Harlequin VariData™

Technical Note Hqn087

March 2014

GLOBAL GRAPHICS® software ™

1 Introduction

This document provides information about Harlequin VariData™ (internal and external mode) features for the Harlequin MultiRIP and Host Renderer v3.0rx release (and later) along with information about its use and background.

2 Background

The advent of variable data jobs means that many parts of a printing job will remain constant with small parts, such as text, being changed for each print. Thus, time savings can be made by processing the constant areas only once, especially if the constant area is a large graphic. This is the idea behind the Harlequin VariData (HVD) feature. The RIP detects constant areas within a PDF file, retains them and then re-uses them as necessary.

Any PDF file with pages that share raster elements and have marks which change from page to page should be accelerated by this optimization in the RIP. The RIP scans the PDF for such pages, RIPs the shared raster elements once, and then retains them for use on subsequent pages with the same page elements.

2.1 Harlequin VariData

HVD can cache any number of rasters per page in external mode, and one in internal mode and build the final raster from these parts. In addition, it can cope with situations of an imposed flat where several images and text layers are placed on top of each other.

Note: HVD supports pre-imposed PDFs only: the Harlequin RIP's own imposition is incompatible with HVD.

HVD (internal and external mode) is supported in both the Harlequin MultiRIP and the Harlequin Host Renderer.

HVD internal mode (iHVD) is where the combination of cached and uncached elements to form the final page raster is performed within the RIP.

In HVD external mode (eHVD), cached and uncached elements are provided to the OEM's own code outside the RIP, along with metadata defining how to reassemble these elements into final pages.

Note: By design, iHVD is more restricted in which marks it can and cannot cache. Hence, the results of eHVD and iHVD scans may produce different results.

HVD external mode is provided for those customers who have their own method of stitching together the resulting rasters, for example, using hardware acceleration. GGSL do not provide any support for these external raster stitching features. If you do not have your own raster-stitching technology you should use HVD internal mode instead.

Note: HVD external mode is the same as previously described by Global Graphics as "ERR2". Some files such as **swevents.h** still refer to ERR2.

3 How to activate Harlequin VariData

This section describes how to activate HVD (internal and external mode) for both the Harlequin MultiRIP and the Host Renderer.

HVD internal and external mode requires a password.

For the Harlequin MultiRIP open **Harlequin MultiRIP > Configure RIP > Extras** and enter a password for either `HVD - Harlequin VariData (external)` or `HVD - Harlequin VariData (internal)` followed by clicking **OK** twice to save and close the open dialogs.

When using Sentinel LDK security Harlequin VariData is activated with a new product key. When using the Harlequin License Server HVD internal and external mode both require a password.

For the Harlequin Host Renderer add the following to a test configuration file in **/SW/TestConfig**. For example:

```
<<
  /HVDExternal 12345678
  /HVDInternal 87654321
>> setsystemparams
```

where `12345678` and `87654321` is the supplied passwords.

3.1 Harlequin VariData parameters

The section describes the various parameters which can be used with HVD:

`/EnableOptimizedPDFScan`

A boolean which turns HVD on or off.

`/OptimizedPDFScanLimitPercent`

An integer percentage (that is, min. `0`, max. `100`). It controls whether the optimized PDF scan ends early or not. The default setting for `/OptimizedPDFScanLimitPercent` is 10%. If the parameter line is missing the RIP will use this value.

This value is present to avoid the scanning overhead on jobs where the optimization is not going to be effective. After scanning each page, the RIP compares the number of unique pages encountered so far against the total number of pages scanned so far. Once the number of unique (non-shared) pages, expressed as a percentage of the number of pages scanned exceeds the `/OptimizedPDFScanLimitPercent` value, the scan is aborted; an aborted scan means that no caching will take place.

`/OptimizedPDFCacheID`

This key is required in the page feature that enables HVD and is the identifier for the cache implementation to connect to. That is, it identifies the HVD back-end which is to be used to perform raster element caching. It may be one of the built-in values (prefixed with `GG_`) or one defined by a custom OEM created back-end cache implementation. This could be a plugin, an HHR back end (see HVDNONE or HVDRAW) or internal HVD (GGIRR).

