[MS-XCA]: Xpress Compression Algorithm

This topic lists the Errata found in [MS-XCA] since it was last published.
Since this topic is updated frequently, we recommend that you subscribe to these RSS or Atom feeds to receive update notifications.
Errata are subject to the same terms as the Open Specifications documentation referenced.

RSS
Atom

Errata below are for Protocol Document Version V6.0 – 2020/03/04.

Errata Published*	Description

2020/08/17	In Section 2.2.4 Processing, we corrected the pseudocode to remove extraneous implementation-specific processing.
	
	Changed from:
	
	`Loop until a decompression terminating condition`
	`  Check for EOF`
	`  Build the decoding table`
	`  CurrentPosition += 256  // start at the end of the Huffman table`
	`  NextBits = Read16Bits(InputBuffer + CurrentPosition)`
	`  CurrentPosition += 2`
	`  NextBits <<= 16`
	`  NextBits |= Read16Bits(InputBuffer + CurrentPosition)`
	`  CurrentPosition += 2`
	`  ExtraBits = 16`
	`  BlockEnd = OutputPosition + 65536`
	
	`  Loop until a block terminating condition`
	`  Loop until a literal processing terminating condition`
	`  If OutputPosition >= BlockEnd then terminate block processing`
	`  Next15Bits = NextBits >> (32 - 15)`
	`  HuffmanSymbol = DecodingTable[Next15Bits]`
	`  HuffmanSymbolBitLength = the bit length of HuffmanSymbol, from the table in`
	`  the input buffer`
	`  If HuffmanSymbol <= 0`
	`  NextBits <<= HuffmanSymbolBitLength`
	`  ExtraBits -= HuffmanSymbolBitLength`
	
	`  Do`
	`  HuffmanSymbol = - HuffmanSymbol`
	`  HuffmanSymbol += (NextBits >> 31)`
	`  NextBits *= 2`
	`  ExtraBits = ExtraBits - 1`
	`  HuffmanSymbol = DecodingTable[HuffmanSymbol]`
	`  While HuffmanSymbol <= 0`
	`  Else`
	`  DecodedBitCount = HuffmanSymbol & 15`
	`  NextBits <<= DecodedBitCount`
	`  ExtraBits -= DedcodedBitCount`
	`  HuffmanSymbol >>= 4 // Shift by 4 bits to get the symbol value`
	`  // (the lower 4 bits are the bit length of the symbol)`
	`  HuffmanSymbol -= 256`
	`  If ExtraBits < 0`
	`  NextBits |= Read16Bits(InputBuffer + CurrentPosition) << (-ExtraBits)`
	`  ExtraBits += 16`
	`  CurrentPosition += 2`
	`  If HuffmanSymbol >= 0`

Errata Published*	Description

```
                            If HuffmanSymbol == 0
                                If the entire input buffer has been read and
                                the expected decompressed size has been written
    to the output buffer
                                    Decompression is complete.  Return with
    success.
                            Terminate literal processing
                        Else
                            Output the byte value of HuffmanSymbol to the output
    stream
                End of literal processing Loop

                MatchLength = HuffmanSymbol mod 16
                MatchOffsetBitLength = HuffmanSymbol / 16
                If MatchLength == 15
                    MatchLength = ReadByte(InputBuffer + CurrentPosition)
                    CurrentPosition += 1
                    If MatchLength == 255
                        MatchLength = Read16Bits(InputBuffer + CurrentPosition)
                        CurrentPosition += 2
                        If MatchLength < 15
                            The compressed data is invalid. Return error.
                        MatchLength = MatchLength - 15
                    MatchLength = MatchLength + 15
                MatchLength = MatchLength + 3
                MatchOffset = NextBits >> (32 - MatchOffsetBitLength)
                MatchOffset += (1 << MatchOffsetBitLength)
                NextBits <<= MatchOffsetBitLength
                ExtraBits -= MatchOffsetBitLength
                If ExtraBits < 0
                    NextBits |= Read16Bits(InputBuffer + CurrentPosition) << (-
    ExtraBits)
                    ExtraBits += 16
                    CurrentPosition += 2
                For i = 0 to MatchLength - 1
                    Output OutputBuffer[OutputPosition – MatchOffset + i]
            End of block loop
        End of decoding loop
```

Changed to:

```
        Loop until a decompression terminating condition
            Build the decoding table
            CurrentPosition = 256              // start at the end of the Huffman
    table
            NextBits = Read16Bits(InputBuffer + CurrentPosition)
            CurrentPosition += 2
            NextBits <<= 16
            NextBits |= Read16Bits(InputBuffer + CurrentPosition)
            CurrentPosition += 2
            ExtraBitCount = 16
            BlockEnd = OutputPosition + 65536
            Loop until a block terminating condition
                If the OutputPosition >= BlockEnd then terminate block processing
                Next15Bits = NextBits >> (32 - 15)
                HuffmanSymbol = DecodingTable[Next15Bits]
                HuffmanSymbolBitLength = the bit length of HuffmanSymbol, from
    the table in
                the input buffer
                NextBits <<= HuffmanSymbolBitLength
                ExtraBitCount -= HuffmanSymbolBitLength
                If ExtraBitCount < 0
                    NextBits |= Read16Bits(InputBuffer + CurrentPosition) << (-
    ExtraBitCount)
                    ExtraBitCount += 16
```

Errata Published*	Description
	CurrentPosition += 2 If HuffmanSymbol < 256 Output the byte value HuffmanSymbol to the output stream. Else If HuffmanSymbol == 256 and the entire input buffer has been read and the expected decompressed size has been written to the output buffer Decompression is complete. Return with success. Else HuffmanSymbol = HuffmanSymbol - 256 MatchLength = HuffmanSymbol mod 16 MatchOffsetBitLength = HuffmanSymbol / 16 If MatchLength == 15 MatchLength = ReadByte(InputBuffer + CurrentPosition) CurrentPosition += 1 If MatchLength == 255 MatchLength = Read16Bits(InputBuffer + CurrentPosition) CurrentPosition += 2 If MatchLength < 15 The compressed data is invalid. Return error. MatchLength = MatchLength - 15 MatchLength = MatchLength + 15 MatchLength = MatchLength + 3 MatchOffset = NextBits >> (32 – MatchOffsetBitLength) MatchOffset += (1 << MatchOffsetBitLength) NextBits <<= MatchOffsetBitLength ExtraBitCount -= MatchOffsetBitLength If ExtraBitCount < 0 Read the next 2 bytes the same as the preceding (ExtraBitCount < 0) case For i = 0 to MatchLength - 1 Output OutputBuffer[CurrentOutputPosition – MatchOffset + i] End of block loop End of decoding loop
2020/06/08	In Section 2.2.4 Processing, we clarified when and how implementations must check for the EOF condition during decompression. We modified the pseudocode and added explanatory text. Changed from: The compression stream is designed to be read in (mostly) 16-bit chunks, with a 32-bit register maintaining at least the next 16 bits of input. This strategy allows the code to seamlessly handle the bytes for long match lengths, which would otherwise be awkward. The following pseudocode demonstrates this method. Loop until a decompression terminating condition Build the decoding table ... Changed to: The compression stream is designed to be read in (mostly) 16-bit chunks, with a 32-bit register maintaining at least the next 16 bits of input. This strategy allows the code to seamlessly handle the bytes for long match lengths, which would otherwise be awkward. The following pseudocode demonstrates this method.

Errata Published*	Description
	During the beginning of processing each block for decompression, an implementation MUST check for EOF. An implementation can do this by comparing the block size against the required space for a Huffman table ” if this condition is met and all output has been written, then processing stops and success is returned. Alternately, an implementation can explicitly examine the input buffer using the Huffman table from the previous block. Loop until a decompression terminating condition Check for EOF Build the decoding table ...
2020/04/27	In Section 2.2.4, Processing, we replaced CurrentOutputPosition with OutputPosition for simplicity and clarity of the pseudocode. Changed from: For i = 0 to MatchLength - 1 Output OutputBuffer[CurrentOutputPosition – MatchOffset + i] Changed to: For i = 0 to MatchLength - 1 Output OutputBuffer[OutputPosition – MatchOffset + i]
2020/04/27	In Section 2.2.4, Processing, we clarified the nesting and termination conditions of the loops in the pseudocode. Changed from: Loop until a block terminating condition If OutputPosition >= BlockEnd then terminate block processing Loop until a literal processing terminating condition Changed to: Loop until a block terminating condition Loop until a literal processing terminating condition If OutputPosition >= BlockEnd then terminate block processing
2020/04/27	In Section 2.2.4, Processing, we altered the pseudocode to advance the CurrentPosition by 256 rather than assigning a fixed value of 256. Changed from: CurrentPosition = 256 // start at the end of the Huffman table

Errata Published*	Description
	Changed to: CurrentPosition += 256 // start at the end of the Huffman table

*Date format: YYYY/MM/DD