dependencies perform moderate logging
- `-vvvv` : `aws-encryption-cli` performs detailed logging, all dependencies perform detailed logging

python logging levels		
verbosity flag	aws-encryption-cli	dependencies
unset	WARNING	CRITICAL
-v	INFO	CRITICAL
-vv	DEBUG	CRITICAL
-vvv	DEBUG	INFO
-vvvv	DEBUG	DEBUG

2.1.7 Configuration Files

As with any CLI where the configuration can get rather complex, you might want to use a configuration file to define some or all of your desired behavior.

Configuration files are supported using Python's native argparse file support, which allows you to write configuration files exactly as you would enter arguments in the shell. Configuration file references passed to `aws-encryption-cli` are identified by the `@` prefix and the contents are expanded as if you had included them in line. Configuration files can have any name you desire.

Note: In PowerShell, you will need to escape the `@` symbol so that it is sent to `aws-encryption-cli` rather than interpreted by PowerShell.

For example, if I wanted to use a common master key configuration for all of my calls, I could create a file `master-key.conf` with contents detailing my master key configuration.

master-key.conf

```
--master-key key=A_KEY key=ANOTHER_KEY
```

Then, when calling `aws-encryption-cli`, I can specify the rest of my arguments and reference my new configuration file, and `aws-encryption-cli` will use the composite configuration.

```
aws-encryption-cli -e -i $INPUT_FILE -o $OUTPUT_FILE @master-key.conf
```

To extend the example, if I wanted a common caching configuration for all of my calls, I could similarly place my caching configuration in a configuration file `caching.conf` in this example and include both files in my call.

caching.conf

```
--caching capacity=10 max_age=60.0 max_messages_encrypted=15
```

```
aws-encryption-cli -e -i $INPUT_FILE -o $OUTPUT_FILE @master-key.conf @caching.conf
```

Configuration files can be referenced anywhere in `aws-encryption-cli` parameters.

```
aws-encryption-cli -e -i $INPUT_DIR -o $OUTPUT_DIR @master-key.conf @caching.conf --
↪recursive
```

Configuration files can have many lines, include comments using `#`. Escape characters are platform-specific: `\` on Linux and MacOS and `` ` `` on Windows. Configuration files may also include references to other configuration files.

my-encrypt.config

```
--encrypt
@master-key.conf # Use existing master key config
@caching.conf
# Always recurse, but require interactive overwrite.
--recursive
--interactive
```

```
aws-encryption-cli @my-encrypt -i $INPUT -o $OUTPUT
```

2.1.8 Encoding

By default, `aws-encryption-cli` will always output raw binary data and expect raw binary data as input. However, there are some cases where you might not want this to be the case.

Sometimes this might be for convenience:

- Accepting ciphertext through stdin from a human.
- Presenting ciphertext through stdout to a human.

Sometimes it might be out of necessity:

- Saving ciphertext output to a shell variable.
  - Most shells apply a system encoding to any data stored in a variable. As a result, this often results in corrupted data if binary data is stored without additional encoding.
- Piping ciphertext in PowerShell.
  - Similar to the above, all data passed through a PowerShell pipe is encoded using the system encoding.

In order to address these scenarios, we provide two optional arguments:

- `--decode` : Base64-decode input before processing.
- `--encode` : Base64-encode output after processing.

These can be used independently or together, on any valid input or output.

Be aware, however, that if you target multiple files either through a path expansion or by targetting a directory, the requested decoding/encoding will be applied to all files.

2.2 Execution

```
usage: aws-encryption-cli [-h] [--version] [-e] [-d] [-S]
                  [--metadata-output METADATA_OUTPUT] [--overwrite-metadata]
                  [-m MASTER_KEYS [MASTER_KEYS ...]]
                  [--caching CACHING [CACHING ...]] -i INPUT -o OUTPUT
                  [--encode] [--decode]
                  [-c ENCRYPTION_CONTEXT [ENCRYPTION_CONTEXT ...]]
                  [--algorithm {
                     AES_256_GCM_IV12_TAG16_HKDF_SHA384_ECDSA_P384,
                     AES_192_GCM_IV12_TAG16_HKDF_SHA384_ECDSA_P384,
                     AES_128_GCM_IV12_TAG16_HKDF_SHA256_ECDSA_P256,
                     AES_256_GCM_IV12_TAG16_HKDF_SHA256,
                     AES_192_GCM_IV12_TAG16_HKDF_SHA256,
                     AES_128_GCM_IV12_TAG16_HKDF_SHA256,
```

(continues on next page)