The value used should be a unique string, and it is recommended that it be prefixed with an abbreviation that identifies your company or product. For example, OEM_HHR_PDF_HVD.

Every page element, whether it is variable data or fixed data, has an ID. If the data is variable it has a usage count of 1. A page is always built from a series of raster "chunks" that have various IDs, and it is the SWEVT_RR_PAGE_DEFINE event which tells the code which IDs go where.

/OptimizedPDFCompactMemory

This option can be used for both eHVD and iHVD.

This is a boolean switch controlling memory behavior and decides between the option of using less memory or faster operation. When this flag is true, the RIP discards some bits of scanner state earlier than it otherwise would. This leads to a lower overall memory usage for many jobs at the possible expense of less flexible mark coalescing and therefore more raster elements per page on average. Also there is a scanning speed penalty for this lowering of memory consumption. The default is false.

/OptimizedPDFCoalesceLimitPercent

This option can only be used for eHVD.

This option provides less scanning flexibility and controls the way marks on the page are coalesced into raster elements. If this parameter is negative (the default), there is no restriction: every eligible mark is coalesced. If it's zero, no coalescing at all takes place. If it's positive, coalescing is allowed. The default is -1

/OptimizedPDFCoverageLimitPercent

This option can only be used for eHVD.

This option only takes effect when OptimizedPDFExternal is true. When greater than zero, the RIP calculates the total area of the bounding boxes of the raster elements it has scanned on every page. When this area expressed as a percentage of the overall page area exceeds the value of this parameter, all marks for the page in question are coalesced into a single raster element. For example, if the scanner calculates that there are three elements on a given page and each is the same size as the page itself, the calculated total area is 300% of the page area and so any value below 300 for this parameter would result in the page being limited to a single raster element (and thus the coverage would be reduced to 100%). The default is 0 (no limit).

/OptimizedPDFExternal

This controls whether external or internal HVD is used. When this PDF parameter is `true`, external HVD is used and the RIP assumes that the OEM's own code has a method to store and combine the output rasters to form a final page.

3.2 Internal mode

To turn on the HVD internal mode optimization, use the following PostScript language fragment:

```
<< /EnableOptimizedPDFScan true
   /OptimizedPDFScanLimitPercent
   /OptimizedPDFCompactMemory false
   /OptimizedPDFCacheID (GGIRR)
>> setpdfparams
```

In the Harlequin MultiRIP use this fragment in a page feature. For the Host Renderer use the fragment as part of a test configuration in the **/SW/TestConfig** folder.

Note: Adding the PostScript language fragment to **HqnOEM**, so that it is used all the time, would potentially invoke a performance penalty. This is because the scan, which is looking for shared raster elements, does takes some computational effort, and so should only be turned on in a workflow where such shared raster elements are known to occur, or are at least likely.

3.3 External mode

To turn on the HVD external mode optimization, use the following PostScript language fragment in a page feature:

```
<</EnableOptimizedPDFScan true
  /OptimizedPDFScanLimitPercent 25
  /OptimizedPDFCacheID (OEM_HHR_PDF_HVD)
  /OptimizedPDFCompactMemory false
  /OptimizedPDFCoalesceLimitPercent 150
  /OptimizedPDFCoverageLimitPercent 250
  /OptimizedPDFExternal true
>> setpdfparams
```

where `OEM_HHR_PDF_HVD` is the HVD back-end to be used to perform raster element caching as described in "Harlequin VariData parameters" on page 4.

For the Host Renderer there are two HVD external mode back-ends available:

`HVDNONE`

This has a `/OptimizedPDFCacheID` value of `GG_HHR_HVDNONE_ERR2`. `HVDNONE` discards the raster data, and generates no files.

`HVDRAW`

This has a `/OptimizedPDFCacheID` value of `GG_HHR_HVDRAW_ERR2`. `HVDRAW` generates a raw file for each element, named **<id>.raw**, and an XML file containing the page and element info, named **<job>.pages.xml**.

