Developing support for additional encryption libraries

for
kbmMW v. 0.93+

One of the strengths of kbmMW is the open API which can be used to extend kbmMW with new functionality by simply adding a new component or class and referring to it.

In the area of cryptology many different algorithms exist in many different implementations. kbmMW comes bundled with support for some of the implementations and algorithms. But if you need support for another implementation or algorithm, kbmMW do not hinder you doing that. And that's what this whitepaper is about… how to extend kbmMW with new encryption/decryption algorithms/implementations.

How does it work?

kbmMW can automatically utilize an encryption/decryption component to handle encryption/decryption of the stream of data that are send via a transport to the receiving end which can be a server or a client. The transport takes care of any additional wrapping of the encrypted data and unwrapping before decryption if that is needed (for example in the case of the HTTP stream format where the datastream have to comply with the requirements of the Hypertext Transport Protocol.

Thus it's the transport logic that choose when to ask the encryption/decryption component to do its work. This way additional variants of encryption/decryption components can be added and used seamlessly.

Creating a custom encryption/decryption component

All encryption/decryption components inherits from the class TkbmMWCustomCrypt which basically only contains two virtual procedures that needs to be overridden.

The class also supports two events (OnEncrypt and OnDecrypt) which optionally can be published and thus allows developers to hook into the encryption/decryption operation potentially doing their own encryption/decryption.

This what the bundled TkbmMWEventCrypt component do. It simply publishes those events and leave it up to developers to put their own encryption/decryption logic into the events. This is an ok adhoc solution, but not advisable since it do not encourage to reuse of code.

Thus its better to subclass TkbmMWCustomCrypt and put the encryption/decryption code in the overridden versions of the two virtual methods Encrypt and Decrypt.

As an example, here is the code for a _very_ simple encryption component which I will not recommend you to use for anything but as an example:

```
Interface

  TkbmMWMyCrypt = class(TkbmMWCustomCrypt)
  public
     procedure Encrypt(Caller:TObject; FromStream,ToStream:TStream); override;
     procedure Decrypt(Caller:TObject; FromStream,ToStream:TStream); override;
  end;

implementation

  procedure TkbmMWMyCrypt.Encrypt(Caller:TObject; FromStream,ToStream:TStream);
  var
    i:integer;
    b:byte;
  begin
     // Do the encryption here by reading data from FromStream, encrypting it
     // and writing it to ToStream.
     For i:=0 to FromStream.Size-1 do
     Begin
        FromStream.Read(b,sizeof(byte));  // Read one byte from the FromStream.
        b:=b XOR $FF;                     // So simple encryption.
        ToStream.Write(b,sizeof(byte));   // Store the encrypted byte in the ToStream.
     End;
     ToStream.Seek(0,soFromBeginning);    // Important to remember!
  end;

  procedure TkbmMWMyCrypt.Decrypt(Caller:TObject; FromStream,ToStream:TStream);
  var
    i:integer;
    b:byte;
  begin
     // Do the decryption here by reading data from FromStream, decrypting it
     // and writing it to ToStream.
     For i:=0 to FromStream.Size-1 do
     Begin
        FromStream.Read(b,sizeof(byte));  // Read one encrypted byte from the FromStream.
        b:=b XOR $FF;                     // So simple decryption.
        ToStream.Write(b,sizeof(byte));   // Store the decrypted byte in the ToStream.
     End;
     ToStream.Seek(0,soFromBeginning);    // Important to remember!
  end;
```

As mentioned this is a very simple example.

Please notice that this example do not publish the OnEncrypt and OnDecrypt events since I do not want the developer to be able to interfere with the operation of this component.

If you do want the developer to be able to access those events, you have to publish them and if you override the Encrypt and Decrypt methods remember to call inherited where the event logic is placed.

Eg.

```
  TkbmMWMyCrypt = class(TkbmMWCustomCrypt)
  public
     procedure Encrypt(Caller:TObject; FromStream,ToStream:TStream); override;
     procedure Decrypt(Caller:TObject; FromStream,ToStream:TStream); override;
  published
     property OnEncrypt;
     property OnDecrypt;
  end;

implementation

  procedure TkbmMWMyCrypt.Encrypt(Caller:TObject; FromStream,ToStream:TStream);
  var
    i:integer;
    b:byte;
  begin
     // Do the encryption here by reading data from FromStream, encrypting it
     // and writing it to ToStream.
     For i:=0 to FromStream.Size-1 do
     Begin
        FromStream.Read(b,sizeof(byte));  // Read one byte from the FromStream.
        b:=b XOR $FF;                     // So simple encryption.
        ToStream.Write(b,sizeof(byte));   // Store the encrypted byte in the ToStream.
     End;
     ToStream.Seek(0,soFromBeginning);    // Important to remember!
     Inherited Encrypt(Caller,FromStream,ToStream);
  end;

  procedure TkbmMWMyCrypt.Decrypt(Caller:TObject; FromStream,ToStream:TStream);
  var
    i:integer;
    b:byte;
  begin
     // Do the decryption here by reading data from FromStream, decrypting it
     // and writing it to ToStream.
     For i:=0 to FromStream.Size-1 do
     Begin
        FromStream.Read(b,sizeof(byte));  // Read one encrypted byte from the FromStream.
        b:=b XOR $FF;                     // So simple decryption.
        ToStream.Write(b,sizeof(byte));   // Store the decrypted byte in the ToStream.
     End;
     ToStream.Seek(0,soFromBeginning);    // Important to remember!
     Inherited Decrypt(Caller,FromStream,ToStream);
  end;
```

As a minimum either the code in the events or the code in the methods must move the data in some form from the FromStream to the ToStream for anything to work.

All whats needed now is to register the new component via the RegisterComponents procedure. Eg.

```
procedure Register;
begin
     RegisterComponents('kbmMW General',          [TkbmMWDCP2Crypt]);
end;
```

What to remember?

Since you are likely using a 3rdparty library of some kind to do the encryption/decryption, it most probably require you to initialize something before doing the cncryption/decryption.

Since kbmMW operates highly threaded, all code and libraries used by the crypt component must be thread safe. If its not its not recommended to use it even if you make it threadsafe via critical sections etc. simply because it will have a high impact on performance.

Also remember always to do all the steps like initialize/creation of an encryption object, preparation of a cipher, encryption, burning of the cipher and finally deinitialization/destruction of the encryption object all in the Encrypt method itself. The same with the Decrypt operation.

If exceptions can occur, remember always to free any allocated resource in a try/final section. If not you will have a potential memory leak which very fast becomes large.

That concludes the whitepaper of how to create support for 3rdparty encryption engines.

Best regards

Kim Madsen
kbm@components4developers.com