(continued from previous page)

```
                    AES_256_GCM_IV12_TAG16,
                    AES_192_GCM_IV12_TAG16,
                    AES_128_GCM_IV12_TAG16
                }]
                [--frame-length FRAME_LENGTH] [--max-length MAX_LENGTH]
                [--suffix [SUFFIX]] [--interactive] [--no-overwrite] [-r]
                [-v] [-q]

Encrypt or decrypt data using the AWS Encryption SDK

optional arguments:
  -h, --help            show this help message and exit
  --version             show program's version number and exit
  -e, --encrypt         Encrypt data
  -d, --decrypt         Decrypt data
  -S, --suppress-metadata
                        Suppress metadata output.
  --metadata-output METADATA_OUTPUT
                        File to which to write metadata records
  --overwrite-metadata  Force metadata output to overwrite contents of file
                        rather than appending to file
  -m MASTER_KEYS [MASTER_KEYS ...], --master-keys MASTER_KEYS [MASTER_KEYS ...]
                        Identifying information for a master key provider and
                        master keys. Each instance must include a master key
                        provider identifier and identifiers for one or more
                        master key supplied by that provider. ex: --master-
                        keys provider=aws-kms key=$AWS_KMS_KEY_ARN
  --caching CACHING [CACHING ...]
                        Configuration options for a caching cryptographic
                        materials manager and local cryptographic materials
                        cache. Must consist of "key=value" pairs. If caching,
                        at least "capacity" and "max_age" must be defined. ex:
                        --caching capacity=10 max_age=100.0
  -i INPUT, --input INPUT
                        Input file or directory for encrypt/decrypt operation,
                        or "-" for stdin.
  -o OUTPUT, --output OUTPUT
                        Output file or directory for encrypt/decrypt
                        operation, or - for stdout.
  --encode              Base64-encode output after processing
  --decode              Base64-decode input before processing
  -c ENCRYPTION_CONTEXT [ENCRYPTION_CONTEXT ...], --encryption-context ENCRYPTION_
↪CONTEXT [ENCRYPTION_CONTEXT ...]
                        key-value pair encryption context values (encryption
                        only). Must a set of "key=value" pairs. ex: -c
                        key1=value1 key2=value2
  --algorithm {
         AES_256_GCM_IV12_TAG16_HKDF_SHA384_ECDSA_P384,
         AES_192_GCM_IV12_TAG16_HKDF_SHA384_ECDSA_P384,
         AES_128_GCM_IV12_TAG16_HKDF_SHA256_ECDSA_P256,
         AES_256_GCM_IV12_TAG16_HKDF_SHA256,
         AES_192_GCM_IV12_TAG16_HKDF_SHA256,
         AES_128_GCM_IV12_TAG16_HKDF_SHA256,
         AES_256_GCM_IV12_TAG16,
         AES_192_GCM_IV12_TAG16,
         AES_128_GCM_IV12_TAG16
      }
```

(continues on next page)

(continued from previous page)

```
                        Algorithm name (encryption only)
  --frame-length FRAME_LENGTH
                        Frame length in bytes (encryption only)
  --max-length MAX_LENGTH
                        Maximum frame length (for framed messages) or content
                        length (for non-framed messages) (decryption only)
  --suffix [SUFFIX]     Custom suffix to use when target filename is not
                        specified (empty if specified but no value provided)
  --interactive         Force aws-encryption-cli to prompt you for verification before
                        overwriting existing files
  --no-overwrite        Never overwrite existing files
  -r, -R, --recursive   Allow operation on directories as input
  -v                    Enables logging and sets detail level. Multiple -v
                        options increases verbosity (max: 4).
  -q, --quiet           Suppresses most warning and diagnostic messages

For more usage instructions and examples, see: http://aws-encryption-sdk-cli.
↪readthedocs.io/en/latest/
```

CHAPTER 3

Modules

aws_encryption_sdk_cli	AWS Encryption SDK CLI.
aws_encryption_sdk_cli.internal	Internal implementation details.
aws_encryption_sdk_cli.internal.arg_parsing	Helper functions for parsing and processing input arguments.
aws_encryption_sdk_cli.internal.encoding	
aws_encryption_sdk_cli.internal.identifiers	Static identifier values for the AWS Encryption SDK CLI.
aws_encryption_sdk_cli.internal.io_handling	Helper functions for handling all input and output for this CLI.
aws_encryption_sdk_cli.internal.master_key_parsing	Helper functions for building crypto materials manager and underlying master key provider(s) from arguments.

3.1 aws_encryption_sdk_cli

AWS Encryption SDK CLI.

Functions

cli([raw_args])	CLI entry point.
process_cli_request(stream_args, parsed_args)	Maps the operation request to the appropriate function based on the type of input and output provided.
stream_kwargs_from_args(args, ...)	Builds kwargs object for aws_encryption_sdk.stream based on argparse arguments and existing CryptoMaterialsManager.

aws_encryption_sdk_cli.**process_cli_request**(*stream_args*, *parsed_args*)

Maps the operation request to the appropriate function based on the type of input and output provided.

