TE004 Test Report on the Module codecs_lib.xor_scrambler

Conventions

Each test is defined following the same format. Each test receives a unique test identifier and a reference to the ID(s) of the requirements it covers (if applicable). The goal of the test is described to clarify what is to be tested. The test steps are described in brief but clear instructions. For each test it is defined what the expected results are for the test to pass. Finally, the test result is given, this can be only pass or fail.

The test format is as follows:

Test Identifier: TEST-[I/A/D/T]-XYZ

Requirement ID(s): REQ-uvw-xyz

Verification method: I/A/D/T

Test goal: Description of what is to be tested

Expected result: What test result is expected for the test to pass

Test steps: Step by step instructions on how to perform the test

Test result: PASS/FAIL

The test ID starts with the fixed prefix 'TEST'. The prefix is followed by a single letter, which defines the test type / verification method. The last part of the ID is a 3-digits *hexadecimal* number (0..9|A..F), with the first digit identifing the module, the second digit identifing a class / function, and the last digit - the test ordering number for this object. E.g. 'TEST-T-112'. Each test type has its own counter, thus 'TEST-T-112' and 'TEST-A-112' tests are different entities, but they refer to the same object (class or function) within the same module.

The verification method for a requirement is given by a single letter according to the table below:

Term	Definition
Inspection (I)	Control or visual verification
Analysis (A)	Verification based upon analytical evidences
Test (T)	Verification of quantitative characteristics with quantitative measurement
Demonstration (D)	Verification of operational characteristics without quantitative measurement

Test preparation

Prepare a set of simple byte string examples with the length from 1 to 6 bytes, which XORed version can be easily computed - store them as pairs 'original' : 'encoded'. Also prepare a dictionary of XORed (manually

filled) byte string representation of a complex Unicode string using different codecs (utf-8, utf-16-le, utf-32-le) using the names of the codecs as the keys.

Test definitions (Test)

Test Identifier: TEST-T-400

Requirement ID(s): REQ-FUN-401, REQ-FUN-410

Verification method: T

Test goal: Test that an arbitrary byte string, bytes array or string (Unicode) is properly XOR encoded, taking the requested codec into account.

Expected result: All byte string comparisons pass, no exception is raised.

Test steps: Perform the following checks

- Pass each of the original example simple byte strings into the encoding method and compare the returned byte string with the expected encoded counter part
- Convert into a bytes array and pass each of the original example simple byte strings into the encoding method and compare the returned byte string with the expected encoded counter part
- Pass the complex string into the encoder together with the keyword argument specifying the codec and compare the returned byte string with the expected encoded counter part for that codec.
- Pass the complex string into the encoder without Encoding argument and with it set to None, compare the returned results with the expected value for the UTF8 codec.

Each time the byte strings must be equal. **N.B.** implemented as a test case in the test suit module codecs_lib.tests.ut004_xor_scrambler.py

Test result: PASS

Test Identifier: TEST-T-401

Requirement ID(s): REQ-FUN-401, REQ-FUN-420

Verification method: T

Test goal: Test that the properly XOR encoded input can be decoded back into the original byte string or Unicode string using the same codec.

Expected result: All byte and Unicode strings comparisons pass, no exception is raised.

Test steps: Perform the following checks

- Pass each of the original example simple byte strings encoded counterpart into the decoding method and compare the returned byte string with the original
- Convert into a bytes array and pass each of the original example simple byte strings encoded counterpart into the encoding method and compare the returned byte string with the original
- Pass each of the encoded byte strings representing the complex Unicode example together with the respective codec as the value of the keyword argument and compare the returned value with the

original complex Unicode string
- Repeat the previous step passing the byte strings converted into byte arrays

Each time the byte strings must be equal. **N.B.** implemented as a test case in the test suit module codecs_lib.tests.ut004_xor_scrambler.py

Test result: PASS

Test Identifier: TEST-T-402

Requirement ID(s): REQ-AWM-400

Verification method: T

Test goal: Only **bytearray** or **bytestring** instances are allowed as an input for the decoding function / method. Only **bytearray** or **bytestring** or **str** instances are allowed as an input for the encoding function / method. An non-string keyword argument *Encoding* is not acceptable for the decoding method, and it is not acceptable for the encoding method if the mandatory argument is a string.

Expected result: TypeError (sub-class of) exception is raised during the respective unit test.

Test steps: Perform the following checks:

- Pass different data types, including Unicode strings, but excepting **bytearray** or **bytestring** instances as the mandatory argument into the decoder
- Pass a byte string as the mandatory argument into the decoder by different data types except string as the keyword
- Pass different data types, including Unicode strings, but excepting **bytearray**, **bytestring** and **str** instances as the mandatory argument into the encoder
- Pass a string as the mandatory argument into the encoder by different data types except string as the keyword

Each time the **TypeError** or its sub-class must be raised. **N.B.** implemented as a test case in the test suit module codecs_lib.tests.ut004_xor_scrambler.py

Test result: PASS

Test Identifier: TEST-T-403

Requirement ID(s): REQ-AWM-401

Verification method: T

Test goal: Only Python registered codecs may be specified via the keyword argument, and they must be compatible with the string to be encoded / byte string (XOR unscrambled) to be decoded.

Expected result: ValueError (sub-class of) exception is raised during the respective unit test.

Test steps: Perform the following checks:

- Try to encode the prepared complex Unicode string with a registered, but unsuitable codec, e.g. 'ascii'
- Try to encode the prepared complex Unicode string with a non-registered codec, e.g. 'ascii2'
- Try to decode the XORed UTF-8 representation of the prepared complex Unicode string with a registered, but unsuitable codec, e.g. 'ascii'
- Try to decode an arbitrary binary string or bytes array with a non-registered codec, e.g. 'ascii2'

Each time the **ValueError** or its sub-class must be raised. **N.B.** implemented as a test case in the test suit module codecs_lib.tests.ut004_xor_scrambler.py

Test result: PASS

Test definitions (Analysis)

Test Identifier: TEST-A-400

Requirement ID(s): REQ-FUN-400

Verification method: A

Test goal: Test that the module implement all of the required functionality.

Expected result: All of the unit tests defined by the test cases TEST-T-400 to TEST-T-403 must pass.

Test steps: Run the test suit module codecs_lib.tests.ut004_xor_scrambler.py

Test result: PASS

Traceability

For traceability the relation between tests and requirements is summarized in the table below:

Requirement ID	**Covered in test(s)**	**Verified [YES/NO])**
REQ-FUN-400	TEST-A-400	YES
REQ-FUN-401	TEST-T-400, TEST-T-401	YES
REQ-FUN-410	TEST-T-400	YES
REQ-FUN-420	TEST-T-401	YES
REQ-AWM-400	TEST-T-402	YES
REQ-AWM-401	TEST-T-403	YES

Software ready for production [YES/NO]	**Rationale**
YES	All tests are passed