Push Calibration

Technical Note Hqn081

May 2016

GLOBAL GRAPHICS®
software

1 Introduction

Calibration is an essential part of keeping a printing process in conformance. In the HMR calibration sets may be created via the Calibration Manager dialog, either by manually entering measurement values or by using Global Graphics' Genlin tool, (or similar as provided by some vendors of the HMR), followed by a final manual step to import the calibration data. This document describes an alternative method called HqnPushCalibration, available in both Harlequin MultiRIP (HMR) v10.0 and Harlequin Host Renderer (HHR) v4.0. As well as providing a route to storing and updating calibration sets in the HHR, it is intended to provide users of the HMR with an alternative which does not require any manual interaction with the Calibration Manager dialog.

Calibration sets may be submitted as a PostScript language file to the RIP, calling the `PushCalibration` procedure in the HqnPushCalibration procset. Calling this procedure does not trigger any raster output, and files may be submitted through any input channel configured on the RIP, such as hot folders, Unix-style sockets and so on. Instead the process creates a stored calibration or *calset* in the **SW** folder. This calibration can be selected later via the Page Setup dialog in the HMR or via a TestConfig file for the HHR.

The `/ProcSetVersion` key is available in the procset so you can determine which version is installed. To determine the current version of the procset use the following:

```
/HqnPushCalibration /ProcSet findresource /ProcSetVersion get ==
```

For more information on the theory of calibration and how it is used within the RIP see *Chapter 12* of the *OEM manual* and *Chapter 10* of the *Extensions manual*.

Note: A limited version of HqnPushCalibration was available in the Harlequin Server RIP v9.0 onwards.

2 Process

There are four steps to creating, using and updating a calibration set (calset) with HqnPushCalibration. These steps are described in the following sections of this document.

1. Prepare to create the calset by printing and measuring a suitable test job, or alternatively by preparing to install data from a linearization profile such as the gray balance profiles or output profiles created by SetGold or SetGoldPro.
2. Create and run a PostScript language job containing the measurement data, or alternatively a reference to the linearization profile. This creates the calset.

3. Select the calset for use with a job.
4. Update the calset periodically. For example, to account for wear in the printer, consumable replacement, or changes in environmental conditions.

Step 1 Preparing to create the calset

This section provides the details for step 1 of the process for both the HMR and HHR.

There are two methods to achieve this:

- By printing and measuring a test job, See 1.1a.
- By installing linearization data.

  When using this route start at "Preparing to create a calset by installing linearization data—HMR/HHR" on page 7 and then move onto step 2.

Step 1.1a By printing and measuring a test job—HMR/HHR

First, a suitable test job (target) must be selected. This may be CMYK, monochrome or a `DeviceN` color space such as Hex. If you want separate calibrations for each of the C, M,Y and K channels, as would be normal when printing a target for CMYK composite, you should print a test job (target) for `DeviceCMYK`. However, in most cases when you print a target for CMYK Separations, you are effectively printing a target for `DeviceGray`, that is, a monochrome target. This is because you are, in effect, calibrating a monochrome plate.

The first time you prepare to create a calibration with a new device, ink and media combination you need to find out the appropriate drying time for your printer and media. That is, the period after which the measurements do not change. Print drying times are determined by testing. Taking measurements for example at 5, 30, 60, minute intervals and then 4, 8, 12 and 24 hours is a commonly used procedure. for a new device, ink and media combination.

If HqnPushCalibration is to be used to import the measurements, the measurement system must be `% Dot`. Other measurement systems may be used in conjunction with linearization data. Go to "Preparing to create a calset by installing linearization data—HMR/HHR" on page 7 if you prefer to use this route instead.

If you are calibrating spot colors they must be present in the target that you are measuring. See "Example target with Spot color" on page 18. The page setup that you are printing the target with must have the spot color enabled in the Color Separations dialog. For example:

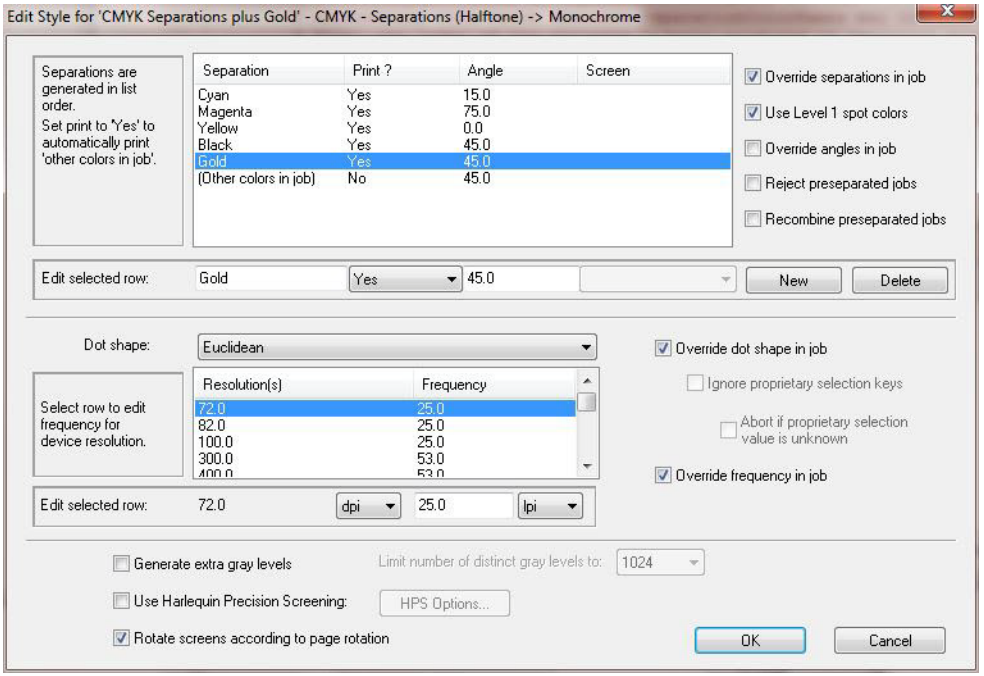


Figure 1 Example dialog with spot color

1.1.1 Printing and measuring a test job—HMR

If you are preparing to create a calset from linearization data you should omit this step and see "Preparing to create a calset by installing linearization data—HMR/HHR" on page 7 and then go to step 2.

When using the HMR for a given device, a test strip is printed from the "Print Calibration" dialog. More information about printing test jobs with the HMR is provided in the *OEM manual*.

If you want to create a monochrome target, you should print your target by selecting **Print for: Monochrome only** and **Print uncalibrated target**. If you want to see your measurements, once you have imported them, (see "Creating the calset—HMR" on page 10), you will need to select the **Monochrome ColorSpace** in the Calibration Manager and then **Edit from Uncalibrated target**.

Note: When you create the PostScript language job for HqnPushCalibration in step 2 you will need to set `/ColorSpace (DeviceGray)` and `/ChannelColors [(Gray)]`.

If you want to create a DeviceCMYK target, you should print your Target by selecting **Print for: Process colors only** or **Print for: Process and spot colors** and **Print uncalibrated target**. If you want to see your measurements once you have imported them, (see "Creating the calset—HMR" on page 10, you will need to select the **CMYK ColorSpace** in the Calibration Manager and then **Edit from Uncalibrated target**.

`DeviceN` color spaces support the use of high-fidelity and multitone color. which is the use of more than the standard CMYK process colorants to produce an extended gamut, or range of colors. An example of such a system is the PANTONE® Hexachrome™ system, which uses six colorants: the standard cyan, magenta, yellow, and black, plus orange and green.

Note: When you create the PostScript language job for HqnPushCalibration in step 2 you will need to set `/ColorSpace (DeviceCMYK)` and `/ChannelColors [(Cyan) (Magenta) (Yellow) (Black)]`.

Once you have printed your target and left it to dry you should measure the patches on the test strip in `% Dot`. There are then several options for importing the measurements into the RIP:

- They may be manually entered into the RIP calibration editor. This method is fully described in *Chapter 12* of the *OEM manual*, and is not discussed any further here.

- The patches can be measured using Genlin and the measurements imported directly into the RIP using the Import button on the calibration editor dialog. See *Chapter 12* of the *OEM manual*. HqnPushCalibration is not required if you are using this method.
- The patches can be measured using a third-party tool and imported into the RIP via HqnPushCalibration (as discussed here). See "Creating the calset—HMR" on page 10.

More information about printing test jobs within the HMR is provided in the *OEM Manual*.

As an alternative to using the targets provided with the HMR you may wish to create your own, or use one provided with a third-party tool. If you do this it is important to print it to a simple non-color managed Page Setup with the same resolution and screening as the Page Setup with which it is ultimately intended to be used to print real jobs.

You must also create a target in the Hqn format. If you would later like to be able to edit this calibration in the HMR Calibration Manager, it must have no more than 36 patches per channel.

Note: Information on the format for Hqn targets can be found in *Appendix B* of the *Plugin Kit Guide for OEMs*.

1.1.2 Printing and measuring a test job—HHR

For the HHR, you must create a suitable `DeviceGray` or `DeviceCMYK` or `DeviceN` target (for example Hex) or alternatively use a target provided with a third-party tool. When creating a test job for the HHR there should be a reasonable number of patches for each channel, (for example, the HMR uses 18 patches per channel for its default CMYK target), with more patches generally producing a more accurate calibration.