Parameters

- **stream_args** (*dict*) – kwargs to pass to *aws_encryption_sdk.stream*
- **args** (*argparse.Namespace*) – Parsed arguments from argparse

aws_encryption_sdk_cli.**stream_kwargs_from_args**(*args*, *crypto_materials_manager*)
Builds kwargs object for aws_encryption_sdk.stream based on argparse arguments and existing CryptoMaterialsManager.

Parameters

- **args** (*argparse.Namespace*) – Parsed arguments from argparse
- **crypto_materials_manager** (*aws_encryption_sdk.materials_manager.base.CryptoMaterialsManager*) – Existing CryptoMaterialsManager

Returns Translated kwargs object for aws_encryption_sdk.stream

Return type dict

aws_encryption_sdk_cli.**cli**(*raw_args=None*)
CLI entry point. Processes arguments, sets up the key provider, and processes requested action.

Returns Execution return value intended for `sys.exit()`

3.2 aws_encryption_sdk_cli.internal

Internal implementation details.

Warning: No guarantee is provided on the modules and APIs within this namespace staying consistent. Directly reference at your own risk.

3.3 aws_encryption_sdk_cli.internal.arg_parsing

Helper functions for parsing and processing input arguments.

Functions

parse_args([raw_args])	Handles argparse to collect the needed input values.

Classes

CommentIgnoringArgumentParser(*args, **kwargs)	ArgumentParser that ignores lines in fromfile_prefix_chars files which start with #.
UniqueStoreAction(option_strings, dest[, ...])	argparse action that requires that arguments cannot be repeated.

aws_encryption_sdk_cli.internal.arg_parsing.**parse_args**(*raw_args=None*)
Handles argparse to collect the needed input values.

Parameters **raw_args** (*list*) – List of arguments

Returns parsed arguments

Return type argparse.Namespace

3.4 aws_encryption_sdk_cli.internal.identifiers

Static identifier values for the AWS Encryption SDK CLI.

Classes

OperationResult(needs_cleanup)	Identifies the resulting state of an operation.

aws_encryption_sdk_cli.internal.identifiers.**OUTPUT_SUFFIX = {'decrypt': '.decrypted', 'enc**
Suffix added to output files if specific output filename is not specified.

class aws_encryption_sdk_cli.internal.identifiers.**OperationResult**(*needs_cleanup*)
Bases: enum.Enum

Identifies the resulting state of an operation.

Parameters **needs_cleanup** (*bool*) – If true, the output file needs to be deleted

Prepares new OperationResult.

3.5 aws_encryption_sdk_cli.internal.io_handling

Helper functions for handling all input and output for this CLI.

Functions

output_filename(source_filename, ...)	Duplicates the source filename in the destination directory, adding or stripping a suffix as needed.

Classes

IOHandler(metadata_writer, interactive, ...)	Common handler for all IO operations.

aws_encryption_sdk_cli.internal.io_handling.**output_filename**(*source_filename*, *destination_dir*, *mode*, *suffix*)
Duplicates the source filename in the destination directory, adding or stripping a suffix as needed.

Parameters

- **source_filename** (*str*) – Full file path to source file

- **destination_dir** (*str*) – Full file path to destination directory
- **mode** (*str*) – Operating mode (encrypt/decrypt)
- **suffix** (*str*) – Suffix to append to output filename

Returns Full file path of new destination file in destination directory

Return type str

class aws_encryption_sdk_cli.internal.io_handling.**IOHandler**(*metadata_writer*, *interactive*, *no_overwrite*, *decode_input*, *encode_output*, *required_encryption_context*, *required_encryption_context_keys*)

Bases: object

Common handler for all IO operations. Holds common configuration values used for all operations.

Parameters

- **metadata_writer** (*aws_encryption_sdk_cli.internal.metadata.MetadataWriter*) – File-like to which metadata should be written
- **interactive** (*bool*) – Should prompt before overwriting existing files
- **no_overwrite** (*bool*) – Should never overwrite existing files
- **decode_input** (*bool*) – Should input be base64 decoded before operation
- **encode_output** (*bool*) – Should output be base64 encoded after operation
- **required_encryption_context** (*dict*) – Encryption context key-value pairs to require
- **required_encryption_context_keys** (*list*) – Encryption context keys to require

Workaround pending resolution of attrs/mypy interaction. https://github.com/python/mypy/issues/2088 https://github.com/python-attrs/attrs/issues/215

process_single_operation(*stream_args*, *source*, *destination*)

Processes a single encrypt/decrypt operation given a pre-loaded source.

