Ultra-fast PDF production aids server scalability and provides greater reliability.

1. Printing to memory

PDFDevice has always been the fastest way[1] to produce smaller and more accurate representations[2] of Omnis reports, but PDFDevice has become even faster with the introduction of memory printing in version 3. By directly printing to a binary variable, one can produce PDF data at lightning speed and return it to the client in PDF mime format without ever touching the hard-disk.

This is made possible by redirecting output to the method of the current remote task using the $ctask notation. The method will receive an output ID which can be used with the external function "PDF Device.$getmemoryoutput(id,binary_var)". All that remains to be done is to make an ultra-thin call into the task from the JS Client to fetch the PDF data for display. This can be achieved with the $cinst.$showurl() notation.

Your remote task's $construct method may look like this …

```
;  tell the device the notation to our method that is to receive the PDF binary data
Do $cdevice.$assign(kDevPdf)
Do $cdevice.$setparam(kDevPdfFileName,"$ctask.$pdfoutput(#)")
;  external devices can access $ctask thus preventing threading issues !!!

;  print the report
Set report name repPDFDevice
Print report {*} ; our $pdfoutput will be called at this point
;  return the mime data ($pdfmime is our own method for producing the PDF mime)
Quit method $cinst.$pdfmime
```

The method $pdfoutput fetches the binary PDF data …

```
;  this function is called by PDF device when the PDF data is complete
;  we can than fetch the PDF binary and store it, so that rtGetPDF can retrieve it
Do PDF Device.$getmemoryoutput(pID,tvPdfBin)
```

The method $pdfmime would convert and return the PDF binary to PDF mime data …

```
;  this function is called to return PDF data in mime format to the client
;  return binary data of PDF document as mime data asset to client
Begin text block
Text: content-type: application/pdf (Carriage return,Linefeed)
Text: content-length: [binlength(tvPdfBin)] (Carriage return,Linefeed)
Text: content-disposition: filename="report.pdf" (Carriage return,Linefeed)
```

[1] PDFDevice's powerful compression of content and direct conversion of Omnis high-level printing objects and optimised C++ code combine to make PDFDevice many times faster than Omnis' own PHP based PDF generation and third party solutions such as Bullzip PDF Printer. PDFDevice also produces smaller files that are faster to return to the client. In section 2 we have listed our performance comparison results between PDFDevice, OmnisPDF and the Bullzip PDF printer driver.