The test target should be printed using a simple non-color managed TestConfig file with the same resolution and screening as the TestConfig that this calset is ultimately intended to be used with for printing real jobs.

If you are calibrating spot colors they must be specified in the TestConfig file. For example:

```
<<
  /PageBufferType /LE
  /Separations true
  /SeparationColorNames [ /Gold ]
  /SeparationOrder [ /Gold /Black /Cyan /Magenta /Yellow ]
>> setpagedevice
```

Once you have printed the target you will need to measure it using a suitable measuring instrument or third-party tool. After printing the test job it is essential to allow enough time for the ink to dry before measuring it. The measurement system must be `% Dot`.

When you create the PostScript language job for HqnPushCalibration in step 2, the nominal patch values of the target must be added to the `/PatchValues` dictionary.

More information about printing jobs with the HHR is provided in the *Developer's Guide*.

You are now ready to move on to step 2 to import the data into the RIP using HqnPushCalibration.

Note: If you have both the HMR and HHR it is possible to use the HMR to print a calibration target and then use the measured values in an HHR RIP by creating a PostScript language job at step 2 in the same way as for HMR, but adding the nominal patch values of the target to the `/PatchValues` dictionary.

Step 1.1b Preparing to create a calset by installing linearization data—HMR/HHR

For many devices (such as most inkjet printers) it is essential to use a calibration set based on a reference characterization—often called a linearization—(or to use some other method of ink control). The typical response of an inkjet printer in its raw state is to produce very dark images and deliver so much ink that there are severe problems with drying time or ink running across the printed page.

In order for HqnPushCalibration to create a calset from linearization (or characterization) data, this data must be made available to the RIP in the format of a Harlequin profile placed in one of a number of specific locations. Some plugins for the HMR provide ready-made profiles which are already in a suitable location. Alternatively, you can make a suitable profile (such as a gray balance profile), using SetGold or SetGoldPro.

A gray balance (or ink limiting only) profile is sufficient for the purposes of creating a calibration set. This can be created with SetGold or SetGoldPro. However, if you have used SetGoldPro to create an output profile for the HMR this may also be used as the start point for creating a calibration set. (Since the HHR uses the ICC route to color management exclusively, it should never be necessary to create an output profile from SetGoldPro for the HHR). For more information see the *SetGoldPro* document.

If you wish to make a profile using some other method, it must be in the format described in the document *Profiles and Calibration*. The Linear profiles in the **SW\Config\Profiles** folder are simple examples of profiles conforming to the required format.

When creating a calset by installing linearization data, any of the measurement systems available in the profile may later be used for updating the calibration, so as to print according to the linearization again, despite, for example, wear in the printer, consumable replacement or different environmental conditions. At step 2 you will need to specify which measurement system you would like to use for any such updates. If you are creating a calibration set using a pre-prepared profile, such as one supplied with an output plugin, you may in fact wish to update the calibration immediately in order to correct it for any differences between the printer on which the profile was made and your own printer. In this case simply proceed to step 4 "Updating the calset" on page 12 straight after finishing steps 2 and 3.

When using spot colors if the profile has no curve for the spot color the Default curve is used. If there is no Default the Black curve is used or a curve that has Black as part of its string for example `(Hex Black)`.

In summary, the procedure is:

- Make a profile using SetGold(Pro) or some other method. (Omit this step when using a profile provided with an output plugin).
- Have a look at the profile to see which measurement systems are available and make a note of the one that you would like to use for any future updates to the calibration set. (If you are using a profile supplied with an output plugin you should be able to find it in one of the locations listed below).
- Unless the profile is supplied as part of an output plugin you will need to ensure it is placed in one of the following locations:

For the HMR:

For plugins for which there is a single top-level directory within the **SW\devices** directory it should be placed in one of the following locations:

SW\devices\plugin_name\Profiles\colorspace\device_name

SW\devices\plugin_name\Profiles\device_name

If the plugin executable is placed directly in the **SW\devices** directory, the associated device profiles must always be linked to a color space and stored in:

SW\devices\Profiles\colorspace\device_name

Lastly, if none of the above locations are suitable, profiles may also be placed in:

SW\config\Profiles\colorspace

For the HHR profiles should be placed in:

SW\config\Profiles\colorspace

For example, the system default linear CMYK profile is:

SW\Config\Profiles\CMYK\Linear

Note: For HMR and HHR, if you wish to create a calset from one of the default Linear profiles—perhaps in order to update it with real measurement data either immediately or later on in order to ensure that you still really do have a linear characterization—there is nothing to do at this step. You can proceed straight to step 2, setting the `/Profile` entry to (Linear) in the PostScript language job provided to HqnPushCalibration. Alternatively, since creating a calset by printing and measuring a test job has the same result as creating a calset by installing Linear characterization data and then updating it, you could wait until you are ready to do the update and do step 1.1a instead.

Note: HqnPushCalibration does not support the installation of linearization data for `Press` except using the default Linear Press profiles (using the `% Dot` measurement system). The procedure is as described above.

Step 2 Creating the calset

This section provides the details for step 2 of the process for both the HMR and HHR.

Step 2.1 Creating the calset—HMR

This section describes the creation of a calset via HqnPushCalibration. Other ways to create a calset are described in *Chapter 12* of the *OEM manual*, and include entering the measured values manually into the HMR calibration editor dialog. Similarly, if Genlin is used to perform the measurements, they can be entered into the RIP using the **Import** button on the HMR calibration editor dialog. For more information see *Chapter 12* of the *OEM manual*. If either of these methods are used, there is no need to make use of HqnPushCalibration at all.

The use of HqnPushCalibration replaces the manual steps involved in using the Calibration Editor dialog. Instead a simple PostScript language job is created and run. The required format is described in "HqnPushCalibration format for creating a calset" on page 12.

This job should be run using the same Page Setup as was used to print the original calibration target. This will automatically create the calibration (calset) which will be added to a file named **Default** or to a file specified by `/CalbrationGroupName` inside the appropriate folder for the device:

SW\Config\Devices\DevCalibration\<device-name>\<colorspace>

For example:

SW\Config\Devices\DevCalibration\TIFF\Default

Before the new calibration is added, a backup file is saved in:

SW\Config\Devices\DevCalibration\<device-name>\<colorspace>_Backup

with the same name such as **<colorspace_Backup>**.For example:

SW\Config\Devices\DevCalibration\TIFF\CMYK_Backup\Default

Step 2.2 Creating the calset—HHR

The measurements taken must be transformed into the correct format for HqnPushCalibration. The required format is described in "HqnPushCalibration format for creating a calset" on page 12 and takes the form of a PostScript language job. This job can be run with any TestConfig file in place, (and would usually be run to a very simple one, such as the one used to print the original calibration target).

This will automatically create the calibration (calset) which will be added to a file named Default or to a file specified by `/CalbrationGroupName` inside the appropriate folder for the device in the following location:

SW\Config\Calibration\<device name>\<colorspace>

For example:

SW\Config\Calibration\TIFF\CMYK\Default

where the DeviceName and ColorSpace are the `/DeviceName` and `/ColorSpace` provided in the PostScript language job.

A backup file is saved in the folder:

SW\Config\Calibration\<DeviceName>\<ColorSpace>_Backup

with the same name as the original. For example:

SW\Config\Calibration\TIFF\CMYK_Backup\Default

Step 3 Selecting the calset for use with a job

This section describes step 3 of the process for both the HMR and HHR.

Further information on the different types of calsets and the calibration curves they represent, such as, `DeviceCurve`, `ToneCurve`, `IntendedPressCurve`, and `ActualPressCurve` can be found in *Section 10.10.7* of the *Extensions Manual*. *Section 10.10.11* discusses the order in which they are applied.

Step 3.1 Selecting the calset for use with a job—HMR

This is done by selecting the calset (or several calsets) in the Page Setup Manager. More information about the use of the Page Setup Manager can be found in the *OEM manual*.

Step 3.2 Selecting the calset for use with a job—HHR

Selecting the calset for use with the job is done by editing a TestConfig file to refer to the calset (or several calsets). In this case the relevant TestConfig file is the one that will be used to print the actual jobs for which the calibration is needed.

The required format is found in "The TestConfig format to use the calset—HHR only" on page 33.

Step 4 Updating the calset

This section provides the details for step 4 of the process for both the HMR and HHR.

It is a good idea to periodically update a calset (regardless of whether it was created by measuring a test job or by importing linearization data). This allows the final printed output of your jobs to remain more stable despite possible changes in your printing conditions, such as wear of the printer, consumable replacement or differing environmental conditions. To do this, a test target must be printed and measured by following steps 1 and 2, but in this case the target must be printed with a Page Setup or TestConfig file which uses the calset that you are trying to update. Updating a calset also requires a slightly different format for the PostScript language job used with HqnPushCalibration at step 2. In this case `/Name` is the name of the existing calset that you want to update, and there is an additional key `/SaveAsName` which provides the name for the updated calset. This format is described in "PushCalibrationUpdate format for updating a calset" on page 30.

If you want to make the new calset overwrite the old one, you can do so by making `/SaveAsName` the same as `/Name`, and setting `/Replace` to `true` in the dictionary passed to HqnPushCalibration—see "info_dictionary" on page 20. If you decide to make the new calset overwrite the old one, it is not necessary to repeat step 3 above as the Page Setup or TestConfig file will already contain the required information to use this calset.

