Firebird and IBExpert White Paper

DBEncryption Plugin Delphi demo Firebird 3 and Firebird 4

Fikret Hasovic, January 2022

The second in this series of Firebird DBEncryption documents shows you how to create a Delphi example and pass encryption keys.

With Delphi and FireDAC I have had to change approach a little, since I encountered some issues with passing encryption keys to the Firebird library. Finally I created the following example, how to load Firebird Embedded and pass the encryption keys, so that you can open a connection to your Firebird database. The same code will work with Firebird's SuperServer.

Simple Delphi example

To use the IBExpert Encryption Plugin you will need to start with the following two functions, and declare them:

```
  function
fbcrypt_key(pszKeyName:Pointer;pKeyValue:Pointer;iKeyLength:Cardinal) :
integer; cdecl; external 'fbcrypt.dll';

  function fbcrypt_callback(provider:Pointer) : integer; cdecl; external
'fbcrypt.dll';
```

In this case you will need to distribute the Firebird embedded library (or Firebird client only) together with the Delphi exe, since it did not work when trying to load the engine or client from a sub-folder.

I declared the following, since the encryption key is 32 bytes in length:

```
  type
  TMyCryptKeyValue = array [0..31] of byte;
  PMyCryptKeyValue = ^TMyCryptKeyValue;

  TMyDBCryptKey = record
    Name   :AnsiString;
    pValue :PMyCryptKeyValue;
  end;

  TMyCryptKeysArray = array of TMyDBCryptKey;
```

I also created these procedures:

```
  procedure ApplyEncryptionKey(MyEncryptionKeyName: ANSIString);
```

```
  procedure AllowDBAccess(MyDBKeysArray:TMyCryptKeysArray);
```

The implementation part is, for example:

```
  procedure TDataModule1.AllowDBAccess(MyDBKeysArray: TMyCryptKeysArray);
var i:Integer;
begin
  for i:=0 to Length(MyDBKeysArray)-1 do begin
    if (fbcrypt_key(PAnsiChar(MyDBKeysArray[i].Name),
MyDBKeysArray[i].pValue, sizeof(MyDBKeysArray[i].pValue^)) < 0) then
begin
      raise Exception.Create('fbcrypt_key failed');
    end;
  end;
  if (fbcrypt_callback(nil) < 0) then begin
    raise Exception.Create('fbcrypt_callback');
  end;
end;
```

And:

```
  procedure TDataModule1.ApplyEncryptionKey(MyEncryptionKeyName:
ANSIString);
var
  strKeyValue   :String;
  strByteValues :TStrings;
  tmpByteArray  :TBytes;
  i, v :Integer;
begin
  ActiveKeyName := MyEncryptionKeyName;
  strKeyValue:= DataModule1.AKeyData;
  if (Length(strKeyValue)=32) then begin
    //Key value as text
    tmpByteArray := TEncoding.ANSI.GetBytes(strKeyValue);
    if (Length(tmpByteArray)<>32) then begin
      abort;
    end;
    for i:=0 to 31 do begin
      ActiveKeyValue[i]:=tmpByteArray[i];
    end;
  end else begin
    //Key values as array of hex codes...
    strByteValues := TStringList.Create;
```

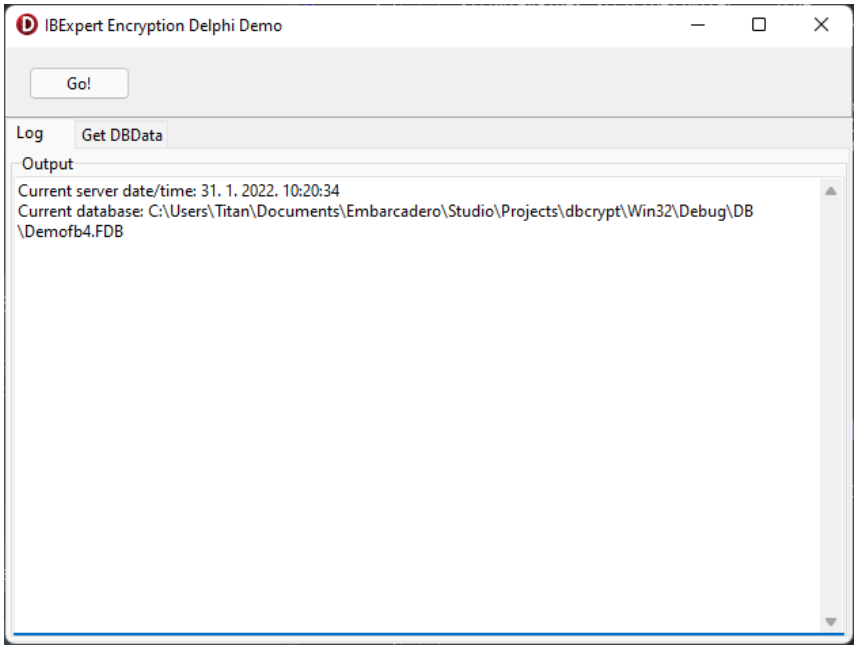
```
    try
      strByteValues.Clear;
      strByteValues.Delimiter       := ',';
      strByteValues.StrictDelimiter := True;
      strByteValues.DelimitedText   := Trim(strKeyValue);
      if (strByteValues.Count<>32) then begin
        abort;
      end;
      for i:=0 to strByteValues.Count-1 do begin
        strKeyValue := UpperCase(Trim(strByteValues.Strings[i]));
        if ( not TryStrToInt(strKeyValue, v) ) then begin
          abort;
        end;
        ActiveKeyValue[i]:=v;
      end;
    finally
      FreeAndNil(strByteValues);
    end;
  end;
end;
```

It is important to call the `OnBeforeConnect` event of your FireDAC Connection component, and this method will work on both dynamically-created or dropped components in your `DatModule`:

```
procedure TDataModule1.DBMainBeforeConnect(Sender: TObject);
begin
  try
    try
      SetLength(KeysArray, 1);
      KeysArray[0].Name   := ActiveKeyName;
      KeysArray[0].pValue := @ActiveKeyValue;
      //Some other keys

      AllowDBAccess(KeysArray);
    except
      on E:Exception do begin
        Raise
      end
    end
  finally
  end;
end;
```

The result of this Delphi demo connecting to an encrypted database can be seen in the following screenshots:



And displaying data from the sample table: