RE003 Requirements for the Module codecs_lib.wichmann_hill

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-300

Title: Pseudo-random numbers codec implementation

Description: The module must provide a proper implementation of the encoding and decoding based on the arithmetic operations between the data and pseudo-random numbers generated using a deterministic algorithm starting from some 'seed' number(s) serving as the key.

Verification Method: A

Requirement ID: REQ-FUN-301

Title: Module interface

Description: The module should provide three class instance methods:

- For setting the encoding / decoding key - i.e. the seed for the pseudo-random numbers generator
- For encoding of an arbitrary integer or floating point number, or a generic sequence of such numbers into a single floating point number or a list of the floats respectively string
- For decoding of an arbitrary integer or floating point number, or a generic sequence of such numbers into a single floating point number or a list of the floats respectively string

Verification Method: A

Requirement ID: REQ-FUN-302

Title: Support for a continuous data feed

Description: Each encoded or decoded integer or floating point number should use a new generated pseudo-random number, and the generator shouldn't be re-seeded unless explicitely requested by the client via the respective method call. Thus, this codec can be used for the encoding and decoding of a continuous data feed, e.g. of an input stream.

Verification Method: T

Requirement ID: REQ-FUN-310

Title: Pseudo-random number generator's algorithm

Description: For the compatibility with the legacy code the specific variant of the Wichmann-Hill generator algorithm should be used:

```
s1 = (171 * (s1 % 177) - 2 * int(s1 / 177)) % 30269
s2 = (172 * (s2 % 176) - 35 * int(s1 / 176)) % 30307
s3 = (170 * (s3 % 178) - 63 * int(s1 / 178)) % 30323
r = s1 / 30269 + s2 / 30307 + s3 / 30323
r -= int(r)
```

I.e., with each call to generate the random number the current values of the seed numbers are re-calculated first, and the requested number is calculated from the already updated seed values.

In this variant, mathematically, all seed values *s1*, *s2* and *s3* are always strictly positive and are upper-bound by the value ~ 30000, whereas the generated pseudo-random numbers form a double-open interval (0, 1), although the zero value is theoretically possible due to round-off error (depending on the platform).

Verification Method: T

Requirement ID: REQ-FUN-320

Title: Data encoding algorithm

Description: Each received number (as a single number argument, or as an element of a sequence) should encoded as follows: *InData* -> *OutData* : *OutData* = *InData* / (*PRN* + 1.0E-6), where *PRN* is a pseudo-random number obtained for each new number to be processed.

Verification Method: T

Requirement ID: REQ-FUN-321

Title: Data decoding algorithm

Description: Each received number (as a single number argument, or as an element of a sequence) should decoded as follows: *InData* -> *OutData* : *OutData* = *InData* * (*PRN* + 1.0E-6), where *PRN* is a pseudo-random number obtained for each new number to be processed.

Verification Method: T

Description:

Alarms, warnings and operator messages

Requirement ID: REQ-AWM-300

Title: Improper type of the generator's seeds

Description: The pseudo-random number generator can be seeded only with the integer numbers - directly or via the method of the coder using this generator. Otherwise - not integer arguments - the sub-class of the **TypeError** exception should be raised.

Verification Method: T

Requirement ID: REQ-AWM-301

Title: Improper values range of the generator's seeds

Description: The pseudo-random number generator can be seeded only with the positive integer numbers - directly or via the method of the coder using this generator. Otherwise - zero or negative integers - the sub-class of the **ValueError** exception should be raised.

Verification Method: T

Requirement ID: REQ-AWM-320

Title: Improper type of the coder input

Description: Only a single integer / floating point number, OR a sequence of such numbers can be accepted as the input data for the both encoding and decoding methods. Otherwise the sub-class of the **TypeError** exception should be raised.

Verification Method: T