An alternative to updating an existing calset is to start again by printing and measuring a new test target, by repeating steps 1 and 2 above. This route is likely to be chosen for tone curve calsets, since the measurements may have been created by repeated experimentation in conditions in which it may be more confusing to try to update an existing calset. In this case the test target should be printed with a similar Page Setup or TestConfig as the original, without any calibration applied.

3 HqnPushCalibration format for creating a calset

This section is divided into two parts. The first part "Using the PushCalibration procedure to create a calset from measurements" on page 13 describes how to create a calset by doing measurements. The second part, "Creating a calset from a linearization profile" on page 27 is what to do when creating a calset from a linearization profile.

3.1a Using the PushCalibration procedure to create a calset from measurements

When using HqnPushCalibration to create a calset from measurements, the `PushCalibration` procedure is called as:

```
color_array     measurements_dictionary     info_dictionary
/HqnPushCalibration /ProcSet findresource /PushCalibration get exec
```

The details of the parameters are described below:

`color_array`

(PostScript language array)

An array consisting of the colors that were measured by the measurement device. It must either contain a single colorant or all the colorants in the `/ChannelColors` array. Typically, for a CMYK device this will consist of `[/Cyan /Magenta /Yellow /Black]` and for monochrome it will be `[/Gray]`.

If you are using a spot color it must be specified here. For example `[/Cyan /Magenta /Yellow /Black /Gold]`

If only one color strip has been measured, and the measured values are expected to be used for all colors, the array should only contain the measured color, for example `[/Cyan]`.

Note: If creating a calset by installing linearization data (see "Creating a calset from a linearization profile" on page 27), this should be empty, that is: `[]`

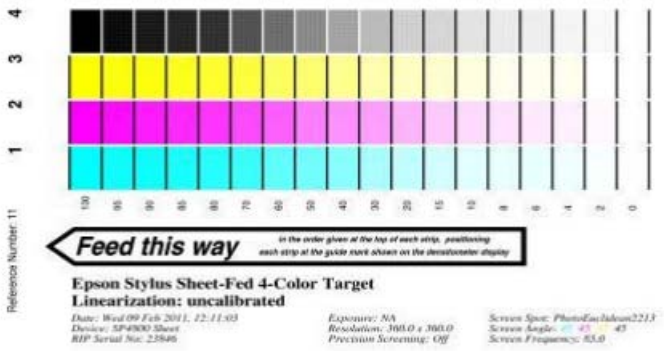


Figure 2 Example target

`measurements_dictionary`

(PostScript language dictionary)

This dictionary contains the measurements that were taken by the measuring device. The keys to the dictionary are the elements of the `color_array`. The value of each key is the array of measurements.

All of the measured value arrays must be the same length and for the HMR the measurements must have been taken at patch values corresponding to those in the appropriate target. For the HHR the measurements must have been taken at the patch values given in the `/PatchValues` array.

Note: If creating a calset by installing linearization data (see "Creating a calset from a linearization profile" on page 27), this should be empty, that is: `<<>>`.

For example, considering the Target in Figure 2, if `color_array` is `[ /Cyan ]`, that is, only the Cyan strip is measured and the color strip is fed as indicated in the Target, the `measurements_dictionary` would be of the form:

```
<<
/Cyan [
    100.0 % C100
    100.0 % C95
    100.0 % C90
    98.5  % C85
    93.0  % C80
    87.5  % C70
    76.5  % C60
    66.0  % C50
    50.5  % C40
    45.0  % C30
    36.0  % C20
    28.5  % C15
    17.5  % C10
    12.5  % C8
    5.0   % C6
    2.5   % C4
    1.0   % C2
    0.0   % C0
  ]
>>
```

If `color_array` is `[/Cyan /Magenta /Yellow /Black]`, the `measurements_dictionary` would be of the form:

```
<<
/Cyan [
    100.0 % C100
    100.0 % C95
    100.0 % C90
    98.5  % C85
    93.0  % C80
    87.5  % C70
    76.5  % C60
    66.0  % C50
    50.5  % C40
    45.0  % C30
    36.0  % C20
    28.5  % C15
    17.5  % C10
    12.5  % C8
    5.0   % C6
    2.5   % C4
    1.0   % C2
    0.0   % C0
  ]
/Magenta [
    100.0 % M100
    100.0 % M95
    100.0 % M90
    98.5  % M85
    93.0  % M80
    87.5  % M70
    76.5  % M60
    66.0  % M50
    50.5  % M40
    45.0  % M30
    36.0  % M20
    28.5  % M15
    17.5  % M10
    12.5  % M8
    5.0   % M6
    2.5   % M4
    1.0   % M2
    0.0   % M0
  ]
```

```
/Yellow [
    100.0 % Y100
    100.0 % Y95
    100.0 % Y90
    98.5  % Y85
    93.0  % Y80
    87.5  % Y70
    76.5  % Y60
    66.0  % Y50
    50.5  % Y40
    45.0  % Y30
    36.0  % Y20
    28.5  % Y15
    17.5  % Y10
    12.5  % Y8
    5.0   % Y6
    2.5   % Y4
    1.0   % Y2
    0.0   % Y0
  ]
/Black [
    100.0 % K100
    100.0 % K95
    100.0 % K90
    98.5  % K85
    93.0  % K80
    87.5  % K70
    76.5  % K60
    66.0  % K50
    50.5  % K40
    45.0  % K30
    36.0  % K20
    28.5  % K15
    17.5  % K10
    12.5  % K8
    5.0   % K6
    2.5   % K4
    1.0   % K2
    0.0   % K0
  ]
>>
```

If `color_array` is `[/Cyan /Magenta /Yellow /Black /Gold]`, with the target as shown below:

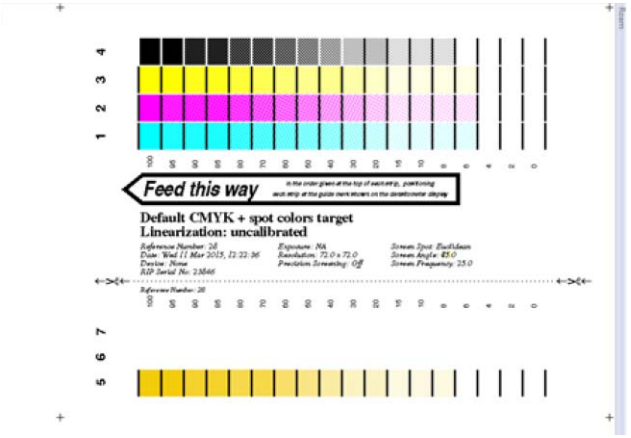


Figure 3 Example target with Spot color

The `measurements_dictionary` would be the same as the `[/Cyan /Magenta /Yellow /Black]` example above but with the addition of:

```
/Gold [
  100.0 % G100
  100.0 % G95
  100.0 % G90
  98.5  % G85
  93.0  % G80
  87.5  % G70
  76.5  % G60
  66.0  % G50
  50.5  % G40
  45.0  % G30
  36.0  % G20
  28.5  % G15
  17.5  % G10
  12.5  % G8
  5.0   % G6
  2.5   % G4
  1.0   % G2
  0.0   % G0
  ]
```

If color_array is `[/Gray]` with the Target as shown below,

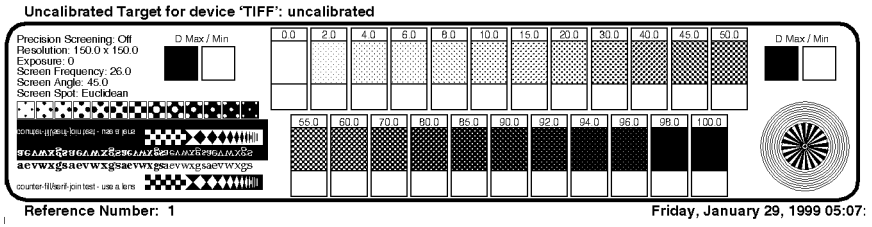


Figure 4 Example target

...the `measurements_dictionary` would be of the form:

```
<<
/Gray [
  0.00  % g0
  0.02  % g2
  0.035 % g4
  0.07  % g6
  0.08  % g8
  0.10  % g10
  0.15  % g15
  0.20  % g20
  0.30  % g30
  0.40  % g40
  0.45  % g45
  0.50  % g50
  0.55  % g55
  0.60  % g60
  0.69  % g70
  0.80  % g80
  0.85  % g85
  0.90  % g90
  0.94  % g92
  0.95  % g94
  0.96  % g96
  0.98  % g98
  1.00  % g100
]
>>
```

Note: The patch values for this case are in the opposite order to that given for the other examples above. This is important for the HMR since it will take the patch values directly

from the monochrome target. For the HHR the patch values must simply match the target used and be listed in the same order as they are given in the `/PatchValues` array.

3.1.1 info_dictionary

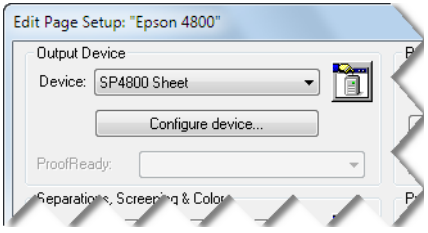
(PostScript language dictionary) This consists of the following key/value pairs:

`/DeviceName`

(PostScript language string) (required)

The device name as it is known in the RIP. This is the name of the device that was used to print the Target as it appears in the Page Setup dialog for the HMR. For example, for the Epson 4800 Sheet device, it is `(SP4800 Sheet)`. For the HHR use a name of your choice, (or `Press` if calibrating for Press). This name will usually become part of the path to the calset.