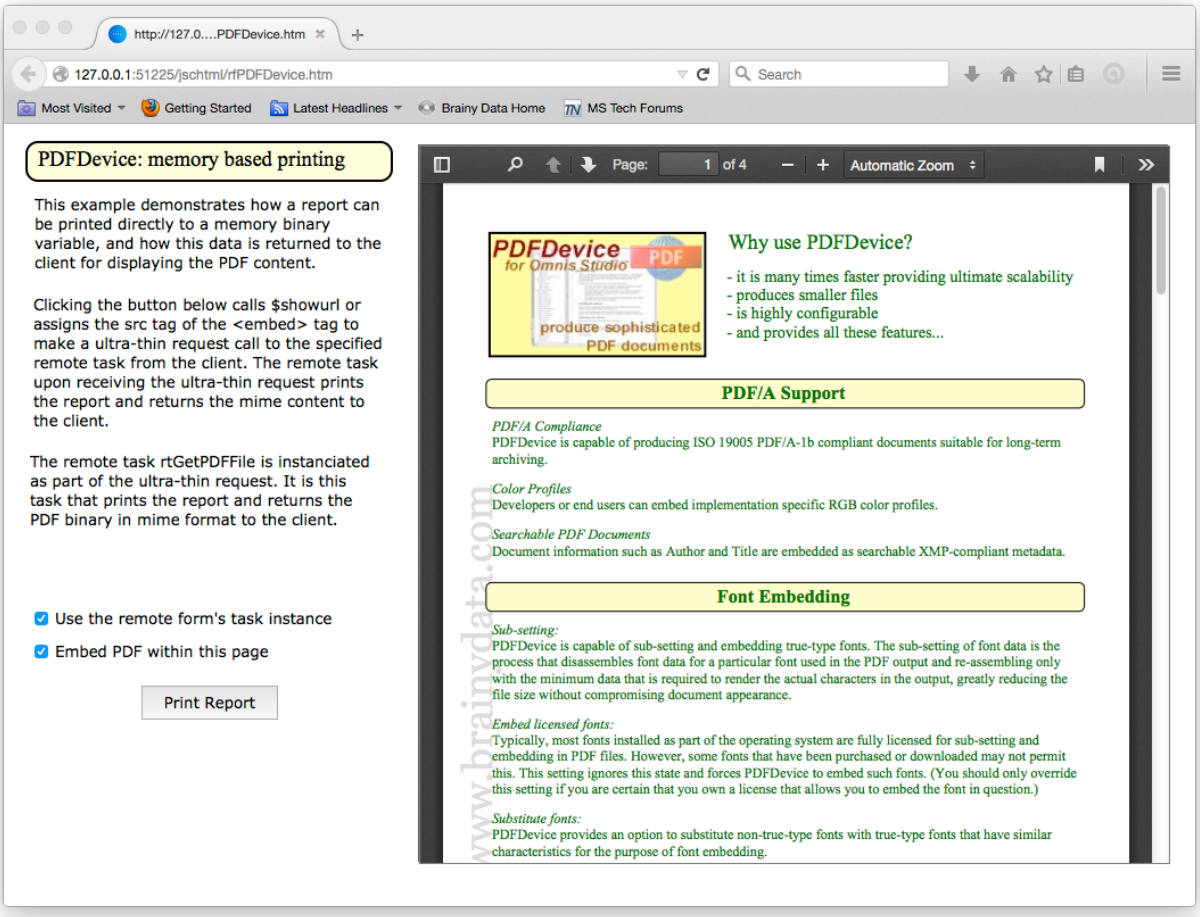
[2] See customer feedback in section 3

```
Text:  (Carriage return,Linefeed)
End text block
Get text block htmlHeader
Calculate bin as bytecon(chartoutf8(htmlHeader),tvPdfBin)
Quit method bin
```

To make an ultra-thin call you can execute $cinst.$showurl(url) where url contains the URL for the ultra-thin call, which may look like this when testing locally.

```
http://127.0.0.1:52736/ultra?
OmnisServer="127.0.0.1:52736"&OmnisLibrary="myLibrary"&OmnisClass="myRemoteTask"
```

We have provided an example library that demonstrates the above which can be downloaded by clicking Download.



The example demonstrates how PDF content can be displayed within a new tab using $cinst.$showurl(), or alternativly how the PDF content can be displayed within your remote form using the generic HTML control in conjunction with an embed tag.

By default, the example uses a separate remote task for printing and returning the PDF data (see rtGetPDFFile) which requires an additional connection and thus license. However, it is possible to direct the ultra-thin request to the remote task instance that instantiates the remote form. This requires the report to be returned by the task's $event method when receiving an evPost event. The ultra-thin url must include the task's $connectionid, and the task's $canclose method must return kFalse, immediately after the evPost event. This approach is also demonstrated in the example (see rtPDFDevice).

2. Performance Comparisons

The following performance tests where carried out using Omnis Studio 6.1.2 (64bit), PDFDevice 3.2.1 (64bit) and Bullzip 10.12.2363 on a 8 Core Windows 7 system. The tests measured the total time taken to execute our test code and to print the report. Consequently, the difference of the performance is diluted by the time it takes Omnis to execute the Omnis code and to produce the report data prior to sending it to the output device. This means that performance differences observed between the different output methods are in reality greater than they are reported here as they were diluted by the extra overhead.

