Harlequin RIP

Font encryption options for the Harlequin RIP

Technical Note Hqn 012

June 2001

GLOBAL GRAPHICS®

1 Introduction

This technical note is a supplement to the *Harlequin RIP User Guide* describing the Font encryption options.

1.1 Contents

This document contains the following sections:

2 Font encryption options for the Harlequin RIP[1]

2.1 Introduction

The Harlequin RIP is capable of encrypting fonts using Harlequin's own proprietary Font Encryption algorithms. The RIP will successfully decrypt a font that has been encrypted with either of these two keys:

- the Harlequin OEM Number, which will be the same for all RIPs supplied by a particular Harlequin OEM
- the Harlequin RIP serial number, which is different for each RIP

As far as we are aware the Harlequin Encryption scheme has not been broken by anyone. There have been claims that the font encryption has been broken but these claims refer only to the DLD1 format itself, which was never intended to be encryption; after all the PostScript® language code used to perform the conversion is provided with the RIP.

---

[1] Please Note: The information in this document is intended for use by a Harlequin RIP OEM, and should not be passed on to other persons or organizations without prior permission from Harlequin.

2.2 The RIP serial number

For Harlequin's dongle-controlled products, both the OEM number and the serial number are acquired from the dongle itself and are thus secure.

For products that are controlled using the Unix-based license server, these two numbers are bound into the license in the same way as a host-id or machine-id, so attempts to change them will invalidate the license file, thus they are secure: if a copy of the RIP has successfully acquired a license, it must have acquired this particular pair of keys. The performance penalty for decryption is 30–40% on an I/O bound job. [The test example was a **`findfont`** of a large (16Mb) composite font, and does not show any characters, so all the time was I/O, on typical Unix (SPARC) platforms.] In practice the I/O time for finding fonts is small compared to the rendering and interpretation time, and font and character caching obviates much of this I/O, so this is not an important penalty.

Harlequin RIPs with GUI front ends report the serial number in the System Monitor when the RIP starts up (from version 3.2, revision 25 onwards).

2.3 Interaction with basic font format

The RIP is also capable of understanding a proprietary font format known as DLD1 format, to which Type 1 font files may be converted. DLD1 format fonts can consume as little as 2k of RIP VM per font, and offer improved performance over Type 1 and Type 4 fonts. Harlequin recommends conversion of all Type 1 fonts to DLD1 on installation in the RIP. Integrated Harlequin RIP products on the Mac and PC do this automatically during installation of Type 1 Latin fonts.

Examination of a Type 1 font is not a very complex process, and a competent PostScript language programmer could write code to process any Type 1 font in virtual memory and dump it to a file in some structured way. This applies whether the Type 1 font is held in an encrypted file in the RIP's file system or not. It is possible to discover everything about a Type 1 font by examining the PostScript language structures that define the font after it has been loaded using **`findfont`**. The possibility of examining the PostScript language structure of a font using a PostScript language program and of creating a PostScript language file in the RIP's file system which, if interpreted, would reproduce

the font's structure and thus reproduce the font itself applies to all Type 1 fonts on all PostScript language compatible interpreters (including those produced by Adobe), regardless of the installation techniques employed. Once the font is in the PostScript language world, a program can examine it and dump it to a file. The DLD1 format prevents this, by not loading the character definitions into the PostScript language world at all, so that such a dumping program would produce a copy of the font with all the important parts missing. Therefore Harlequin recommends DLD1 conversion as the first stage of the encryption process, because without it, Type 1 fonts are not strongly protected.

If a font is to be encrypted, and it is to be converted to DLD1 format, the conversion to DLD1 format *must* precede the encryption stage.

To convert Type 1 fonts to DLD1 format they may be installed into a RIP using the Install Font menu option on a RIP with a GUI front end, by downloading over AppleTalk using FireWorks, or through the **`-download`** option on a Unix command-line RIP. If this is not possible (e.g. for some leaf fonts of composite fonts) then you may convert the files using the following two steps:

- Copy the Type 1 fonts to the **`SW/fonts`** directory in the RIP.
- Interpret the file **`SW/Usr/Optimize Fonts (SW\USR\OPTIFONT.PS`** on an MS-DOS or NT RIP).

All these comments about Type 1 fonts apply equally to those Type 1 fonts that are base or "leaf" fonts of a composite (Type 0) font.

Type 3 font programs can also be encrypted, after minor modifications, but once loaded into virtual memory their entire structure can be examined trivially, regardless of any protection method attempted, so they are not strongly protected.

