Thoroughbred File Encryption
Supporting GDPR data at rest
FAQ and Notes

Thoroughbred's products include an interface to AES256 bit strong encryption provided by the OpenSSL Software Foundation. This method was selected because of the excellent reputation of the foundation and the minimal impact on performance.

How to use it?

An Auto-Expanding Direct file can be specified as encrypted by setting the sector number to -9. 3GL Developers can create an encrypted file using:

```
DIRECT "NEW_FILE", key_size, 0, rec_size, disk, -9
```

OPENworkshop™ developers will need to set the LINK-ENCRYPT-FLAG value to 'Y' in Link Maintenance and then have your application create a new file.

Once the file is created as encrypted, there is nothing else required in application code.

How does it work?

The AES256 bit encryption requires a key. Thoroughbred generates the key during startup using information from the product's license and installation procedure. The generated key is never stored physically. Once a file defined as encrypted Thoroughbred Basic™ will use the OpenSSL libraries to encrypt records when doing Writes and decrypt records when doing Reads.

Does Thoroughbred distribute the OpenSSL libraries?

Products from the OpenSSL Software Foundation are licensed. For Window products, a DLL is installed along with our product. For many UNIX and Linux installations, the library will already be installed or is available from the manufacturer. We have a few libraries available and may be able to compile a library if needed.

What products have it?

The encryption feature is built into the Thoroughbred run-time engine starting with version 8.8.2. Any Auto expanding direct file can be defined as encrypted. The Thoroughbred run-time engine is included in the following products: Basic/OPENworkshop Environments, Web Basic, XML DataServer, TWEB, TS Network DataServer, and TS ODBC DataServer.

What about multiple products on the same system?

Each Thoroughbred product generates a different encryption key. Files can only beread (decrypted) by the product that created them. So if another Thoroughbred product installed on the same system wants to access an encrypted file, you will need to specify which product's encryption key to use. This is done in the global IPLPRM file with PRM ENCRYPT='code', where the 'code' values are:

B for Basic
W for Web Basic
N for Network DataServer
O for ODBC DataServer

For most installations, we would recommend setting this to B.

PRM ENCRYPT=B

Once this is defined, other Thoroughbred products on the same system (like Network & ODBC DataServers) will be able to access the shared encrypted data.

How to determine if a file is encrypted?

The first 3 characters of the FID will be 'ENC' when a file is encrypted.

OPEN (ch) "encrypted_file";
FID$ = FID(ch)
IF FID$(1,3) = "ENC" THEN file is encrypted.

What happens if you try to encrypt a file on a system that is missing the OpenSSL libraries?

The interface looks for the library using the name libcrypto.so. If that fails, it looks for /usr/lib/basic/libcrypto.so. If that also fails, the encryption feature may not be available and trying to create an encrypted file will generate an ERR=-1 (feature not available). If you need to have the library in a particular path or use a specific name, you can configure it in the global IPLPRM by adding:

PRM ENCLIB='library_full_path_name'

How do you encrypt an existing file?

There is no magic converting an existing non-encrypted file to an encrypted file.

```
DIRECT "new_file",key_siz,0,rec_siz,disk,-9;
OPEN(CH1) "old_file"; OPEN(CH2) "new_file"; CLEAR ERC;
WHILE ERC=0;
     KEY$ = KEY(CH1,ERC=2);
     IF ERC=0
          READ RECORD(CH1) RECORD$;
          WRITE RECORD(CH2,KEY=KEY$) RECORD$
     FI;
WEND
```

How do you decrypt an existing file?

First keep in mind that Basic will do all the necessary encrypting/decrypting behind the scenes. Your applications will require no modifications in that regard. But it may be necessary to copy an encrypted file to another system. The other system will not be able to read the file unless you decrypt it first. You can use a similar program as above. Just create your new file with a sector number of 0 (rather than -9). Then read the records from the encrypted file and write them to the non-encrypted file.

Encryption is intended to be used in a secure environment. Thoroughbred's file level encryption protects your data at rest (storage). Your application is responsible for when the data is in use (what users have access to what files). And if you decide to move one of these files to another system keep in mind that data in motion also requires protection.

Miscellaneous notes:

The record size for encrypted data is always a multiple of 16. So your encrypted files will be larger on disk. If you create a file with a 100 byte record size, it will actually be 112 on disk. But all Basic file functions (like FID, XFD) will return the record size as you defined it. In this example it would be 100.

Only records are encrypted. Keys are not encrypted.

There is new Basic syntax that allows you to use the same Basic encryption key with strings.

A$ = "this is a test",
B$ = TFF(A$,"TE"), TE – Thoroughbred Encrypt
C$ = TFF(B$,"TD"), TD – Thoroughbred Decrypt

In this example, A$ and C$ should be the same value. If you try to decrypt data or a string that was not encrypted, C$ = TFF(A$,"TD"), you will not get an error.

When using DataSafeGuard™, the before and after records from encrypted files will be stored encrypted in the Journal. Assuming your DR/Playback server has the same license as the primary server, Playback will first determine the file is encrypted, it will then decrypt the record and write that data. It is up to Basic to re-encrypt the record before it's written to the data file.