



Firebird and IBExpert White Paper

DBEncryption Plugin for Firebird 3.0 and 4.0

Fikret Hasovic, January 2022

IBExpert has developed an encryption plugin for Firebird 3.0 and Firebird 4.0. The Firebird 3.0 plugin is currently available for Windows (32/64 bit) and Linux (32/64 bit). The IBExpert Developer Studio full customer version includes the 32-bit embedded version, free to use in embedded mode. Server versions (32 bit and x64) require the IBExpert Server Tools.

The Firebird 4.0 plugin is currently available for Windows (32/64 bit), Linux is Work in Progress.

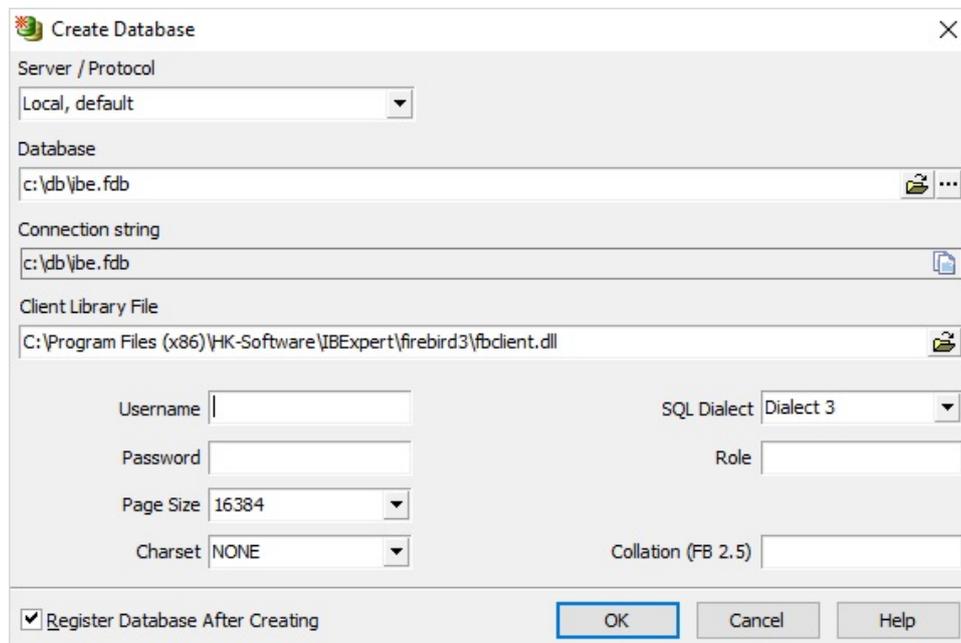
Important: each program, which needs to access an encrypted server, has to recognize and know the encryption, regardless of whether an embedded or server version.

Installation

1. Install the latest IBExpert customer version, it includes all files and configs you need. The first run of IBExpert should be done using Admin rights, so that it can generate the correct license key for using *dbcrypt* plugin.