2.4 Options for encrypting Type 4 fonts

The RIP will successfully decrypt a font file only if it is accessed using the **`findfont`** operator or its equivalent **`selectfont`** and **`findresource`** operators. Thus such a protected font file cannot usefully be read even by PostScript language file operations; the decryption will not occur. These considerations apply equally to those Type 1 and Type 4 fonts that are base or leaf fonts of a composite (Type 0) font and those which are Latin, non-composite, fonts.

Type 1 fonts converted to DLD1 consist of a single file, which is executed using **`findfont`**, and which can therefore be encrypted and made completely secure.

The Type 4 font format is a less efficient way of solving the problem that DLD1 format addresses – reducing the amount of virtual memory that a font consumes. Type 4 fonts typically consist of a PostScript language program file that is executed using **`findfont`** as usual, plus one or more data files, and sometimes other common runtime PostScript language program files for procedures or data that are shared between many Type 4 fonts. This is particularly common in the case of Type 4 fonts that are leaf fonts of a composite font. The data files typically contain font outline data and offset tables to facilitate access to it. The **`BuildChar`** procedure of the Type 4 font uses PostScript language file operators to read a chunk of data from the font outline file and then execute it to build the character bitmap using a special operator called **`CCRun`**. The common runtime files are usually just **`run`** by the Type 4 font program.

These arrangements pose some problems for secure encryption of Type 4 fonts. The data files are accessed using general PostScript language procedures using file operators. If the data were encrypted, these routines would not decrypt it, and the character outline data would be nonsense. If these routines were to note that the files were encrypted and decrypt the data automatically, or a file mode or filter could be used to cause automatic decryption, then any encrypted font could be copied using a PostScript language program. Thus it is not feasible to encrypt the font outline data securely. The run time files are executed using an explicit **`run`** invocation, which does not decrypt an encrypted file, and which will stop immediately with a PostScript error if presented with an encrypted file to run. Thus these files cannot be securely encrypted.

It is worth mentioning that even if it were possible to encrypt all files associated with a Type 4 font securely, all the font data could be accessed by redefining PostScript language operators to dump the data in clear format to a different file, and by writing code to traverse the font dictionary structure and dump information to a file; the font could be recreated in clear format by a skilled PostScript language programmer.

There are several options available regarding encryption and protection strategies for Type 4 fonts that are to be bundled with a Harlequin RIP product.

- Obtain Type 1 versions, convert to DLD1 and encrypt.

  Font vendors usually use some standard font design tool which keeps the font data in its own internal format, and which can generate a choice of font formats on demand. It would not be technically difficult for most font vendors to produce Type 1 versions of fonts that they normally produce in Type 4. This is the only option that provides complete security, and so the font vendor may be happy to assist with such a request. Therefore Harlequin strongly recommends this option where it is possible.

- Encrypt only the parts of the Type 4 fonts that are run using **`findfont`**.

  This option requires the RIP vendor to inspect the fonts, work out which files are interpreted with **`findfont`** and encrypt those. This usually means the files in the fonts directory, but in some examples there are files in the fonts directory that are run directly. The files can be sorted into those for encryption and those that are `run` by judicious use of a string searching tool such as **`grep`**, and looking for **`findfont`** to determine files that can be encrypted, and **`run`** for those that cannot.

  Encrypting only parts of the font is a fairly secure subterfuge, but not completely secure because the font structure can always be unpicked using a PostScript language program enabling its re-creation by a skilled programmer.

- Obscure the outline data and encrypt all PostScript language program files.

  This option requires the RIP vendor to modify the font programs, in a very straightforward way, to use a special operator instead of **`run`** to execute the common and runtime files that normally cannot be encrypted. The special operator is in **`internaldict`**. This step increases the obfuscation. It also protects the procedures within the font, and so the font outline data may also be obscured by some means, being decoded in the **`BuildChar`** procedure within the now somewhat secure font. This method is still not completely secure because the font structure can always be unpicked using a PostScript language program, enabling its re-creation by a skilled programmer.

In summary, Harlequin Font Encryption can be used to obscure Type 4 fonts considerably, but not to make their font data completely secure. This is done by selectively encrypting some of the font files, or taking further steps to use the decrypting **run** operator on all PostScript language files in the font explicitly. In any case, some understanding of the working of the font is required of the RIP vendor, assuming it is they who do the encryption. The analysis of the font is quite straightforward.

A Type 4 font will never be completely secure because the secrets that provide its protection are held entirely in the PostScript language world, and can therefore be unpicked by a determined programmer, unless a font vendor works closely with Harlequin to implement an additional encryption scheme within the RIP.

