C#.NET Application call:

```
-----------------------------------------------------------------------------------------------------------------
public static Mechanism encMech = null;
byte[] iv = Convert.FromBase64String("AAECAwQFBgcICQoLDA0ODw==");

encMech = new Mechanism(CKM.CKM_AES_CBC_PAD, iv);

// ipstream and opstream both are of type MemoryStream // fpk ==> AES256 key object handle
```
***session.Decrypt(encMech, fpk, ipstream, opstream, 512);***

```
-----------------------------------------------------------------------------------------------------------------
```

Inside Pkcs11Interop's HighlevelAPI 41:

```
-----------------------------------------------------------------------------------------------------------------

C_DecryptInit ---> Mechanism : CKM_AES_CBC_PAD , IV : AAECAwQFBgcICQoLDA0ODw==

C_DecryptUpdate
----------------
Input :
-------
f26e7f66aab6fb86e347b895a6fd27af5d8b27e34a63f053a6529fb4d95debacb2f04b8488b0ffd090cea2
1a17c05c69bda0eca6372980296673e91a8753b62d964d8fc6c6e423908cb98bd4ad4bf2703995b090b
0aef8f137ae4c6d1b740a5f7949128ff28544e1502a27b7c034d714082edff2329f6789d8e77dedef60c55
74e8039629208eb86979a166571d81cf6fbea4c37a4110625fb23a26326641da1000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000

decrypted part output (part)
-----------------------
3c3f786d6c2076657273696f6e3d22312e302220656e636f64696e673d225554462d38223f3e0d0a3c7
26f6f743e0d0a20203c706572736f6e3e0d0a202020203c6e616d653e4261746d616e3c2f6e616d653e0
d0a202020203c6167653e32353c2f6167653e0d0a202020203c636974793e476f7468616d3c2f636974
793e0d0a20203c2f706572736f6e3e0d0a3c2f0000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000

C_DecryptFinal  call with null value passed in-order to fetch the buffer size
--------------
              ===> buffer size fetched 7 (in bytes)
C_DecryptFinal call to get the last part
---------------
              last part : 726f6f743e0d0a (hex)
-----------------------------------------------------------------------------------------------------------------
```

================================================================================
Decrypted combined output (Decrypt Update and Decrypt Final)
-------------------------------------------------------------------------------------------------------------------------

After **session.Decrypt** function call we get the data in opstream (Pkcs11Interop removes the trailing zeros internally but adds some Unintended Characters at the end - "25fb23a26326641da1" , this data "25fb23a26326641da1" was observed while debugging the outputstream )

```
3c3f786d6c2076657273696f6e3d22312e302220656e636f64696e673d225554462d38223f3e0d0a3c7
26f6f743e0d0a20203c706572736f6e3e0d0a202020203c6e616d653e4261746d616e3c2f6e616d653e0
d0a202020203c6167653e32353c2f6167653e0d0a202020203c636974793e476f7468616d3c2f636974
793e0d0a20203c2f706572736f6e3e0d0a3c2f726f6f743e0d0a25fb23a26326641da1
```

Checking Decrypted combined output in ASCII format
-------------------------------------------------------------------------------------------------------------------------

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <person>
        <name>Batman</name>
        <age>25</age>
        <city>Gotham</city>
  </person>
</root>
%û#¢c&d  ¡
```
-------------------------------------------------------------------------------------------------------------------------
================================================================================

Now, we used below code for debugging Pkcs11Interop

```
//HighLevelAPI 41 source
------------------------
 public static string ByteArrayToString(byte[] ba)
{
       StringBuilder hex = new StringBuilder(ba.Length * 2);
       foreach (byte b in ba)
             hex.AppendFormat("{0:x2}", b);

       return hex.ToString();
}

public void Decrypt(Mechanism mechanism, ObjectHandle keyHandle, Stream
inputStream, Stream outputStream, int bufferLength)
{

       Stream outputStream1;

       if (this._disposed)
             throw new ObjectDisposedException(this.GetType().FullName);

       if (mechanism == null)
             throw new ArgumentNullException("mechanism");

       if (keyHandle == null)
             throw new ArgumentNullException("keyHandle");

       if (inputStream == null)
             throw new ArgumentNullException("inputStream");

       if (outputStream == null)
             throw new ArgumentNullException("outputStream");
```

```
        if (bufferLength < 1)
                throw new ArgumentException("Value has to be positive number",
"bufferLength");

        CK_MECHANISM ckMechanism = mechanism.CkMechanism;

        CKR rv = _p11.C_DecryptInit(_sessionId, ref ckMechanism,
keyHandle.ObjectId);
        if (rv != CKR.CKR_OK)
                throw new Pkcs11Exception("C_DecryptInit", rv);

        byte[] encryptedPart = new byte[bufferLength];
        byte[] part = new byte[bufferLength];
        uint partLen = Convert.ToUInt32(part.Length);

        int bytesRead = 0;
        while ((bytesRead = inputStream.Read(encryptedPart, 0,
encryptedPart.Length)) > 0)
        {
                partLen = Convert.ToUInt32(part.Length);
                Console.WriteLine("Input : " +
ByteArrayToString(encryptedPart));              //
                Console.WriteLine("Input len : " + Convert.ToUInt32(bytesRead));
                // 160 (in bytes)
                rv = _p11.C_DecryptUpdate(_sessionId, encryptedPart,
Convert.ToUInt32(bytesRead), part, ref partLen);
                if (rv != CKR.CKR_OK)
                        throw new Pkcs11Exception("C_DecryptUpdate", rv);
                Console.WriteLine("part : " + ByteArrayToString(part)); //
                Console.WriteLine("part len : " + partLen);
        // 144 (in bytes)
                outputStream.Write(part, 0, Convert.ToInt32(partLen));
        }

        byte[] lastPart = null;
        uint lastPartLen = 0;
        rv = _p11.C_DecryptFinal(_sessionId, null, ref lastPartLen);
        if (rv != CKR.CKR_OK)
                throw new Pkcs11Exception("C_DecryptFinal", rv);

        Console.WriteLine("buffer size : " + Convert.ToInt32(lastPartLen));

        lastPart = new byte[lastPartLen];

        rv = _p11.C_DecryptFinal(_sessionId, lastPart, ref lastPartLen);
        if (rv != CKR.CKR_OK)
                throw new Pkcs11Exception("C_DecryptFinal", rv);

        if (lastPartLen > 0)
        {
                Console.WriteLine("last part : " + ByteArrayToString(lastPart));
        // 726f6f743e0d0a (hex)
                Console.WriteLine("last part len : " +lastPartLen);
                        // 7 (in bytes)
                outputStream.Write(lastPart, 0, Convert.ToInt32(lastPartLen));
        }

        MemoryStream memoryStream;
        memoryStream = new MemoryStream();
        outputStream.CopyTo(memoryStream);

        Console.WriteLine("memory stream :
"+ByteArrayToString(memoryStream.ToArray()));//===> 25fb23a26326641da1 (hex)
//This is the additional data which we are gettign as a part of original
decrypted data.
        Console.WriteLine("memory stream size : " + memoryStream.Length);
                                        // 9 (in bytes)
}
```