When calibrating for Press, the `/DeviceName` should be `(Press)`.



`/Name`

(string) (required)

The name of the new calibration set. It can be up to 60 characters long.

`/CalibrationGroupName`

(PostScript language string) (optional)

This is the calibration group name for the calibration set.

The default value is: `(Default)`.

`/ColorSpace`

(PostScript language string) (required)

One of `(DeviceCMYK)` or `(DeviceGray)` or an appropriate DeviceN color space such as `(Hex)`.

For CMYK plus one or more spot colors use `(DeviceCMYK)`.

`/NumberOfChannels`

(Integer) (required)

It must be the number of elements in the `ChannelColors` array.

`/ChannelColors`

(PostScript language array of strings) (required)

One of: `[(Cyan) (Magenta) (Yellow) (Black)]` or `[(Gray)]`, or `[(Cyan) (Magenta) (Yellow) (Black) (Gold)]` for spot color Gold, or a suitable array for a `DeviceN` color space, for example: `[(Hex Cyan) (Hex Magenta) (Hex Yellow) (Hex Black) (Hex Orange) (Hex Green)]` for a Hex color space.

`/ChannelType`

(A 16 element PostScript language array of integers) (optional, but required for HMR when spots are in use)

It corresponds to the `/ChannelColors` array as follows: For each color in `/ChannelColors`, if the color is a process color, the corresponding `/ChannelType` is `1`. For each spot color, the corresponding `/ChannelType` is `2`. The rest of the elements should be set to `0`.

So for `/ColorSpace (DeviceCMYK)`

```
/ChannelColors [(Cyan) (Magenta) (Yellow) (Black)
(Orange) (Green)]
```

the `/ChannelType` should be:

```
/ChannelType [1 1 1 1 2 2 0 0 0 0 0 0 0 0 0 0]
```

For `/ColorSpace (Hex)`

```
/ChannelColors [Hex (Cyan) (Hex Magenta) (Hex Yellow)
(Hex Black) (Hex Orange) (Hex Green)]
```

the `/ChannelType` should be:

```
/ChannelType [1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0]
```

`/HWResolution`

(2-element PostScript language array) (required)

The resolution `[horizontal vertical]` in dpi. For example, `[360.0 360.0]`.

`/Replace`

(PostScript language boolean) (optional: default=`false`)

If a calibration set already exists with the name set by the `/Name` key in the `info_dictionary`, this key specifies whether the existing calibration set will be replaced. If the value is set to `false`, a unique string is constructed by appending `_n` to the `Name`, where `n` is a number between `0` and `999`.

`/ScreenName`

(PostScript language string) (required)

The screen name. For example, for the target above, it will be `(PhotoEuclidean2213)`.

`/ScreenFrequencyRange`

(PostScript language 2 element array) (required)

The screen frequency range in lpi for which this calibration set should be applied. For example, `[ 130.0 140.0 ]`.

`/NegativeMedia`

(PostScript language boolean) (optional)

Whether the calibration set is intended to be applied to negative media from the RIP: `true` or `false`.

`/ForceSolids`

(PostScript language boolean) (optional)

Defaults to `true` to be backwards compatible, and match the user interface in the HMR where it is on by default. When `true` it indicates that real jobs printed with this calset should always use solid color when the job requests 100% color.

`/PushToneCurves`

(PostScript language boolean) (optional)

Defaults to `false`. When `true`, the curves are pushed as Tone Curves rather than Device or Press curves.

Note: The `/PushToneCurves` key is ignored when creating a calset from a linearization profile.

`/MatchResolution`

(PostScript language boolean) (optional)

Indicates whether a warning should be given if the actual resolution used, when printing a job with this calset, is not the same as the resolution in the calset.

Defaults to `true`.

`/MatchScreenName`

(PostScript language boolean) (optional)

Indicates whether a warning should be given if the actual screen name used, when printing a job with this calset, is not the same as the screen name in the calset.

Defaults to `true`.

`/MatchScreenFrequency`

(PostScript language boolean) (optional)

Indicates whether a warning should be given if the actual screen frequency used, when printing a job with this calset, is not the same as the screen frequency in the calset.

Defaults to `false`.

`/MatchSign`

(PostScript language boolean) (optional)

Indicates whether this calibration can be used for both positive and negative media. If `MatchSign` is `false` it can be used for both, otherwise only for the one indicated by `NegativeMedia`.

Defaults to `false`.

For the HHR there is an extra key:

`/PatchValues`

(Dictionary) (required for the HHR)

A dictionary containing nominal values where the measurements were taken. If the patch values are the same for all the channels it is only necessary to supply the first channel. (If installing a linearization profile for the HHR these values will not yet have been used but it is recommended that they are filled-in with the `patchValues` where future measurements will be made to update the calset.) Required for the HHR and optional and ignored for the HMR.

Example:

```
/PatchValues << /Cyan [1.0 0.95 0.90 0.85 0.80 0.70
0.6 0.5 0.40 0.30 0.20 0.15 0.10 0.08 0.06 0.04 0.02
0.0] >>
```

Note: The above patch values are correct for the default CMYK target. The default monochrome target has the following patch values:

```
/PatchValues <</Gray [0.00 0.02 0.04 0.06 0.08 0.10
0.15 0.20 0.30 0.40 0.45 0.50 0.55 0.60 0.70 0.80 0.85
0.90 0.92 0.94 0.96 0.98 1.00]>>
```

For the Target shown in Figure 2, page 14, the `info_dictionary` might look like this for the HHR:

```
<<
  /DeviceName (SP4800 Sheet)
  /Name (My Epson 4800 Sheet calset)
  /ColorSpace (DeviceCMYK)
  /ChannelColors [ (Cyan) (Magenta) (Yellow) (Black) ]
  /ChannelType [1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0]
  /NumberOfChannels 4
  /HWResolution [ 2880.0  1440.0 ]
  /ForceSolids  true
  /PushToneCurves  false
  /Replace  false
  /ScreenName (PhotoEuclidean2213)
  /ScreenFrequencyRange [ 130.0 140.0 ]
  /NegativeMedia false
  /PatchValues
   << /Cyan [ 1.0 0.95 0.90 0.85 0.80 0.70 0.6 0.5 0.40 0.30 0.20 0.15 0.10
              0.08 0.06 0.04 0.02 0.0]
   >>
>>
```

...and for the HMR:

```
<<
  /DeviceName (SP4800 Sheet)
  /Name (My Epson 4800 Sheet calset)
  /ColorSpace (DeviceCMYK)
  /ChannelColors [ (Cyan) (Magenta) (Yellow) (Black) ]
  /ChannelType [1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0]
  /NumberOfChannels 4
  /HWResolution [ 2880.0  1440.0 ]
  /ForceSolids  true
  /PushToneCurves  false
  /Replace  false
  /ScreenName (PhotoEuclidean2213)
  /ScreenFrequencyRange [ 130.0 140.0 ]
  /NegativeMedia false
>>
```

This is an example for a spot color:

```
<<
  %  /DeviceName (SP4800 Sheet) % target device
  /DeviceName (None) % target device
  /Name (Test calset 2)
  /ColorSpace (DeviceCMYK)
  /NumberOfChannels 5
  /ChannelColors [ (Cyan) (Magenta) (Yellow) (Black) (Gold) ]
  /ChannelType [1 1 1 1 2 0 0 0 0 0 0 0 0 0 0 0]
  /HWResolution [ 300.0 300.0 ]
  /ScreenName (Euclidean)
  /ScreenFrequencyRange [ 130.0 140.0 ]
  /NegativeMedia false
  /MeasurementSystems   [(% Dot) (% Dot) (% Dot) (% Dot) (% Dot)]
>>
```

This is an example for DeviceN:

```
>>
  % /DeviceName (SP4800 Sheet) % target device
  /DeviceName (6 Color Halftone) % target device
  /Name (Tinas Hex calset)
  /ColorSpace (Hex)
  /NumberOfChannels 6
  /ChannelColors [
  (Hex Cyan)
  (Hex Magenta)
  (Hex Yellow)
  (Hex Black)
  (Hex Orange)
  (Hex Green) ]
  /ChannelType [1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0]
  /HWResolution [ 300.0 300.0 ]
  /ScreenName (Euclidean)
  /ScreenFrequencyRange [ 130.0 140.0 ]
  /NegativeMedia false
  /MeasurementSystems [(Status T \(X-Rite\))(Status T \(X-Rite\))
  (Status T \(X-Rite\))(Status T \(X-Rite\))(Status T \(X-Rite\))
  (Status T \(X-Rite\))]
>>
```

3.1b Creating a calset from a linearization profile

When creating a calset from a linearization profile the `info_dictionary` is as described below.

The `PushCalibration` procedure is called as:

```
color_array measurements_dictionary info_dictionary
/HqnPushCalibration /ProcSet findresource /PushCalibration get exec
```

In this case, the `color_array` and `measurements_dictionary` should be empty as this information is contained in the linearization profile, so the HqnPushCalibration procedure call becomes:

```
[] <<>> info_dictionary
/HqnPushCalibration /ProcSet findresource /PushCalibration get exec
```

It is not possible to push tone curves at the same time as creating a calset from a linearization profile so if there is a `/PushToneCurves` entry in the `info_dictionary`, this will be ignored.