The only way to achieve complete security is to acquire a Type 1 version of the font, convert it to DLD1 format and encrypt that. In this case the secrets stay inside the RIP software and are inaccessible to the PostScript language world, and so the font is highly secure.

Of course, for a composite font, the Type 0 root font should be encrypted regardless of the types of the leaf fonts. (If it is not encrypted, it is possible to find out which glyphs are in which leaf fonts.)

2.5 Strategies for supplying encrypted fonts

There are several options available regarding encryption and protection strategies:

- The RIP vendor supplies fonts keyed to individual RIPs.

  The RIP vendor must have a clear format version of the font. The RIP vendor converts fonts to DLD1 format as appropriate, then encrypts the resulting collection of files to an individual customer's RIP ID on receipt of order.

  The RIP ID is available from the customer's license file or dongle, which the RIP vendor will have kept in a database for maintenance reasons. The RIP vendor needs font encryption tools, available from Harlequin.

  The installation procedure is a binary file copy into the fonts folder in the RIP's file system. Installation tools are trivial.

  This is very secure, because the fonts supplied to the customer will only work with their particular RIP, with their particular license or dongle. In addition, the customer never has a clear format copy of the font, even in transit in an install program.

- The font vendor supplies fonts keyed to individual RIPs.

  The font vendor has a clear format version of the font, and converts fonts to DLD1 format as appropriate. A copy of the RIP is required for this. The font vendor then encrypts the resulting collection of files to an individual customer's RIP ID on receipt of order. The RIP ID is available from the customer's license file (which the customer can read but not alter) or from the RIP System Monitor. The font vendor needs the Harlequin RIP and Harlequin encryption tools, perhaps supplied by the RIP vendor.

  The installation procedure is a straight-forward file copy into the fonts folder in the RIP's file system. Installation tools are trivial.

  This is very secure, because the fonts supplied to the customer will only work with their particular RIP, with their particular license or dongle. The customer never has a clear format copy of the font, even in transit in an install program and neither does the RIP vendor, so the very protective font vendor will be happy.

From the customer's viewpoint, these methods are equally secure. For this level of security, the RIP ID information must be folded into the font data, before it reaches the customer. Someone must do this, either the RIP vendor or the font vendor.

- The font vendor supplies fonts keyed to a particular vendor's RIP.

  The font vendor has a clear format version of the font, and converts fonts to DLD1 format as appropriate. A copy of the RIP is required for this. The font vendor then encrypts the resulting collection of files to the RIP vendor's Harlequin OEM Number. The encrypted files can be sent to all customers with the Harlequin RIP from this particular RIP vendor. The font vendor needs the Harlequin RIP and Harlequin encryption tools, perhaps supplied by the RIP vendor.

  The installation procedure is a binary file copy into the fonts folder in the RIP's file system. Installation tools are trivial.

  This is only fairly secure against unauthorized use of the font, because the supplied font files will work with all Harlequin RIP RIPs supplied by the particular RIP vendor. It is very secure against theft of the font data by other font houses, since the font data are encrypted and only the Harlequin RIP can decrypt it.

- The RIP vendor supplies fonts keyed to their own RIP.

  The comments here are much like those for the previous option, except that the RIP vendor must have a clear format copy of the fonts, and the font vendor need not have the RIP or the encryption tool.

These two methods allow a customer with several copies of the RIP from the RIP vendor to buy only one copy of the font and use it several times without paying, simply by copying the files to the other RIPs.

2.6 Other notes

DLD1 conversion need not be performed using exactly the RIP that the RIP vendor sells; a Macintosh or PC Demo version of Harlequin's integrated RIP product could be used to do the job, and be protected by a dongle. Please contact Harlequin to discuss the matter further if required.

The DLD1 font converter cannot convert encrypted Type 1 fonts to DLD1 format, because it runs the files it is instructed to convert and the PostScript language **`run`** operator does not understand encrypted files. Running the files loads the fonts into virtual memory just as **`findfont`** does.

2.7 Encryption methodology

Harlequin can supply tools to encrypt fonts on Macintosh computers, MS-DOS based PCs and Unix systems. Please contact Harlequin should you require such tools.

Macintosh users should use the FireWorks utility. Please see FireWorks documentation (Tech note Hqn011) for details.

The process of encryption is as follows:

1. Decide whether you wish to encrypt files against the RIP vendor's OEM number, or against the serial number for a specific RIP.

