Security and Encryption using kbmMW Components.

This whitepaper provides the basics of implementing several concepts of both user authentication and the security components of kbmMW. **Section I** covers user authentication concepts and **Section II** data security concepts.

Section I Basic User Authentication

The kbmMW Client has 3 properties that can be used for simple user authentication or you can create a custom service to provide a higher level of security.

Both the kbmMWClient and kbmMWFileClient controls have 3 properties that can be passed to the server prior to processing any data stream.
Username
Password
Token

These are independent of your OS security system and you must provide the information for these properties. The “Token” property can be use for a variety of information that you want to pass and compare at the server and in the example below I will provide a few examples.

First open your server application and on the kbmMWServer set the EarlyAuthentication property to true.

Next in the kbmMWServer.OnAuthenticate event enter

```
If (ClientIdent.UserName = ‘me’) and (ClientIdent.Password = ‘mypassword’) then
  // set permissions
   perm := [mwapRead,mwapWrite,mwapDelete,mwapExecute];
```

That’s it. The kbmMWClient control must pass the UserName and Password to access the system. Obviously you would want to validate the UserName and Password against a table with the users information in it.

The token property can be used for most anything, again you can do a comparison of the string in the OnAuthenticate event for what ever purpose you desire, for example you may want to have in the client app a Unique ID that must match a server side Unique ID telling the server which services the user is allowed to use or to simply make your client app unique to the server app, for example if you set your server ID to 1234 then the client must pass 1234 to gain access.

```
If (ClientIdent.Token = ‘1234’)  then
  // set permissions
  perm := [mwapRead,mwapWrite,mwapDelete,mwapExecute];
```

You can also control access to each individual query or custom service.

Custom Service Authentication
Using a custom service to validate user provides a higher level of security if used in conjunction with an encrypted stream (covered in Section II).

First you must create a custom service then add the function to process the request.

```
{ Protected declarations }
```

```
function ProcessRequest(const Func:string; const
ClientIdent:TkbmMWClientIdentity;        const Args:array of Variant):Variant;
override;
function ValidateUser(ClientIdent:TkbmMWClientIdentity; const Args:array of
Variant):Variant;


{Process Request}
function TACustomService.ProcessRequest(const Func:string; const
ClientIdent:TkbmMWClientIdentity; const Args:array of Variant):Variant;
begin
      if AFunc = 'VALIDATE_USER' then
         Result:=ValidateUser(ClientIdent,Args)
end;

{validate user}
function TACustomService.CVEFileValidate(ClientIdent:TkbmMWClientIdentity;
const Args:array of Variant):Variant;
var
 Username : string;
 Password : string;
begin
       UserName := Args[0];
       Password := Args [1];
       <match username and password against your database>

       Result := true/false
end;

Now on the client application you simply need to call the custom service to
validate the user.

function TForm1.ValidateUser(UserName,UserPassword:string): boolean;
var
v : boolean;
begin
    v:=kbmMWClient.Request('YOUR_SERVICE','','VALIDATE_USER',
                           [Username,UserPassword]);


    result := v;
end;
```

These examples are the basics and can be extended anyway you feel comfortable to validate a user.

Section II Data Encryption

Data encryption in kbmMW can be achieved by simply adding a kbmMWEventCrypt component to your applications and attaching it to the Client or Server Transport that want to use encrypted data. IMPORTANT both server and client must be setup with the same encryption scheme for encryption and decryption to take place.

Once the controls are added to the project you can use your own encryption methods via the OnDecrypt and OnEncrypt Events or you may choose to add the freeware DCCrypt controls to your IDE and then you simply drop a TkbmMWDCP2Crypt control on the server and client application and set the encryption methods you want to use. These controls are available on the components for developers web site under kbmMW downloads (DCPCrypt22.zip.).

Encypting data will effect performance so you should experiment and research which algorythms work best with your information. The loss of performance is not your kbmMW system. Usually the best performace is achived with 128 bit encryption.

Using Third Party Encryption Components
To use a third party component the requirement is that it encrypts/decrypts streams.

{Examples using an old TSM encryption component}

```
procedure TService.MWEventDecrypt(Sender: TObject; FromStream,
  ToStream: TStream);
var
edMethod   : TTwofish;
begin
  edMethod   := TTwofish.Create(nil);
   try
     edMethod .CipherMode := CBC ;
     edMethod.InitialiseString(FKey);
     edMethod.LoadIVString(FKey);
     edMethod.DecStream(FromStream,ToStream);
     edMethod.Burn;
   finally
    edMethod.Free;
   end;

end;


procedure TService.MWEventEncrypt(Sender: TObject; FromStream,
  ToStream: TStream);
var
edMethod   : TTwofish;
begin
  edMethod   := TTwofish.Create(nil);
   try
     edMethod.CipherMode := CBC ;
     edMethod.InitialiseString(FKey);
     edMethod.LoadIVString(FKey);
     edMethod.EncStream(FromStream,ToStream);
     edMethod.Burn;
   finally
    edMethod.Free;
   end;
end;
```

Other ways to protect your information.

Another excellent product that allows the use of certificates is available at
http://www.streamsec.com/ .

If you are using .net or ISAPI then SSL is another option for consideration.
NOTE: Please remember if you are using kbmMW in an ActiveX control, Activeform or a java app then SSL will not apply to the kbmMW connection of the component because it is connecting via another stream than the one used for the browser. You must handle the multiple connections individually in most cases.

Happy Ciphering