In addition to the `info_dictionary` keys described in section 3.1a above, the following two keys must be supplied:

`/Profile`

(String containing the profile name)

For example: `(SWOP \(CGATS TR001\))`

Note: When creating a calset for Press via HqnPushCalibration, only the (Linear) profile is supported.

`/MeasurementSystems`

(Array of PostScript language strings)

The measurement system to use. These must all be contained in the profile, and are also the systems that must be used for any future updates to the calset. If all the channels use the same measurement system, as would normally be the case, only the first channel need be specified here.

Any absolute measurement system may be used, as long as there is a conversion table in the profile with a corresponding name. A list of recommended measurement system names can be found in the *Profiles and Calibration* document.

Note: If installing a (Linear) profile the measurement system must be `[(% Dot)]`.

The `MeasurementSystem` names are compared with the conversion table names after conversion of both to upper case and stripping any non-alphanumerical characters, (so `(% Dot)` matches `(%dot)` but not `(% Dot1)`. In addition, `(Positive % Dot)` matches `(% Dot)`.

Some examples:

```
[(% Dot) (% Dot) (% Dot) (% Dot)]       % or [(% Dot)]

[(Status T \(X-Rite\))]

[(DIN \(Gretag\))]

[(DIN NB \(Gretag\))]

[(Status T \(Gretag\))]

[(DIN \(X-Rite\))]

[(DIN NB \(X-Rite\))]

[(Status E \(X-Rite\))]

[(Status I \(X-Rite\))]
```

Note: HqnPushCalibration does not support the use of relative measurement systems, such as Dot Gain.

An example `info_dictionary` for this case could be for the HMR:

```
<<
  /DeviceName (SP4800 Sheet)
  /Name (My Epson 4800 Sheet calset)
  /ColorSpace (DeviceCMYK)
  /ChannelColors [ (Cyan) (Magenta) (Yellow) (Black) ]
  /ChannelType [1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0]
  /NumberOfChannels 4
  /MatchResolution true
  /MatchScreenName true
  /MatchScreenFrequency true
  /HWResolution [ 2880.0  1440.0 ]
  /ForceSolids  true
  /Replace  false
  /ScreenName (PhotoEuclidean2213)
  /ScreenFrequencyRange [ 130.0 140.0 ]
  /NegativeMedia false
  /MeasurementSystems [(% Dot) (% Dot) (% Dot) (% Dot)]
  /Profile (SWOP \(CGATS TR001\))
>>
```

...and for the HHR:

```
<<
  /DeviceName (SP4800 Sheet)
  /Name (My Epson 4800 Sheet calset)
  /ColorSpace (DeviceCMYK)
  /ChannelColors [ (Cyan) (Magenta) (Yellow) (Black) ]
  /NumberOfChannels 4
  /MatchResolution true
  /MatchScreenName true
  /MatchScreenFrequency true
  /HWResolution [ 2880.0  1440.0 ]
  /ForceSolids  true
  /Replace  false
  /ScreenName (PhotoEuclidean2213)
  /ScreenFrequencyRange [ 130.0 140.0 ]
  /NegativeMedia false
  /MeasurementSystems [(% Dot) (% Dot) (% Dot) (% Dot)]
  /Profile (SWOP \(CGATS TR001\))
  /PatchValues
   << /Cyan [ 1.0 0.95 0.90 0.85 0.80 0.70 0.6 0.5 0.40 0.30 0.20 0.15 0.10
              0.08 0.06 0.04 0.02 0.0]
   >>
>>
```

4 PushCalibrationUpdate format for updating a calset

To update a calset call the /PushCalibrationUpdate procedure as follows:

```
color_array measurements_dictionary info_dictionary
/HqnPushCalibration /ProcSet findresource /PushCalibrationUpdate get
exec
```

Note: The /PushCalibrationUpdate procedure is a different PostScript language procedure to the /PushCalibration procedure described in the sections above.

Note: When using spot colors if the profile has no curve for the spot color the Default curve is used. If there is no Default the Black curve is used or a curve that has Black as part of its string for example (Hex Black).

The details of the parameters are described below:

color_array

(PostScript language array) (required)

An array consisting of the colors that were measured by the device. It must contain either a single colorant, all the colorants in the /ChannelColors array, or all the colorants in the /UpdateColors array. Typically, for a CMYK device this will consist of [/Cyan /Magenta /Yellow /Black] and for monochrome it will be [/Gray]. Where spot colors are in use this could be (for example), [/Cyan /Magenta /Yellow /Black /Gold].

If only one color strip has been measured, and the measured values are expected to be used for all colors, the array should only contain the measured color for example, [/Cyan].

Note: If creating rather than updating a calset this must contain either a single colorant or the same number of colorants as in the ChannelColors array (or be empty, that is [] if creating a calset by installing linearization data).

`measurements_dictionary`

(PostScript language dictionary) (required)

This dictionary contains the measurements that were taken by the measuring device. The keys to the dictionary are the elements of the `color_array`. The value of each key is the array of measurements.

All of the measured value arrays must be the same length and for the HMR the measurements must have been taken at the patch values corresponding to those in the appropriate target. For the HHR the measurements must have been taken at the patch values given in the `/PatchValues` array.

If only one color is given in the color array, for example `[/Cyan]`, it is assumed that the other colorants have the same measurements.

For example, if `color_array` is `[ /Cyan ]`, that is, only the Cyan strip is measured and the color strip is fed as indicated in the Target, the `measurements_dictionary` would be of the form:

```
<<
/Cyan [
    100.0 % C100
    100.0 % C95
    100.0 % C90
    98.5  % C85
    93.0  % C80
    87.5  % C70
    76.5  % C60
    66.0  % C50
    50.5  % C40
    45.0  % C30
    36.0  % C20
    28.5  % C15
    17.5  % C10
    12.5  % C8
    5.0   % C6
    2.5   % C4
    1.0   % C2
    0.0   % C0
  ]
>>
```

Note: If creating rather than updating a calset the `measurements_dictionary` must contain either a single colorant or the same number of colorants as in the `ChannelColors` array, (or be empty, that is <<>> if creating a calset by installing linearization data).

`info_dictionary`

This is as described for `/PushCalibration` in "info_dictionary" on page 20, with the exception that, when updating a calset, only the following keys are used:

`/DeviceName`
`/Name`
This is now the name of the calset to update.
`/ColorSpace`
`/NumberOfChannels`
`/ChannelColors`
`/Replace`
`/PatchValues`

As with `/PushCalibration` this is only needed for the HHR.

In addition, the following keys are added:

`/SaveAsName`

(PostScript language string) (required)

Name to call the updated calset. It can be made the same as `/Name` if required.

Note: Even if this is set to the same name as the original calset, the original calset will not be overwritten unless `/Replace` is also set to `true`. Otherwise, a new name is created for the updated calset based on the `/SaveAsName` but with a numeral added, if required. See also the `/PushCalibration` description of `/Replace` on page 22).

`/UpdateColors`

(PostScript language array of strings) (optional)

The colors to update—any other channels will be left unaffected.

Defaults to `/ChannelColors`.

Note: If the calset was originally created with the HMR editor all the channels listed here must have had some data applied before they can be updated using `/PushCalibrationUpdate`.

For example:
`[(Cyan) (Magenta) (Black)]`

...would leave the yellow channel unaffected.

If using spot color Gold, `[(Black) (Gold)]` would only update the Black and Gold channels.

Note: `/PushToneCurves` should also be set to `true` if updating a calset for tone curves.

5 The TestConfig format to use the calset—HHR only

This section provides the detail on what to do as previously described in "Selecting the calset for use with a job—HHR" on page 11 (step 3 of the process).

Edit a TestConfig file to use one or more calsets by adding the following to the file:

```
calibration_setup_dictionary /HqnCalibrate /ProcSet findresource
begin
HHRCalibrate
end
```

Note: When a job is run using this TestConfig the `/HWResolution` of the device will be checked against the `/HWResolution` in the calsets. To avoid incorrect warnings it is best to call `HHRCalibrate` after the setpagedevice call which sets up the `/HWResolution` for the device.

The contents of the `calibration_setup_dictionary` should be as follows:

`/DeviceCalset`

(PostScript language string) (optional)

The calset name for the primary device calset—optional, (but usually required as you would usually want to supply a calset for the primary device, for example, an inkjet printer).

For example, `(My Updated Calset)`

`/ColorSpace`

(PostScript language string) (required)

The colorspace for primary device.

For example, `(DeviceCMYK)`

If using CMYK with one or more spot colors use, `(DeviceCMYK)`.

`/DeviceName`

(PostScript language string) (optional)

Must match the `DeviceName` supplied to the HqnPushCalibration procset unless that is `Press`. This name also cannot be `Curves`. This key is necessary so the calset folder for the primary device can be found.

Optional, but required if `/DeviceCalset` is provided.

Note: If the `DeviceName` supplied to HqnPushCalibration was `Press`, this entry should be omitted and either `/ActualPressCalset` or `/IntendedPressCalset` should be provided instead. If the PostScript language job used with the HqnPushCalibration procset used tone curves, this should be omitted and `/ToneCalset` should be provided instead. However, if using calsets for both `Press` and the primary device, this would contain the `DeviceName` for the primary device.