   OEMs may obtain their numbers from Harlequin on demand. It can also be determined directly from a dongle – it is the first two hexadecimal digits of the fourth part of the dongle part number (which is on the dongle label). So, for example, if your dongle part number is ME1-N-12-0BD-0-ALL, your OEM customer number is 0B (hexadecimal), which is 11 decimal.

   Serial numbers are determined from one of several sources:

   - Any RIP from version 3.2 revision 25 onwards (20.2 revision 25 for core RIPs and Unix command line versions): The PostScript language **`serialnumber`** operator will return the correct number. RIPs with GUI front ends will report the number in the System Monitor during RIP start-up. From revision 26 this report is accompanied by a checksum.
   - Previous Unix revisions running from a permit file: The permit file will contain a section similar to:

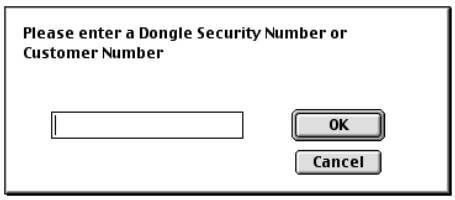
```
Harlequin RIP - Version JAC 20

# License     Node      Floating  Start     Expiry  Update
# Policy(hex) Licenses  Licenses  Date (ymd)Date    Period (mins)
  1           0         1         19940420  20000101   5

# Application data (signed values)

DATA: 6
            0x0B 0  20   0x00ffff00
      #     resolution limit
            3000
            5678
```

- The serial number is shown by the last of the 6 DATA entries – `5678` in the example above.

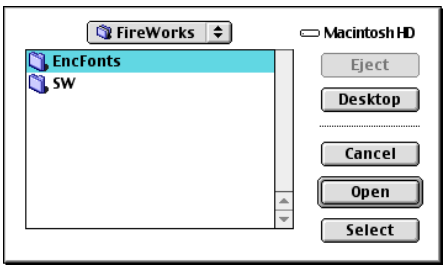
From July 1992 Harlequin has supplied lists of dongle serial numbers when dongles have been delivered to OEMs.

2. Determine which format the fonts will be in when encrypted. Type 1 fonts should be converted to DLD1 before this stage. If necessary determine which files of a Type 4 font require encryption.
3. Move the fonts to be encrypted into a temporary directory or folder which we will refer to as **`OrigFont`** on the Macintosh or PC. Create a second directory or folder into which the encrypted fonts are to be placed. We will call this **`EncFonts.`**
4. Start up the FireWorks utility.

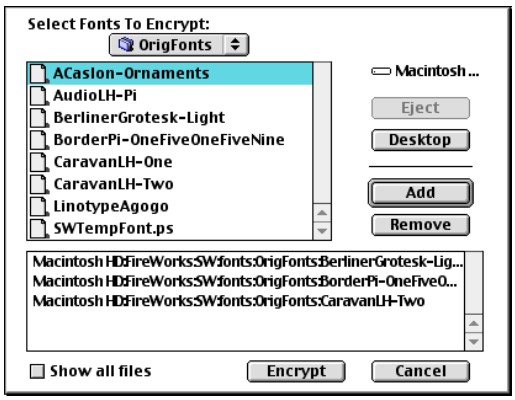
   Selecting **Encrypt fonts** displays the following dialog box. Enter the dongle security number or customer number and click on **OK**. Numbers should be entered in decimal.



A prompt appears asking for a location at which to save the encrypted fonts. Highlight the folder you want to use and click **Select**.



Now you must select the fonts to be encrypted. These fonts can be Type 1, Type 3, Type 4, Type 111 (DLD1 format) or Type 0 (composite header) format. For maximum security, Harlequin recommend the use of DLD1 format fonts.



The encrypted fonts are created in the selected folder.

5. You now have a folder of encrypted fonts that can be copied around as required and shipped to the customer who has the corresponding dongle, or any customers of a particular RIP vendor if you are encrypting using the OEM number.

6. To use these encrypted fonts, copy them into the **`SW/fonts`** folder of the RIP and install the dongle that they were encrypted for. The RIP will automatically decrypt the fonts as required, ensuring that the fonts are never available in decrypted form. The RIP will only be able to decrypt the fonts if the correct dongle is installed.

Change history		
v 1.0	1994.05.03	First issued.
v 1.1	2000.10.02	Removed references to qdongle and hqcrypt. Removed two font encryption methods which are not used. Change document to new format.
v.1.2	2001.06.07	Updated cover page and copyright page. Removed references to ScriptWorks and replaced with Harlequin RIP. No other changes made to text.