4 Harlequin VariData API—external mode

The RIP supplies page descriptions (“pages in flight”) and rasters for sub-page collections of graphical elements to OEM code through the output plugin API.

There are more details on the exact nature of the API in the file **swevents.h**.

The following should be added to the **SW/Sys/ExtraPageDeviceKeys**:

```
/OptimizedPDFId ()
/OptimizedPDFUsageCount 0
```

The OEM plugin is expected to implement plugin parameters that will store and combine the raster elements appropriately.

Implement the following two page device parameters:

```
{STIO_INLINE_STRING, szNull, szNull, (uint8 *)"/OptimizedPDFId",
  Stio_Offset(TestDevParams, OptimizedPDFId[0]), 33, 0, 0, 0 },
{STIO_INT, szNull, szNull, (uint8 *)"/OptimizedPDFUsageCount",
  Stio_Offset(TestDevParams, OptimizedPDFUsageCount), 0, 0, 0,
MAXINT32 },
```

Having done this, the RIP will pass these two parameters through to the plugin for each raster element. The parameters are:

`OptimizedPDFId`

This is either an empty string, or a 32-character hex string, representing a unique ID for this raster element. The IDs tie in with the IDs in the `SWEVT_RR_PAGE_DEFINE` event. Thus, plugin developers should store what is in the `SWEVT_RR_PAGE_DEFINE` event as a table. As rasters arrive they are associated with a slot in the table with the same ID and the next time it is referenced it is simply played back.

`OptimizedPDFUsageCount`

This integer indicates how many times this raster element is used. The OEM code can use this to determine how many times a page uses this raster element, and delete the raster element once all of the pages using it have been processed. The OEM code needs to note all of the pages and raster elements associated with a job, and remove them all once the job is complete or fails.

When outputting separations, the IDs are the same across all separations. It is up to the OEM code to identify the right separations to combine.

Sub-page raster elements are delivered for all parts of every page where HVD—external mode is active. The variable data elements of each page are supplied with an `OptimizedPDFId` as well as those elements which are intended to be cached. The `OptimizedPDFUsageCount` is `1` for variable data parts of the page.

The OEM code receives a series of PGBs with associated IDs that it must store for later retrieval.

The delivery of raster to the OEM code, however, provides no information about how to assemble the raster elements into final pages for output, and because of this an API is supplied which contains the following parts:

- "API discovery" on page 9
- "Raster element handling " on page 10
- "Pages in flight " on page 11

4.1 API discovery

There is a layer in the plugin framework (PFI) which gives plugins access to the RIP functionality known as RDR (ROM Data Resource) and Events (`SWEvent`). The HVD external mode interface is reliant on the latter. Events provide a way of communicating across modular boundaries in an abstract and thread-safe way. Given the way they are designed, they provide an ideal way to implement an extensible and customizable API. Default event handlers can be overridden, augmented or left alone to do their job.

In the case of HVD external mode events are used to track and synchronize the various elements involved in caching sub-page raster elements and uses them to construct the representation of the page in a RIP plugin and/or other modules downstream in the customer's workflow.

A bootstrap phase occurs during initialization during which API discovery takes place. That is, once the PFI is in place, the plugin uses the RDR mechanism to discover RIP APIs in which it is interested: APIs themselves are RDR instances.

An example for HVD follows:

```
sw_rdr_api_20110201 *     rdr_api     = NULL ;
sw_event_api_20110330 *   event_api   = NULL ;

  if ((rdr_api = (sw_rdr_api_20110201*)PlgFindAPI(RDR_API_RDR,
20110201)) != NULL) {
  if ((event_api = (sw_event_api_20110330*)PlgFindAPI(RDR_API_EVENT,
20110330)) != NULL) {
```

4.2 Raster element handling

There is a set of events for handling sub-page raster elements.

`SWEVT_RR_ELEMENT_DEFINE`

Triggered by the RIP.

This event has two parameters; the ID to attach to a given element and its bounding box. No raster data accompanies it as this stage: the event is generated by the RIP during its scanning phase.

`SWEVT_RR_ELEMENT_UPDATE_HITS`

Triggered by the RIP.

