Hex Encoder

```
//http://www.bouncycastle.org/
//MIT X11 License
using System;
using System.IO;

namespace Org.BouncyCastle.Utilities.Encoders
{
public class HexEncoder
{
private static readonly byte[] encodingTable =
{
(byte)'0', (byte)'1', (byte)'2', (byte)'3', (byte)'4',
(byte)'5', (byte)'6', (byte)'7',
(byte)'8', (byte)'9', (byte)'a', (byte)'b', (byte)'c',
(byte)'d', (byte)'e', (byte)'f'
};

/*
* set up the decoding table.
*/
internal static readonly byte[] decodingTable = new byte[128];

static HexEncoder()
{
for (int i = 0; i < encodingTable.Length; i++) {
decodingTable[encodingTable[i]] = (byte)i; }
decodingTable['A'] = decodingTable['a']; decodingTable['B'] =
decodingTable['b']; decodingTable['C'] = decodingTable['c'];
decodingTable['D'] = decodingTable['d']; decodingTable['E'] =
decodingTable['e']; decodingTable['F'] = decodingTable['f']; }
/** * encode the input data producing a Hex output stream. * *
@return the number of bytes produced. */ public int Encode(
byte[] data, int off, int length, Stream outStream) { for (int
i = off; i < (off + length); i++) { int v = data[i];
```

```
outStream.WriteByte(encodingTable[v >> 4]);
outStream.WriteByte(encodingTable[v & 0xf]);
}

return length * 2;
}

private bool ignore(
char c)
{
return (c == '
' || c =='
' || c == ' ' || c == ' ');
}

/**
* decode the Hex encoded byte data writing it to the given
output stream,
* whitespace characters will be ignored.
*
* @return the number of bytes produced.
*/
public int Decode(
byte[] data,
int off,
int length,
Stream outStream)
{
byte b1, b2;
int outLen = 0;
int end = off + length;

while (end > off)
{
if (!ignore((char)data[end – 1]))
{
break;
}
```

```
end–;
}

int i = off;
while (i < end) { while (i < end && ignore((char)data[i])) {
i++; } b1 = decodingTable[data[i++]]; while (i < end &&
ignore((char)data[i])) { i++; } b2 = decodingTable[data[i++]];
outStream.WriteByte((byte)((b1 < 0)
{
if (!ignore(data[end – 1]))
{
break;
}

end–;
}

int i = 0;
while (i < end) { while (i < end && ignore(data[i])) { i++; }
b1 = decodingTable[data[i++]]; while (i < end &&
ignore(data[i])) { i++; } b2 = decodingTable[data[i++]];
outStream.WriteByte((byte)((b1 <
```