For example, `(My Epson Printer)`

`/ToneCalset`

(PostScript language string) (optional)

The tone curves calset name.

For example, `(My tone calset)`

`/ToneColorSpace`

(PostScript language string) (optional)

Used to find calset folder for tone curves. Optional, but required if `/ToneCalset` is provided.

For example, `(DeviceCMYK)`

`/IntendedPressCalset`

(PostScript language string) (optional)

The name of a calset created from a linearization for the press for which the job was prepared.

For example, `(My SWOP Press calset)`

`/ActualPressCalset`

(PostScript language string) (optional)

The name of a calset created from a linearization for the press on which the job is actually going to be printed.

For example, `(My Fogra press calset)`

`/PressColorSpace`

(PostScript language string)

The press colorspace(s)—required if a press calset is provided. It is used to find the calset folder for the press(es).

Note: It is quite possible for this to be, for example, `(DeviceCMYK)` whilst the `/ColorSpace` provided for the primary device is `(DeviceGray)`.

For example, `(DeviceCMYK)`

`/AbortOnNoMatch`

(PostScript language string) (boolean) (optional)

Whether to abort the job if no suitable calset is found.

Default: `false`.

`/CalibrationGroupName`

(PostScript language string) (optional) (from HHR v4.1r1)

This is the calibration group name for the calibration set. If it is omitted, it is assumed that the group is called `(Default)` otherwise, the group is called the string that `/CalibrationGroupName` points to.

The calset is created in a file under

SW\Config\Calibration\<StringFromDeviceNameKey>\<ColorSpaceFromKey>\<CalibrationGroupNameString>

For example:

SW\Config\Calibration\None\CMYK\MyNewCalGroup

If `/CalibrationGroupName` is omitted, the file is created like this:

SW\Config\Calibration\<StringFromDeviceNameKey>\<ColorSpace>\Default

For example:

SW\Config\Calibration\None\CMYK\Default

Thus, a `calibration_setup` dictionary could be:

```
<<
  /DeviceCalset           (My Calset)
  /ColorSpace             (DeviceGray)
  /DeviceName             (MyPrinter)
  /ToneCalset             (MyToneCalset)
  /ToneColorSpace         (DeviceCMYK)
  /IntendedPressCalset    (MyIPCalset)
  /ActualPressCalset      (MyPressCalset)
  /PressColorSpace        (DeviceCMYK)
  /AbortOnNoMatch         false
  /CalibrationGroupName   (MyNewCalGroup)
>>
```

This dictionary would allow the maximum number of calsets, that is, 4 to be applied at once.

Further information on the corresponding curves, for example, `DeviceCurve`, `ToneCurve`, `IntendedPressCurve`, and `ActualPressCurve` can be found in *Section 10.10.7* of the *Extensions Manual*. *Section 10.10.11* discusses the order in which they are applied.

However, a more usual case, for example for an inkjet device would be expected to look more like this:

```
<<
  /DeviceCalset            (My Calset)
  /ColorSpace              (DeviceCMYK)
  /DeviceName              (MyPrinter)
>>
```

Or if tone curves were being used, like this:

```
<<
  /ColorSpace               (DeviceCMYK)
  /ToneCalset               (My tone curves)
  /ToneColorSpace           (DeviceCMYK)
>>
```

6 Restrictions

For the HMR only, when data is derived from a test target printed from third-party software, the number of patches and the nominal color value for each patch must match those printed using the targets supplied with the HMR. These values are used in the examples shown throughout this document. It is also important to ensure the values in the `measurements_dictionary` arrays are provided in the order shown.

If, for some reason, you need to use some other set of values (such as matching patches used in color bars used for normal press runs), you would need to define a new target within the RIP with the correct number of patches and the correct color values.

Information on the format for Hqn targets can be found in *Appendix B* of the *Plugin Kit Guide for OEMs.* This restriction does not apply to the HHR where you simply enter the `/PatchValues` that you actually used, ensuring that the values in the `measurements_dictionary` arrays are given in the same order as the patches in the `/PatchValues` arrays.

When creating a new calibration this should be done with no calibration in place and pre-color management.

When updating a calset, the Target should be printed using the calset that you wish to update. (This calset should be selected in the Page Setup or used in the TestConfig file). If printing a Target directly from the HMR, any color management will automatically be turned off. However, with the HHR it is necessary to ensure the TestConfig file used does not include color management.

For both the HMR and HHR, but only in the case where a calset is originally created by printing and measuring a target, the measurement system used is `% Dot`. This is because in the absence of a linearization profile the default Linear profile is assumed, which contains the `% Dot` measurement system. When creating a calset from a linearization profile such as a gray balance profile (created by SetGold or SetGoldPro), any absolute measurement system found in the profile can be used. That is, not Dot Gain as that is a relative system.

When updating a calset, the same measurement system must be used as was originally used to create it.

It should also be noted that HqnPushCalibration provides only limited support for `Press`. Specifically, it does not support the creation (or update) of calsets from characterization data other than the (Linear) profile. It is, however, possible to do some basic calibration for Press setups, for example by creating a calset from the (Linear) profile and then updating the calset to compensate for drift away from linear.

7 Errors and warnings

You can view the error messages by opening the HqnPushCalibration procset and searching for `/Messages`.

8 Examples

This section contains a number of examples.

8.1 Creating a calset from measurements—example

This example is for creating a calset from measurements, that is, not from a linearization profile. The example is for the HMR. For the HHR the `info_dictionary` (last parameter) would need a `PatchValues` array added as previously described.

```
[/Cyan]
% array of colorants measured. Here we only measure Cyan.
% The value of /ColorSpace will indicate whether we are asked to assume
% that measurements of all colorants are the same. So, if we only provide
% say /Cyan, we assume that the other colorants will have the same
% measurements. The dictionary below, should have as many key/value pairs,
% as the number of colorants that we indicated above.
```

```
  <<
  /Cyan [
      100.0 % C100
      100.0 % C95
      100.0 % C90
      98.5  % C85
      93.0  % C80
      87.5  % C70
      76.5  % C60
      66.0  % C50
      50.5  % C40
      45.0  % C30
      36.0  % C20
      28.5  % C15
      17.5  % C10
      12.5  % C8
      5.0   % C6
      2.5   % C4
      1.0   % C2
      0.0   % C0
    ]  % array of  measured values. These will correspond to the patches as
       % printed on the target
  >>

  <<
    /DeviceName (SP4800 Sheet) % target device
    /Name (My test calset)
    /ColorSpace (DeviceCMYK)
    /NumberOfChannels 4
    /ChannelColors [ (Cyan) (Magenta) (Yellow) (Black) ]
    /MatchResolution true
    /MatchScreenName true
    /MatchScreenFrequencyRange true
    /MatchSign true
    /HWResolution [ 2880.0  1440.0 ]
    /ScreenName (PhotoEuclidean2213)
    /ScreenFrequencyRange [ 130.0 140.0 ]
    /NegativeMedia false
    /ForceSolids  true
    /PushToneCurves  false
    /Replace  false
  >>

/HqnPushCalibration /ProcSet findresource /PushCalibration get exec
```

8.2 Creating a calset from a linearization profile—examples

This example is for the HHR. For the HMR there is no need to provide the `/PatchValues` entry.

```
[]
<<>>
<<  /DeviceName (Preview)
    /Name (My clrip calset)
    /ColorSpace (DeviceCMYK)
    /NumberOfChannels 4
    /ChannelColors [ (Cyan) (Magenta) (Yellow) (Black)]
    /HWResolution [ 300.0 300.0 ]
    /ScreenName (Euclidean)
    /ScreenFrequencyRange [ 130.0 140.0 ]
    /MatchResolution true
    /MatchScreenName false
    /MatchScreenFrequency false
    /MatchSign true
    /MeasurementSystems [(Status T (X-Rite))]
    /Profile (SWOP (CGATS TR001))
    /NegativeMedia false
    /ForceSolids true
    /Replace false
    /PatchValues << /Cyan [1.0 0.95 0.90 0.85 0.80 0.70 0.6 0.5 0.40 0.30
     0.20 0.15 0.10 0.08 0.06 0.04 0.02 0.0] >>
>>

/HqnPushCalibration /ProcSet findresource /PushCalibration get exec
```

This example has Gold and Silver spot colors. Again, this example is for the HHR. For the HMR there is no need to provide the `/PatchValues` entry.

```
[]
<<>>
<<  /DeviceName (Preview)
    /Name (My clrip calset)
    /ColorSpace (DeviceCMYK)
    /NumberOfChannels 6
    /ChannelColors [ (Cyan) (Magenta) (Yellow) (Black) (Gold) (Silver)]
    /ChannelType [1 1 1 1 2 2 0 0 0 0 0 0 0 0 0 0]
    /HWResolution [ 300.0 300.0 ]
    /ScreenName (Euclidean)
    /ScreenFrequencyRange [ 130.0 140.0 ]
    /MatchResolution true
    /MatchScreenName false
    /MatchScreenFrequency false
    /MatchSign true
    /MeasurementSystems [(Status T (X-Rite))]
    /Profile (My Shiny Profile)
    /NegativeMedia false
    /ForceSolids true
    /Replace false
    /PatchValues << /Cyan [1.0 0.95 0.90 0.85 0.80 0.70 0.6 0.5 0.40 0.30
     0.20 0.15 0.10 0.08 0.06 0.04 0.02 0.0] >>
>>

/HqnPushCalibration /ProcSet findresource /PushCalibration get exec
```