Updates the cache implementation's hit count (usage count) for the element in question. There are two parameters: the hit count delta and a flag, `raise` which indicates whether this is an increase or a decrease in the hit count.

What the OEM code does with the hit count is private to the cache implementation, but the default might be to delete the raster straight away for a cache intended for single jobs. An OEM implementation may choose to ignore the hit count in the raster metadata maintained by the RIP.

A simple OEM implementation may choose to read `OptimizedPDFUsageCount` from the supplied rasters and treat those rasters with a value of `1` in that field as being variable and suitable for early purging.

4.3 Pages in flight

This event is used to transfer information regarding pages in flight to the OEM code:

`SWEVT_RR_PAGE_DEFINE`

: Triggered by the RIP.

: The `SWEVT_RR_PAGE_DEFINE` event passes an array of pages (currently every page in the job), and is generated by the RIP during the scanning process; this event informs the client code of the pages and their contents, in terms of raster element handles.

: These must be replayed in turn by the client code in order to stitch together the final page. If the client code knows that all raster elements are already to hand, it can begin page output immediately.

: The page definition provided by the RIP must be stored by the OEM code until it has been used as it will not be re-sent by the RIP.

5 Calling sequence

The sequence of events and raster deliveries in HVD external mode is deliberately not formally defined. It is highly likely to change between HMR 3.0 and HMR 4.0. OEMs are, therefore, strongly recommended not to rely on any specific ordering in their code.

However, some generalities can be described that may help in understanding the intended use of the various events.

The SWEVT_RR_ELEMENT_DEFINE and SWEVT_RR_PAGE_DEFINE events occur during the scanning process, as commonality in page elements is identified. In HMR 3.0 the whole requested page range is scanned before the first page is fully interpreted, although that situation is not guaranteed to continue in further versions and should not be assumed in OEM code.

The OEM code can make a decision as to whether it has everything it needs to print a page by reviewing a page in-flight definition (as provided by the SWEVT_RR_PAGE_DEFINE event) and checking whether it holds raster data for all of the sub-page elements required by that page.

In general, event handlers should assume that any memory referenced by the messages, including the messages themselves, belongs to the module originating the event. No assumptions about the lifetime of such memory should be made beyond the scope of the event handler itself: if client code needs to reference such data, it must make a copy.

In order to make a real page from the pagebuffers it gets sent via the HVD external mode interface, a cache implementation (the client code), should play back each raster element in turn that is included for the page in the SWEVT_RR_PAGE_DEFINE event. The order is essential: the zeroth element in the RR_PAGE_DEFINE_PAGE "elements" array should be played back first and so on.

The sequence is as follows:

1. The RIP scans the requested page range of the PDF file. The first event it sends is SWEVT_RR_PAGE_DEFINE with, currently, one entry per page in the whole of the requested page range. Note that throughout the HVD external mode interface, the page_num fields are indices into the pages array—that is, they represent an ordinal within the page range requested at pdfexecid time. They should not be assumed to bear any relation to PDF page labels or page numbers (one logical page usually is made up of several pagebuffers given that it is split into page elements).

2. For each page in turn, the RIP issues a SWEVT_RR_ELEMENT_QUERY event for each raster element on the page. For each raster element in turn, if it is not already in the cache, the RIP sets the OptimizedPDFId and OptimizedPDFUsageCount PGB parameters and outputs a PGB. The same mechanism is used for elements intended to be cached as for those which are variable data and appear only once—the only difference is the

usage count. The single parameter set by the RIP is the desired element ID, and the cache implementation should set the opaque handle in the event message to a non-null value to signify it has the element in its cache. If not, it should set the handle to null which signifies the RIP should send the element. If that is the case, the RIP will issue a `SWEVT_RR_ELEMENT_PENDING` event which means that the client code should remember the fact that the RIP has promised the element (though it may not yet have arrived), so that subsequent queries are not answered with a null handle, leading to redundant sending of the same element. Once the element is in place, the RIP issues a `SWEVT_RR_ELEMENT_LOCK`. This requests that the cache instance does not purge the given element until it sees the corresponding `SWEVT_RR_ELEMENT_UNLOCK` event. `SWEVT_RR_ELEMENT_UNLOCK` unlocks a raster element, and if no other locks apply, it is eligible for purging at the discretion of the cache implementation.

