# Author: Trevor Perrin

# See the LICENSE file for legal information regarding use of this file.

"""Classes for reading/writing binary data (such as TLS records)."""

from .compat import \*

class Writer(object):

def \_\_init\_\_(self):

self.bytes = bytearray(0)

def add(self, x, length):

self.bytes += bytearray(length)

newIndex = len(self.bytes) - 1

for count in range(length):

self.bytes[newIndex] = x & 0xFF

x >>= 8

newIndex -= 1

def addFixSeq(self, seq, length):

for e in seq:

self.add(e, length)

def addVarSeq(self, seq, length, lengthLength):

self.add(len(seq)\*length, lengthLength)

for e in seq:

self.add(e, length)

class Parser(object):

def \_\_init\_\_(self, bytes):

self.bytes = bytes

self.index = 0

def get(self, length):

if self.index + length > len(self.bytes):

raise SyntaxError()

x = 0

for count in range(length):

x <<= 8

x |= self.bytes[self.index]

self.index += 1

return x

def getFixBytes(self, lengthBytes):

if self.index + lengthBytes > len(self.bytes):

raise SyntaxError()

bytes = self.bytes[self.index : self.index+lengthBytes]

self.index += lengthBytes

return bytes

def getVarBytes(self, lengthLength):

lengthBytes = self.get(lengthLength)

return self.getFixBytes(lengthBytes)

def getFixList(self, length, lengthList):

l = [0] \* lengthList

for x in range(lengthList):

l[x] = self.get(length)

return l

def getVarList(self, length, lengthLength):

lengthList = self.get(lengthLength)

if lengthList % length != 0:

raise SyntaxError()

lengthList = lengthList // length

l = [0] \* lengthList

for x in range(lengthList):

l[x] = self.get(length)

return l

def startLengthCheck(self, lengthLength):

self.lengthCheck = self.get(lengthLength)

self.indexCheck = self.index

def setLengthCheck(self, length):

self.lengthCheck = length

self.indexCheck = self.index

def stopLengthCheck(self):

if (self.index - self.indexCheck) != self.lengthCheck:

raise SyntaxError()

def atLengthCheck(self):

if (self.index - self.indexCheck) < self.lengthCheck:

return False

elif (self.index - self.indexCheck) == self.lengthCheck:

return True

else:

raise SyntaxError()