HexDecoder

From Crypto++ Wiki

The **HexDecoder** converts base 16 encoded data to bytes. Since the **HexDecoder** inherits from BufferedTransformation, the filter can participate in pipelining. The partner encoder is a HexEncoder. The class documentation is located at HexDecoder Class Reference (http://www.cryptopp.com/docs/ref/class_hex_decoder.html) .

Contents

- 1 Construction
- 2 Parsing
- 3 Sample Programs
  - 3.1 Decoding a String (Non-Filter)
  - 3.2 Decoding a String (Filter)
  - 3.3 Attaching a BufferedTransformation
  - 3.4 Scripting and Strings
  - 3.5 Missing Data

Construction

```
HexDecoder (BufferedTransformation *attachment=NULL)
```

`attachment` is a BufferedTransformation, such as another filter or sink.

Parsing

A HexDecoder can parse many formats, including colon, comma, and whitespace delimited. The example strings bellow will all decode correctly using a HexDecoder.

```
string str1 = "0xFF, 0xEE, 0xDD, 0xCC, 0xBB, 0xAA, 0x99, 0x88, 0x77, 0x66, 0x55, 0x44, 0x33, 0x22, 0x11, 0x00";
string str2 = "0xFF 0xEE 0xDD 0xCC 0xBB 0xAA 0x99 0x88 0x77 0x66 0x55 0x44 0x33 0x22 0x11 0x00";
string str4 = "FFh EEh DDh CCh BBh AAh 99h 88h 77h 66h 55h 44h 33h 22h 11h 00h";
string str4 = "FF:EE:DD:CC:BB:AA:99:88:77:66:55:44:33:22:11:00";
string str5 = "FFEEDDCCBBAA99887766554433221100";
```

Sample Programs

Decoding a String (Non-Filter)

The following demonstrates decoding a string using `Put` and `Get`.

```
string encoded = "FFEEDDCCBBAA99887766554433221100";
string decoded;

HexDecoder decoder;

decoder.Put( (byte*)encoded.data(), encoded.size() );
decoder.MessageEnd();

word64 size = decoder.MaxRetrievable();
if(size && size <= SIZE_MAX)
{
    decoded.resize(size);
    decoder.Get((byte*)decoded.data(), decoded.size());
}
```

Running the program under GDB shows the binary string contained in *decoded*. \377 is octal for 0xFF, \365 is octal for 0xEE, etc.

```
Breakpoint 1, main (argc=1, argv=0x7fffffffe398) at cryptopp-test.cpp:44
(gdb) p encoded
$1 = {static npos = 18446744073709551615,
  _M_dataplus = {<std::allocator<char>> = {<__gnu_cxx::new_allocator<char>> = {<No data fields>}, <No data field
    _M_p = 0x60c078 "FFEEDDCCBBAA99887766554433221100"}}
(gdb) p decoded
$2 = {static npos = 18446744073709551615,
  _M_dataplus = {<std::allocator<char>> = {<__gnu_cxx::new_allocator<char>> = {<No data fields>}, <No data field
    _M_p = 0x60c868 "\377\356\335\252\231\210wfUD3\"\021"}}
```

Decoding a String (Filter)

Decoding a String (Non-Filter) performed a Put/Get sequence to transform the data. Crypto++ offers filters, which can simplify the process as shown below by taking advantage of Crypto++'s pipeline design.

```
string encoded = "FFEEDDCCBBAA99887766554433221100";
string decoded;

StringSource ss(encoded,
    new HexDecoder(
        new StringSink(decoded)
    ) // HexDecoder
); // StringSource
```

Attaching a BufferedTransformation

Sometimes its advantageous to attach (or change an attached) BufferedTransformation on the fly. The code below attaches a StringSink at runtime.

```
string encoded = "FFEEDDCCBBAA99887766554433221100";
string decoded;

HexDecoder decoder;

decoder.Attach( new StringSink( decoded ) );
decoder.Put( (byte*)encoded.data(), encoded.size() );
decoder.MessageEnd();

// decoded holds the converted data as a binary string
```

Scripting and Strings

On occasion, the mailing list will receive questions on cross-validation. For example, see *AES CTR Chiper. Different output between PHP-mcrypt and Crypto++ (http://groups.google.com/group/cryptopp-users/browse_thread/thread/73765be8f6334bbb)* . In the question, PHP-mcrypt strings are used as follows:

```
$key = "1234567890123456789012345678901234567890123456789012345678901234";
$key = pack("H".strlen($key), $key);
$iv = "1111111111222222222233333333334444444444555555555566666666667777";
$iv = pack("H".strlen($iv), $iv);
```

One of the easiest ways to avoid typos is via Copy/Paste and a **HexDecoder**:

```
string encodedKey = "1234567890123456789012345678901234567890123456789012345678901234";
string encodedIv = "1111111111222222222233333333334444444444555555555566666666667777";
string key, iv;

StringSource ssk(encodedKey, true /*pumpAll*/,
    new HexDecoder(
        new StringSink(key)
    ) // HexDecoder
); // StringSource

StringSource ssv(encodedIv, true /*pumpAll*/,
    new HexDecoder(
        new StringSink(iv)
    ) // HexDecoder
); // StringSource
```

After running the above code, *key* and *iv* are hexadecimal (i.e., binary) strings rather than printable (i.e., ASCII) strings. Below, GDB displays the binary digits in octal, so 0x12 = \022, 0x34 = \064, 0x56 = V (printable), 0x78 = x (printable), and 0x90 = \220.

```
Breakpoint 1, main (argc=1, argv=0x7fffffffe398) at cryptopp-test.cpp:38
(gdb) p encodedKey
$1 = {static npos = 18446744073709551615,
  _M_dataplus = {<std::allocator<char>> = {<__gnu_cxx::new_allocator<char>> = {<No data fields>}, <No data field
    _M_p = 0x614078 "1234567890123456789012345678901234567890123456789012345678901234"}}
(gdb) p key
$2 = {static npos = 18446744073709551615,
  _M_dataplus = {<std::allocator<char>> = {<__gnu_cxx::new_allocator<char>> = {<No data fields>}, <No data field
    _M_p = 0x6142a8 "\022\064Vx\220\022\064Vx\220\022\064Vx\220\022\064Vx\220\022\064Vx\220\022\064Vx\220\022\06
```

Finally, the strings *key* and *iv* can be used with Encryption or Decryption objects as follows.

```
CTR_Mode< AES >::Encryption enc;
enc.SetKeyWithIV(key.data(), key.size(), iv.data());
```

Missing Data

Its not uncommon to send data through a pipeline and then have nothing in the sink. This is usually due to the compiler matching the wrong function. For example:

```
string source = "ABCD...WXYZ", destination;
```

```
StringSink ss(source,
    new HexEncoder(
        new StringSink(destination)
    ) // HexEncoder
); // StringSink
```

After the above code executes, `destination` will likely be empty because the compiler coerces the `HexEncoder` to a `bool` (the `pumpAll` parameter), which leaves the `StringSource`'s attached transformation `NULL`. The compiler will do so without warning, even with `-Wall -Wextra -Wconversion`. To resolve the issue, explicitly specify the **`pumpAll`** parameter:

```
string source = "ABCD...WXYZ", destination;
StringSink ss(source, true /*pumpAll*/
    new HexEncoder(
        new StringSink(destination)
    ) // HexEncoder
); // StringSink
```