3. The RIP issues a `SWEVT_RR_PAGE_READY` event when all raster elements have been exported for the page in question. This means that the cache implementation can go ahead and stitch the rasters to complete the page.
4. When the plugin is finished the cache implementation *must* send a `SWEVT_RR_PAGE_COMPLETE` event to signal that the RIP is free to tidy up after the page, including sending a disconnect event if `SWEVT_RR_PAGE_COMPLETE` events for all of the pages have been received.

Note: `SW_ERR_ELEMENT_UPDATE` is now split into `SWEVT_RR_ELEMENT_UPDATE_HITS` and `SWEVT_RR_ELEMENT_UPDATE_RASTER`. `SWEVT_RR_ELEMENT_UPDATE_RASTER` is only used in HVD internal mode and is not relevant for HVD external mode.

The `SWEVT_RR_ELEMENT_DEFINE` events are issued for all elements needed for the current job, one after the other, with bounding boxes already complete. This will not be the case in the future where the `DEFINE` will happen, for instance, before the bounding box is known. An `UPDATE` event will supply such data subsequently.

There may be elements which legitimately have degenerate bounding boxes (`x1 > x2` or `y1 > y2`). These represent marking operators in the content stream which for some reason did not lead to a mark on the page. They may be clipped out, off the page, or otherwise empty. Caching an empty raster for these elements is still worthwhile because it means that the RIP can avoid interpreting the marking operators each time the page is played back.

5.1 Cache management

When using HVD external mode it is the responsibility of the OEM code to manage whatever caching of sub-page rasters is required. The cache implementation may choose to make use of the `OptimizedPDFUsageCount` value from each raster delivered and/or the hit count managed through the `SWEVT_RR_ELEMENT_UPDATE_HITS` event. Alternatively, it may choose to follow some other strategy.

To avoid stalling the RIP, the OEM code must copy raster data from the buffers to the plugin as rapidly as possible.

If the cache implementation deleted a cached raster before the next time the RIP queries for it, the RIP will regenerate the raster.

5.2 Raster dimensions

All rasters delivered will have nominally the full dimensions of the page, but a bounding box is supplied for each in the raster metadata supplied by the `SWEVT_RR_ELEMENT_DEFINE` event. Empty bands above and below the area actually marked in each raster may be suppressed with `TrimPage` as usual. White space to the left and right of that marked area can be easily skipped in the cache implementation using the bounding box.

6 Restrictions

Please note the following points:

- When the Recombine feature of the RIP is used HVD is disabled and a warning message is displayed:

  ```
  %%[ Warning: Recombine enabled - disabling Harlequin VariData ]%%
  ```

- When the TrapPro feature of the RIP is used HVD is disabled and a warning message is displayed:

  ```
  %%[ Warning: TrapPro enabled - disabling Harlequin VariData ]%%
  ```

- Some jobs, including some variable-data jobs, do not gain enough performance increase from this optimization to be worthwhile. Typically, jobs which fall into this category are those which have short interpretation times and long rendering times—which usually means the output format is a large one like CMYK composite at a high resolution.
- HVD is not compatible with the use of any in-RIP imposition code, such as Simple Imposition, that places multiple pages on a sheet, repositions the pages or adds marginalia.
- No element of RIP configuration (such as `pagedevice`), which has any effect on the RIP's output may be modified during an invocation of `pdfexec` or `pdfexecid`.

7 Known issues

- HVD internal mode uses a very large amount of memory, which gets larger as the bit-depth, page size and number of colorants rises, as well as the number of rasters retained. Therefore, using PCs of a lower specification will cause VM errors. This cannot be reproduced on all PCs.
- By not using subset fonts or by using the same subset fonts on every page you may get faster output with HVD enabled.

8 Document history

Change history		
v3.0	20.08.2012	New Document
v3.1	07.03.2012	Updated
V3.2	19.03.2014	Added new parameters