RE004 Requirements for the Module codecs_lib.xor_scrambler

Conventions

Requirements listed in this document are constructed according to the following structure:

Requirement ID: REQ-UVW-XYZ

Title: Title / name of the requirement

Description: Descriprion / definition of the requirement

Verification Method: I / A / T / D

The requirement ID starts with the fixed prefix 'REQ'. The prefix is followed by 3 letters abbreviation (in here 'UVW'), which defines the requiement type - e.g. 'FUN' for a functional and capability requirement, 'AWM' for an alarm, warnings and operator messages, etc. 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 requirement ordering number for this object. E.g. 'REQ-FUN-112'. Each requirement type has its own counter, thus 'REQ-FUN-112' and 'REQ-AWN-112' requirements 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

Functional and capability requirements

Requirement ID: REQ-FUN-400

Title: XOR scrambler implementation

Description: The module must provide a proper implementation of a simple scrambler / un-scrambler algorithm using per-byte XOR operation with the value 255 ('0xFF').

Verification Method: A

Requirement ID: REQ-FUN-401

Title: API

Description: The module should provide two functions or class methods: one for encoding of the data and one for decoding of the data

Verification Method: T

Requirement ID: REQ-FUN-410

Title: Data encoding

Description: The function / method for encoding of the data should accept an arbitrary byte string (type **bytes**), byte array (type **bytearray**) or string (Unicode in Python3 by default) and return a byte string, with each byte within being the result of XORing of the original byte in the same position with the 255 value. In case of a string argument the UTF-8 encoding is used to convert it into a bytestring unless a different encoding is explicitely specified using a keyword-only argument *Encoding*, which value must be supported by the Python language, see Codecs registry and base classes.

Verification Method: T

Requirement ID: REQ-FUN-420

Title: Data decoding

Description: The function / method for decoding of the data should accept an arbitrary byte string (type **bytes**) or byte array (type **bytearray**) object as its argument, apply per byte XORing with the 255 value and return the result as a byte string. Only if a keyword-only argument *Encoding* is provided the resulted byte string is to be converted (decoded) into a Unicode string using the specified codec.

Verification Method: T

Alarms, warnings and operator messages

Requirement ID: REQ-AWM-400

Title: Improper input type for encoding or decoding raises an exception

Description: The passed mandatory argument is neither of a byte string, a byte array or a string; OR the passed keyword (for the string mandatory argument) is neither *None* nor a string. **TypeError** exception or its sub-class must be raised.

Verification Method: T

Requirement ID: REQ-AWM-401

Title: Improper input value of a proper type raises an exception

Description: A passed string (Unicode) argument cannot be encoded into a byte string using the specified codec, OR a decoded byte string / bytes array cannot be decoded into a string using the specificed codec, OR such codec is not registered. **ValueError** exception or its sub-class must be raised.

Verification Method: T