8.3 Updating a calset—examples

The example given is for the HHR. For the HMR there is no need to provide the `/Patch-Values` entry in the `info_dictionary`.

```
[/Cyan /Magenta /Black ]
% array of colorants measured.
% But if we only provide 1 e.g. [/Cyan] it will be assumed that the others
% are the same.
% The dictionary below should have as many key/value pairs, as the number of
% colorants that we indicated above.
% If we wanted to provide measurements for all the colorants we would have
% e.g.
% [/Cyan /Magenta /Yellow /Black] and provide the measurements below.
% Unless updating a calset this must contain either 1 colorant or all of the
% colorants in ChannelColors.  For updating a calset this can contain any
% number of colorants as long as they match those provided in the
% measurements dictionary, and as long as there is either a single colorant
% or all the colorants in /ChannelColors or all the colorants in the
% /UpdateColors array.

<<
/Cyan [
  100.0 % C100
  98.0 % C95
  95.0 % C90
  85.0 % C85
  80.0 % C80
  75.0 % C70
  60.0 % C60
  53.0 % C50
  40.0 % C40
  30.0 % C30
  20.0 % C20
  15.0 % C15
  10.0 % C10
  8.0 % C8
  6.0 % C6
  4.5 % C4
  2.0 % C2
  0.0 % C0
] % array of measured values. These will correspond to the patches as
% printed on the target
```

```
/Magenta [
  100.0 % M100
  98.0 % M95
  95.0 % M90
  85.0 % M85
  80.0 % M80
  75.0 % M70
  60.0 % M60
  53.0 % M50
  40.0 % M40
  30.0 % M30
  20.0 % M20
  15.0 % M15
  10.0 % M10
  8.0 % M8
  6.0 % M6
  4.5 % M4
  2.0 % M2
  0.0 % M0
] % array of measured values. These will correspond to the patches as
  % printed on the target

/Black [
  100.0 % K100
  98.0 % K95
  95.0 % K90
  85.0 % K85
  80.0 % K80
  75.0 % K70
  60.0 % K60
  53.0 % K50
  40.0 % K40
  30.0 % K30
  20.0 % K20
  15.0 % K15
  10.0 % K10
  8.0 % K8
  6.0 % K6
  4.5 % K4
  2.0 % K2
  0.0 % K0
] % array of measured values. These will correspond to the patches as
  % printed on the target
>>
```

```
<<
  /DeviceName (Preview)
  /Name (My clrip calset)
  /SaveAsName (My clrip calset updated Jan 01 2013)
  /ColorSpace (DeviceCMYK)
  /NumberOfChannels 4
  /ChannelColors [ (Cyan) (Magenta) (Yellow) (Black)]
  /UpdateColors[ (Cyan) (Magenta) (Black)]
   % All of these colors must be contained in the /ChannelColors array
   % These are the colors to update - so this will leave yellow unaffected
  /Replace false
  /PatchValues << /Cyan [1.0 0.95 0.90 0.85 0.80 0.70 0.6 0.5 0.40 0.30 0.20
  0.15 0.10 0.08 0.06 0.04 0.02 0.0] >>
>>

/HqnPushCalibration /ProcSet findresource /PushCalibrationUpdate get
exec
```

The following is an example where CMYK and spot color Silver are in use. This is updating a calset previously created from a linearization profile with the MeasurementSystems selected as `[(Status T \(X-Rite\))]` for all channels.

```
[/Cyan /Magenta /Silver ]

<<
/Cyan [
1.72
1.69
1.54
1.32
1.20
1.11
1.09
0.99
0.78
0.76
0.75
0.54
0.44
0.23
0.11
0.09
0.08
0.05
]
```

```
/Magenta [
1.69
1.54
1.32
1.20
1.11
1.09
0.99
0.78
0.76
0.75
0.54
0.44
0.23
0.11
0.09
0.08
0.05
0.04
]
/Silver [
1.80
1.69
1.54
1.32
1.21
1.11
1.09
0.99
0.78
0.76
0.75
0.54
0.44
0.22
0.11
0.09
0.08
0.05
]
>>
```

```
 <<
   /DeviceName (TIFF)
   /Name (My Silver Calset From Profile)
   /SaveAsName (Updated Silver Calset)
   /ColorSpace (DeviceCMYK)
   /NumberOfChannels 5
   /ChannelColors [ (Cyan) (Magenta) (Yellow) (Black) (Silver)]
   /UpdateColors[ (Cyan) (Magenta) (Silver)]
   /Replace false
   /PatchValues << /Cyan [1.0 0.95 0.90 0.85 0.80 0.70 0.6 0.5 0.40 0.30 0.20
   0.15 0.10 0.08 0.06 0.04 0.02 0.0] >>
 >>

/HqnPushCalibration /ProcSet findresource /PushCalibrationUpdate get
exec
```

8.4 A TestConfig for using a calset—examples

The following example shows how the standard CMYKComposite300dpi TestConfig has been modified to use a device calset.

```
% CMYK composite (pixel-interleaved), 300 dpi
% Copyright (C) 2007, 2009 Global Graphics Software Ltd. All rights
reserved.

<<
 /PageBufferType /LE
 /HWResolution [ 300 300 ]
 /InterleavingStyle 2
 /ProcessColorModel /DeviceCMYK
 /SeparationDetails
 <<
 /SeparationStyle 4
 /CompositeColorNames [ /Cyan /Magenta /Yellow /Black ]
 /CompositeOrder [ /Cyan /Magenta /Yellow /Black ]
 >>
 /ValuesPerComponent 256
 /Halftone false
>> setpagedevice

% An example of how to use a calset
% In this case (My HP Device) and (My clrip calset) should be the
% /DeviceName and /Name respectively that were originally supplied
% to the PS job used to create the calset.
```

```
<<
 /ColorSpace              (DeviceCMYK)
 /DeviceName              (My HP Device)
 /DeviceCalset            (My clrip calset)
>>

/HqnCalibrate /ProcSet findresource begin
 HHRCalibrate
 end
```

The following example uses CMYK plus spot colors Gold and Silver.

```
% CMYK separations, 300 dpi
% Copyright (C) 2007-2016 Global Graphics Software Ltd. All rights
reserved.

<<
  /PageBufferType /LE
  /Separations true
  /SeparationColorNames [ /Gold /Silver ]
  /SeparationOrder [ /Cyan /Magenta /Yellow /Black /Gold /Silver]
  /HWResolution [ 300 300 ]
>> setpagedevice

% An example of how to use a calset
% In this case (My HP Device) and (My clrip shiny calset) should be the
% /DeviceName and /Name respectively that were originally supplied
% to the PS job used to create the calset.

<<
/ColorSpace (DeviceCMYK)
/DeviceName (My HP Device)
/DeviceCalset (My clrip shiny calset)
>>
/HqnCalibrate /ProcSet findresource begin
HHRCalibrate
end
```

9 Document history

Change history		
2.0	12.05.2016	Added spot color details and examples

Change history		
v1.6	15.12.2015	Added /ChannelType
v1.5	23.02.2015	Added /CalibrationGroupName, spot color and DeviceN capability
v1.4	14.05.2013	Updated for the HMR v10.0 and HHR v4.0
v1.2.1.12	18.09.2012	Updated info_dictionary
v1.2	27.01.2012	Updated /HWResolution
v1.1	01.09.2011	Added /ProcSetVersion information
v1.0	02.06.2011	New Document

Copyright and Trademarks

Push Calibration

Version 1.7: May 2016

Part number: Hqn081

Document issue: 120

The ECI and FOGRA ICC color profiles supplied with this Harlequin RIP are distributed with the kind permission of the ECI (European Color Initiative) and FOGRA respectively, and of Heidelberger Druckmaschinen AG (HEIDELBERG).

The IFRA ICC profiles supplied with this Harlequin RIP are distributed with the kind permission of IFRA and of GretagMacbeth.

Harlequin and the Harlequin RIP are trademarks of Global Graphics Software Ltd, which may be registered in certain jurisdictions. Harlequin ColorPro, Harlequin Dispersed Screening (HDS), Harlequin Precision Screening (HPS), TrapPro, SetGold, SetGoldPro, Harlequin MultiRIP, Harlequin Host Renderer, Harlequin Parallel Pages, Harlequin VariData, Harlequin Contour Processor and the Harlequin Screening Engine are all trademarks of Global Graphics Software Ltd. Other brand or product names are the registered trademarks or trademarks of their respective holders.

TrueType is a registered trademark of Apple Computer, Inc.

Microsoft, Win32, Windows, Windows NT, Windows Server, Windows Vista, Windows 7, Windows 8 and WinFX are either registered trademarks or trademarks of the Microsoft Corporation in the United States and/or other countries.

PANTONE® Colors displayed herein may not match PANTONE-identified standards. Consult current PANTONE Color Publications for accurate color.

PANTONE® and other Pantone trademarks are the property of Pantone LLC. © Pantone LLC, 2014.

Font data copyright 1991 -1995 Linotype Hell Corp.

Fonts copyright © 2000-2004 Timo Lehtinen. All Rights Reserved. http://www.timolehtinen.com/type/.

Adobe, Adobe Photoshop, Adobe Type Manager, Acrobat, Display PostScript, Adobe Illustrator, PostScript, Distiller and PostScript 3 are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries which may be registered in certain jurisdictions.