Encrypt database

2. Create a database using *Local*, default as *Server / protocol* to use the included embedded version.





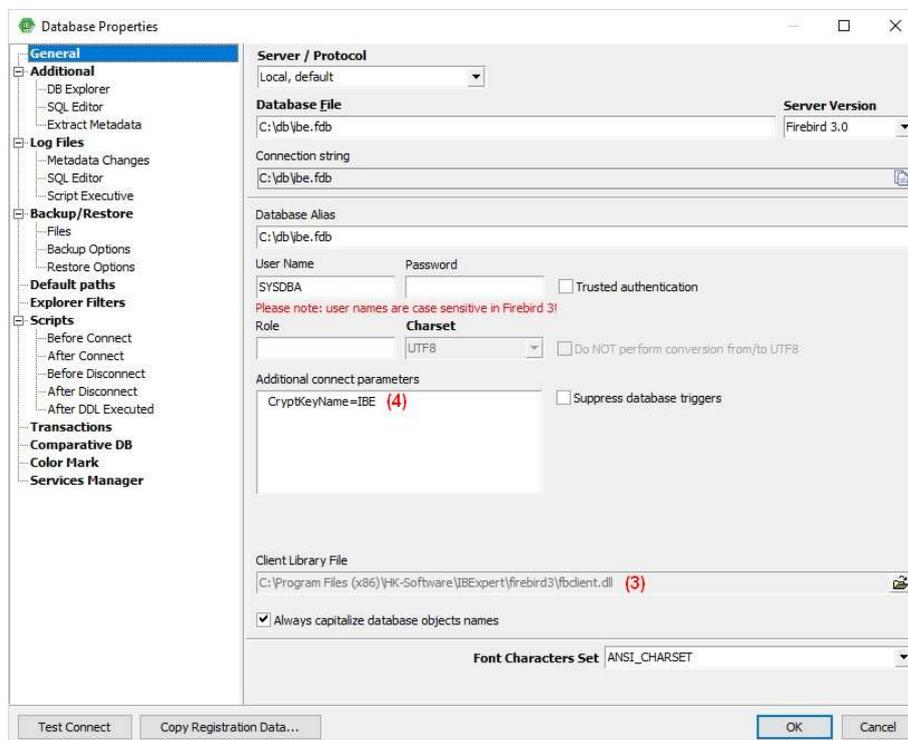
3. If you want to use it with Firebird 3.0:

Specify C:\Program Files (x86)\HK-Software\IBExpert\firebird3\fbclient.dll as the client library in the *IBExpert Database Registration*.

If you want to use it with Firebird 4.0:

Specify C:\Program Files (x86)\HK-Software\IBExpert\firebird4\fbclient.dll as the client library in the *IBExpert Database Registration*.

4. In the Database Registration's *Additional* connect parameters add this key: CryptKeyName=IBE



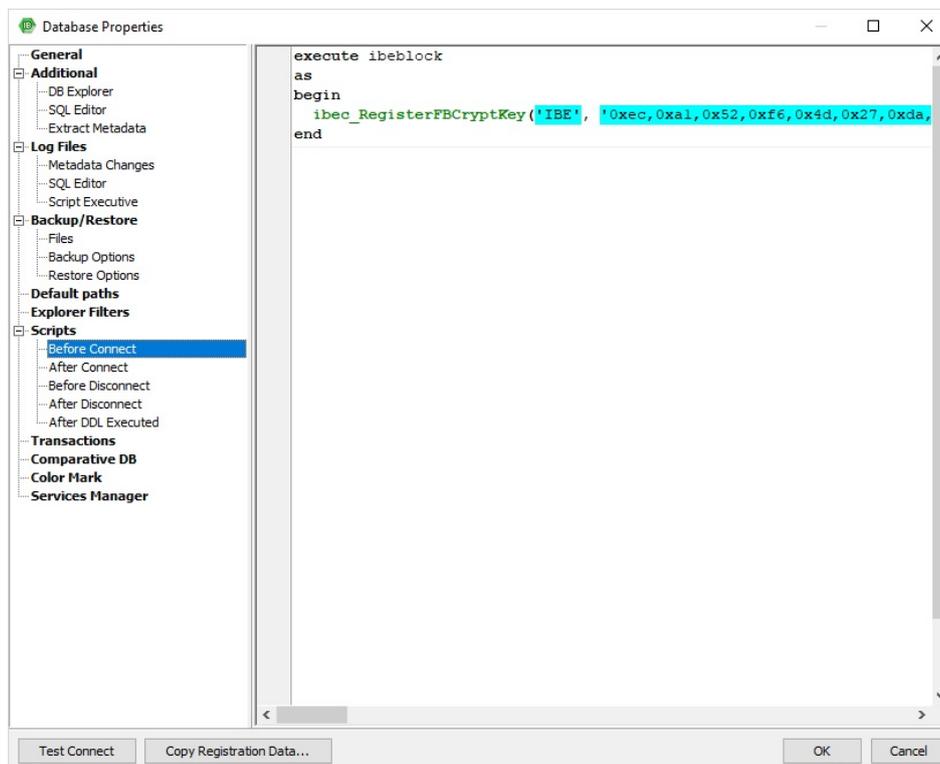
5. Still in the IBExpert Database Registration under *Scripts / Before Connect* add the following:

```
execute ibeblock
as
begin
    ibec_RegisterFBCryptKey('IBE',
```

```
'0xec,0xa1,0x52,0xf6,0x4d,0x27,0xda,0x93,0x53,0xe5,0x48,0x86,0xb9,0x7d,0x
e2,0x8f,
```



```
0x3b,0xfa,0xb7,0x91,0x22,0x5b,0x59,0x15,0x82,0x35,0xf5,0x30,0x1f,0x04,0xd  
c,0x75,', ' ');  
end
```