Parameters

- **stream_args** (*dict*) – kwargs to pass to *aws_encryption_sdk.stream*
- **source** (*str or file-like object*) – source to write
- **destination** (*str*) – destination identifier

Returns OperationResult stating whether the file was written

Return type *aws_encryption_sdk_cli.internal.identifiers.OperationResult*

process_single_file(*stream_args*, *source*, *destination*)

Processes a single encrypt/decrypt operation on a source file.

Parameters

- **stream_args** (*dict*) – kwargs to pass to *aws_encryption_sdk.stream*
- **source** (*str*) – Full file path to source file

- **destination** (*str*) – Full file path to destination file

process_dir(*stream_args*, *source*, *destination*, *suffix*)
Processes encrypt/decrypt operations on all files in a directory tree.

Parameters

- **stream_args** (*dict*) – kwargs to pass to *aws_encryption_sdk.stream*
- **source** (*str*) – Full file path to source directory root
- **destination** (*str*) – Full file path to destination directory root
- **suffix** (*str*) – Suffix to append to output filename

3.6 aws_encryption_sdk_cli.internal.master_key_parsing

Helper functions for building crypto materials manager and underlying master key provider(s) from arguments.

Functions

build_crypto_materials_manager_from_args(...)	Builds a cryptographic materials manager from the provided arguments.

aws_encryption_sdk_cli.internal.master_key_parsing.**build_crypto_materials_manager_from_args**

Builds a cryptographic materials manager from the provided arguments.

Parameters

- **key_providers_config** (*list*) – List of one or more dicts containing key provider configuration
- **caching_config** (*dict*) – Parsed caching configuration

Return type aws_encryption_sdk.materials_managers.base.CryptoMaterialsManager

CHAPTER 4

Changelog

4.1 1.1.7 – 2019-10-15

4.1.1 Operational

- Completely remove `typing` as an install requirement. #166 #167

4.2 1.1.6 – 2019-09-30

4.2.1 Operational

- Update requirements to only require the typing module for Python versions earlier than 3.5. #165

4.3 1.1.5 – 2018-08-01

4.3.1 Operational

- Remove base64 stream encoding/decoding logic in in favor of `base64io` library. #154
- Move the `aws-encryption-sdk-cli` repository from `awslabs` to `aws`.

4.4 1.1.4 – 2018-01-15

4.4.1 Bugfixes

- Fixed config file handling of quotes in Windows #110

4.5 1.1.3 – 2017-12-05

4.5.1 Bugfixes

- Blacklist pytest 3.3.0 #125 pytest-dev/pytest#2956
- Expand input and output file paths in metadata #120
- Move metadata file writer to write in binary #121
- Skip symlink tests when running tests in Windows #128

4.5.2 Operational

- Move integration tests away from using config files to using environment variables #62

4.6 1.1.2 – 2017-11-22

4.6.1 Bugfixes

- Fixed permissions issue from installing metadata files #122

4.7 1.1.1 – 2017-11-21

4.7.1 Bugfixes

- Fixed import issue with Python 3.5.0 and 3.5.1 #114

4.8 1.1.0 – 2017-11-18

Public release

4.8.1 Known Issues

- Single and double quote characters break config file parsing on Windows platforms #110 #111
- typing imports fail on Python 3.5.0 and 3.5.1 #114 #115

4.8.2 Bugfixes

- Handle quoting in config files #35
- Allow empty custom suffix #33
- Handle non-POSIX paths in config files in non-POSIX environments #78
- Expand user (~) and environment variables in config files #89
- Parameter key-value pairs will no longer accept empty key or value elements #94

4.8.3 New Features

- Built-in base64 encoding and decoding #29
- Strip plaintext data keys from boto3 logs #54
- Enforce that parent directories always exist #57 #100
- Catch single-dash dummy argument catchers for long-form arguments #5
- Optionally output operation metadata #65
- Optionally encryption context enforcement on decrypt #69

4.8.4 Operational

- Custom master key providers now handled through setuptools entry points #30
- Default master key provider is now namespace-specific #81
- PyPI-Parker configuration and tox testenv added #36
- Custom user agent value added to generated botocore client #70
- AWS KMS master key provider configuration will no longer accept `key` parameter #80

4.9 1.0.2

4.9.1 Bugfixes

- Fixed helpstring output to show input/output as required #1
- Fixed bug when processing encrypt request with no master key provider configuration #3
- Fixed caching CMM construction failure #9

4.9.2 New Features

- Added support for filename expansion #4
- Added ability to specify profile and region for KMSMasterKeyProvider using AWS CLI-like syntax #6
- Reworked verbosity configuration to be more useful #10
- Addded ability to define custom output filename suffix #12

4.9.3 Operational

- Added mypy coverage #13

4.10 1.0.1

- Updated aws-encryption-sdk dependency to `>=1.3.2` to pull in fix for #7

4.11 1.0.0

- Initial creation

Python Module Index

a

Index