Portions include software licensed under the following terms:

OpenSSL - general purpose cryptography library

Copyright © 1998-2011 The OpenSSL Project. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

4. If you include any Windows specific code (or a derivative thereof) from the apps directory (application code) you must include an acknowledgement: "This product includes software written by Tim Hudson (tjh@cryptsoft.com)".

THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The licence and distribution terms for any publicly available version or derivative of this code cannot be changed. i.e. this code cannot simply be copied and put under another distribution licence [including the GNU Public Licence.

ICU - IBM library providing Unicode and Globalization support

Copyright © 1995-2003 International Business Machines Corporation and others All rights reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, provided that the above copyright notice(s) and this permission notice appear in all copies of the Software and that both the above copyright notice(s) and this permission notice appear in supporting documentation.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Except as contained in this notice, the name of a copyright holder shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization of the copyright holder.

Expat - XML parser library

Copyright © 1998, 1999, 2000 Thai Open Source Software Center Ltd and Clark Cooper

Copyright © 2001, 2002 Expat maintainers.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

---

pthreads-win32 - a POSIX threads library for Microsoft Windows

This file is Copyrighted

This file is covered under the following Copyright:

Copyright © 2001,2006 Ross P. Johnson All rights reserved.

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Pthreads-win32 is covered by the GNU Lesser General Public License

Pthreads-win32 is open software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation version 2.1 of the License.

Pthreads-win32 is several binary link libraries, several modules, associated interface definition files and scripts used to control its compilation and installation.

Pthreads-win32 is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

A copy of the GNU Lesser General Public License is distributed with pthreads-win32 under the filename: COPYING.LIB

You should have received a copy of the version 2.1 GNU Lesser General public License with pthreads-win32; if not, write to:

Free Software Foundation, Inc.

59 Temple Place

Suite 330

Boston, MA02111-1307

USA

The contact addresses for pthreads-win32 is as follows:

Web:http://sources.redhat.com/pthreads-win32

Email: Ross Johnson

Please use: Firstname.Lastname@homemail.com.au

Pthreads-win32 copyrights and exception files

With the exception of the files listed below, Pthreads-win32 is covered under the following GNU Lesser General Public License copyrights: threads-win32 - POSIX Threads Library for Win32:

Copyright© 1998 John E. Bossom

Copyright© 1999,2006 Pthreads-win32 contributors

The current list of contributors is contained n the file CONTRIBUTORS included with the source code distribution. The current list of CONTRIBUTORS can also be seen at the following WWW location:

http://sources.redhat.com/pthreads-win32/contributors.html

Contact Email: Ross Johnson

Please use: Firstname.Lastname@homemail.com.au

These files are not covered under one of the Copyrights listed above:

COPYING

COPYING.LIB

tests/rwlock7.c

This file, COPYING, is distributed under the Copyright found at the top of this file. It is important to note that you may distribute verbatim copies of this file but you may not modify this file.

The file COPYING.LIB, which contains a copy of the version 2.1 GNU Lesser General Public License, is itself copyrighted by the Free Software Foundation, Inc. Please note that the Free Software Foundation, Inc. does NOT have a copyright over Pthreads-win32, only the COPYING.LIB that is supplied with pthreads-win32.

The file tests/rwlock7.c is derived from code written by Dave Butenhof for his book 'Programming

With POSIX® Threads'. The original code was obtained by free download from his website http://home.earthlink.net/~anneart/family/Threads/source.html and did not contain a copyright or author notice. It is assumed to be freely distributable.

In all cases one may use and distribute these exception files freely. And because one may freely distribute the LGPL covered files, the entire pthreads-win32 source may be freely used and distributed.

General Copyleft and License info

For general information on Copylefts, see: http://www.gnu.org/copyleft/

For information on GNU Lesser General Public Licenses, see: http://www.gnu.org/copyleft/lesser.html http://www.gnu.org/copyleft/lesser.txt

---

zlib - general purpose compression library

Copyright © 1995-2013 Jean-loup Gailly and Mark Adler

This software is provided 'as-is', without any express or implied warranty. In no event will the authors be held liable for any damages arising from the use of this software.

Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.

3. This notice may not be removed or altered from any source distribution.

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR

CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

Libpng - PNG (Portable Network Graphics) image library

libpng version 1.2.6, December 3, 2004, is Copyright © 2004 Glenn Randers-Pehrson, and is distributed according to the same disclaimer and license as libpng-1.2.5 with the following individual added to the list of Contributing Authors

Cosmin Truta

libpng versions 1.0.7, July 1, 2000, through 1.2.5 - October 3, 2002, are Copyright © 2000-2002

Glenn Randers-Pehrson, and are distributed according to the same disclaimer and license as libpng-1.0.6 with the following individuals added to the list of Contributing Authors

Simon-Pierre Cadieux

Eric S. Raymond

Gilles Vollant

and with the following additions to the disclaimer:

There is no warranty against interference with your enjoyment of the library or against infringement.

There is no warranty that our efforts or the library will fulfill any of your particular purposes or needs. This library is provided with all faults, and the entire risk of satisfactory quality, performance, accuracy, and effort is with the user.

libpng versions 0.97, January 1998, through 1.0.6, March 20, 2000, are Copyright © 1998, 1999

Glenn Randers-Pehrson, and are distributed according to the same disclaimer and license as libpng-0.96, with the following individuals added to the list of Contributing Authors:

Tom Lane

Glenn Randers-Pehrson

Willem van Schaik

libpng versions 0.89, June 1996, through 0.96, May 1997, are Copyright © 1996, 1997 Andreas Dilger

Distributed according to the same disclaimer and license as libpng-0.88, with the following individuals added to the list of Contributing Authors:

John Bowler

Kevin Bracey

Sam Bushell

Magnus Holmgren

Greg Roelofs

Tom Tanner

libpng versions 0.5, May 1995, through 0.88, January 1996, are Copyright © 1995, 1996 Guy Eric Schalnat, Group 42, Inc.

For the purposes of this copyright and license, "Contributing Authors" is defined as the following set of individuals:

Andreas Dilger

Dave Martindale

Guy Eric Schalnat

Paul Schmidt

Tim Wegner

The PNG Reference Library is supplied "AS IS". The Contributing Authors and Group 42, Inc. disclaim all warranties, expressed or implied, including, without limitation, the warranties of merchantability and of fitness for any purpose. The Contributing Authors and Group 42, Inc. assume no liability for direct, indirect, incidental, special, exemplary, or consequential damages, which may result from the use of the PNG Reference Library, even if advised of the possibility of such damage.

Permission is hereby granted to use, copy, modify, and distribute this source code, or portions hereof, for any purpose, without fee, subject to the following restrictions:

1. The origin of this source code must not be misrepresented.

2. Altered versions must be plainly marked as such and must not be misrepresented as being the original source.

3. This Copyright notice may not be removed or altered from any source or altered source distribution.

The Contributing Authors and Group 42, Inc. specifically permit, without fee, and encourage the use of this source code as a component to supporting the PNG file format in commercial products. If you use this source code in a product, acknowledgment is not required but would be appreciated.

Little cms - color management engine

Little cms Copyright © 1998-2004 Marti Maria

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

---

sRGB Profile Licensing Agreement:

To anyone who acknowledges that the files "sRGB_IEC61966-2-1_noBPC.icc" and "sRGB_IEC61966-2-1_withBPC.icc" are provided "AS IS" WITH NO EXPRESS OR IMPLIED WARRANTY, permission to use, copy and distribute these file for any purpose is hereby granted without fee, provided that the files are not changed including the HP copyright notice tag, and that the name of Hewlett-Packard Company shall not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission. Hewlett-Packard Company makes no representations about the suitability of this software for any purpose.

---

libtiff

Version string is 20060323

Copyright © 1992-1997 Sam Leffler

Copyright © 1992-1997 Silicon Graphics, Inc.

Permission to use, copy, modify, distribute, and sell this software and its documentation for any purpose is hereby granted without fee, provided that (i) the above copyright notices and this permission notice appear in all copies of the software and related documentation, and (ii) the names of Sam Leffler and Silicon Graphics may not be used in any advertising or publicity relating to the software without the specific, prior written permission of Sam Leffler and Silicon Graphics.

THE SOFTWARE IS PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EXPRESS, IMPLIED OR OTHERWISE, INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

IN NO EVENT SHALL SAM LEFFLER OR SILICON GRAPHICS BE LIABLE FOR ANY SPECIAL, INCIDENTAL, INDIRECT OR CONSEQUENTIAL DAMAGES OF ANY KIND, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER OR NOT ADVISED OF THE POSSIBILITY OF DAMAGE, AND ON ANY THEORY OF LIABILITY, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

US Government Use

Harlequin RIP software is a computer software program developed at private expense. If the Harlequin Host Renderer software is acquired under the terms of a proposal or agreement with the United States Government or any contractor therefor, the software is subject to the following restricted rights notice: "This Software is commercial computer software provided with RESTRICTED RIGHTS. Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013, FAR 52.227-17 Alternate III (g)(3), or subparagraphs (c)(1) and (2) of the Commercial Computer Software -- Restricted Rights at 48 CFR 52.227-19, as applicable, and their successor provisions. Contractor/Manufacturer is Global Graphics Software Incorporated, Waltham, MA 02451."