6. Connect to the database and encrypt it with:

```
ALTER DATABASE ENCRYPT WITH "DbCrypt" KEY IBE
```

7. Now remove CryptKeyName=IBE from the Database Registration.

And you are done.

Create encryption key

You can use the supplied *aesKeyGen.exe* to create the correct keys for database encryption.

Also take into account this:

KeyHolder.conf - when placed into server's plugins directory, this file works as a 'developer mode' switch enabling use of any client utility to work with encrypted databases. It must contain all known keys in the form "Key=Value", where the value's format is a sequence of bytes in C-compiler readable form. The current sample has the same keys as the sample application.



aesKeyGen.exe – a trivial utility performing a call to a random numbers generator and printing the result in a format compatible with **KeyHolder.conf**

rsaKeyGen.exe - this utility is needed if you want to build set of plugin components with unique pairs of RSA keys used to pass AES keys from the client to the **dbcrypt** plugin. Run:

```
rsaKeyGen >keysA2H.h rsaKeyGen >keysH2P.h
```

and copy the resulting files into the **crypt/db/lib** directory. This ensures that nobody except yourself has legal access to private keys in those pairs.

sample.exe is an example of **fbcrypt** API use.

Check database encryption

There are two different ways to check if the database is successfully encrypted. You can invoke *isql*:

```
isql -user SYSDBA -password masterkey
Use CONNECT or CREATE DATABASE to specify a database
SQL> connect C:\db\ibe.fdb;
Statement failed, SQLSTATE = HY000
Key not set
SQL>
```

Also, you can use *gstat* to check:

```
gstat -e c:\db\ibe.fdb
```

```
Database "C:\DB\IBE.FDB"
Gstat execution time Wed Jan 26 14:54:09 2022
```

Database header page information:

```
Flags 0
Generation 12605
System Change Number 0
Page size 16384
ODS version 12.0
Oldest transaction 7477
Oldest active 7937
Oldest snapshot 7937
Next transaction 7937
Sequence number 0
Next attachment ID 10980
Implementation HW=Intel/i386 little-endian OS=Windows
```

CC=MSVC



IBEXPERT WHITE PAPER



```
Shadow count          0
Page buffers          0
Next header page      0
Database dialect      3
Creation date         Mar 27, 2018 11:25:07
Attributes            force write, encrypted, plugin DbCrypt
```

Variable header data:

```
Crypt checksum: sHv0fE/Tfw9DKwDLyYIQ0qQ/hkk=
Key hash:       ask88tfWbinvC6b1JvS9Mfuh47c=
Encryption key name: IBE
*END*
```

Gstat completion time Wed Jan 26 14:54:09 2022

However, *gstat* output is a bit different in Firebird 4:

Database header page information:

```
Flags                0
Generation           375
System Change Number 0
Page size            16384
ODS version          13.0
Oldest transaction   42
Oldest active        336
Oldest snapshot      336
Next transaction     336
Sequence number      0
Next attachment ID   102
Implementation       HW=Intel/i386 little-endian OS=Windows
```

CC=MSVC

```
Shadow count          0
Page buffers          0
Next header page      0
Database dialect      3
Creation date         Jan 25, 2022 10:14:24
Attributes            force write, encrypted, plugin DbCrypt
```

Variable header data:

```
Crypt checksum: sHv0fE/Tfw9DKwDLyYIQ0qQ/hkk=
Key hash:       ask88tfWbinvC6b1JvS9Mfuh47c=
Encryption key name: IBE
Database GUID:  {117F0C3B-D795-46E7-B899-C47D4BDAA87A}
Sweep interval:  20000
*END*
```





Data pages: total 109, encrypted 109, non-encrypted 0
Index pages: total 66, encrypted 66, non-encrypted 0
Blob pages: total 0, encrypted 0, non-encrypted 0
Generator pages: total 1, encrypted 1, non-encrypted 0
Gstat completion time Wed Jan 26 14:51:31 2022

FPC and Lazarus example

To use the encryption plugin developed by IBExpert, you need to use the *cdecl* calling convention, as the following code shows:

```
Tfbcrypt_key = function (AName : PChar; AData : PChar; ALength : DWORD)
: integer; cdecl;
Tfbcrypt_callback = function (Provider : Pointer) : integer; cdecl;
```

To make the connection to the encrypted database, you need to use the following code:

```
function PassCryptKey(const FBCryptPath, AKeyName, AKeyData : string) :
integer;
var
  hFBCryptLib : THandle;
  fFBCryptKey : Tfbcrypt_key;
  fFBCryptCallback : Tfbcrypt_callback;
begin
  Result := 0;
  hFBCryptLib := LoadLibraryEx(PChar(FBCryptPath), 0,
LOAD_WITH_ALTERED_SEARCH_PATH);
  if (hFBCryptLib > HINSTANCE_ERROR) then
  begin
    Pointer(fFBCryptKey) := GetProcAddress(hFBCryptLib, 'fbcrypt_key');
    Pointer(fFBCryptCallback) := GetProcAddress(hFBCryptLib,
'fbcrypt_callback');
    if (@fFBCryptKey <> nil) and (@fFBCryptCallback <> nil) then
    begin
      Result := fFBCryptKey(PChar(AKeyName), PChar(AKeyData),
Length(AKeyData));
      if Result = 0 then
        Result := fFBCryptCallback(nil);
    end
  else
    Result := -2; // fbcrypt_key or fbcrypt_callback function not found
  end
end
```



```
Result := -1; // Cannot find/load fbcrypt.dll
end;
```

Also, you should declare a private procedure to load the Firebird client library and to pass the encryption key. For example:

```
procedure LoadLibrary(Filename: String);
var
  KeyData : ansistring;
begin
  with LibLoader do
  begin
    ConnectionType:='Firebird';
    LibraryName:=Filename;
    Enabled:=True;
  end;

  KeyData :=

chr($ec)+chr($a1)+chr($52)+chr($f6)+chr($4d)+chr($27)+chr($da)+chr($93)+c
hr($53)+chr($e5)+chr($48)+chr($86)+chr($b9)

+chr($7d)+chr($e2)+chr($8f)+chr($3b)+chr($fa)+chr($b7)+chr($91)+chr($22)+
chr($5b)+chr($59)+chr($15)+chr($82)+chr($35)
  +chr($f5)+chr($30)+chr($1f)+chr($04)+chr($dc)+chr($75);
  PassCryptKey(ExtractFilePath(Filename)+'fbcrypt.dll','IBE',KeyData);

end;
```

We have provided a sample encryption key name and key data value in the previous code.

You can see screenshots of Firebird 3 and Firebird 4 database usage written using Lazarus 2.2.0.