In our test design, we also considered that both Bullzip and the Omnis PDF device return control to Omnis prior to the PDF file being completed. We therefore made three comparisons. Comparison (a) simply measured the time it took for the test to complete regardless of wether the files had all been written, comparison (b) entered a loop after the last file was printed to ensure the final file existed before measuring the time, and comparison (c) entered a loop to test the existence of each file before the next report was printed.

Each test run printed 100 reports and the total time was used to calculate the mean-time to print a single report which is displayed below. We tested three different types of reports. Report (1) consisted of four pages of plain text, background boxes and logos; report (2) consisted of a 1MB CS24 Photograph and report (3) consisted of eighteen pages of text and ten screenshots of small to medium size.

As well as testing the performance of each output device, we also recorded the size of the PDF files.

Report (1) test results - 4 pages of text, background boxes and a logo

Output Device	tics/report (a)	tics/report (b)	tics/file (c)	size in KB
PDFDevice to disk	2.32	n/a	n/a	97
PDFDevice to memory	2.18	n/a	n/a	97
OmnisPDF	7.44	8.68	20.80	130
Bullzip	15.09	15.82	73.18	93

Report (2) test results - 1MB high-res CS24 photograph

Output Device	tics/report (a)	tics/report (b)	tics/file (c)	size in KB
PDFDevice to disk	17.75	n/a	n/a	349
PDFDevice to memory	17.33	n/a	n/a	349
OmnisPDF	16.30	17.09	59.93	9286
Bullzip	31.78	33.81	150.17	367

Report (3) test result - 18 pages of text and 10 screen-shots

Output Device	tics/report (a)	tics/report (b)	tics/file (c)	size in KB
PDFDevice to disk	18.64	n/a	n/a	360
PDFDevice to memory	18.09	n/a	n/a	360
OmnisPDF	36.62	37.42	81.27	457
Bullzip	55.75	59.52	142.99	441

Discussion

In all of the tests, printing via PDFDevice and OmnisPDF substantially outperformed printing via Bullzip PDF Printer in terms of speed. This was expected as generating PDF through printer drivers involves additional steps and conversions. Both PDFDevice and OmnisPDF plug into the Omnis print manager and convert Omnis report objects directly to PDF. However, printing via PDFDevice overall outperformed printing via OmnisPDF and Bullzip in terms of speed and file size.

Note: *For server implementations, timings (a) or (b) are probably the most relevant, as these timings account for the benefit provided by multi-threading servers. Timing (c) will be more relevant to fat-client implementations that have to wait for each file to be completed before printing the next file. Furthermore, report (1) will be the most relevant test for Omnis server implementations. Consisting of a few pages of text and background boxes as well as company logos, this kind of document will be the most likely type of document that will require scalability.*

Report (1) test: Considering only timings (a) and (b), printing via PDFDevice was about 3 to 4 times faster than printing via OmnisPDF and 6 to 7 times faster than printing via Bullzip. The file produced by PDFDevice was also 25% smaller than that produced by OmnisPDF, although the file produced by Bullzip was fractionally smaller still.

Report (2) test: Considering only timings (a) and (b), both printing via PDFDevice and OmnisPDF was fairly evenly matched in terms of speed (printing via OmnisPDF was fractionally faster), but the file produced by PDFDevice was 26 times smaller compared to the files produced by OmnisPDF! The size of the file will impact how quickly data is returned to the client. We believe OmnisPDF produces such large files when dealing with photographs because it uses run-length encoding without compression. In the tests, PDFDevice used 50% JPEG compression which is all-round superior and which results in very little or no visible loss of quality.

Report (3) test: Printing via PDFDevice outperformed printing via OmnisPDF and Bullzip in both speed and file size. Considering only timings (a) and (b), printing via PDFDevice was about twice as fast as printing via OmnisPDF and three times as fast as printing via Bullzip. The file produced by PDFDevice was also about 20% smaller compared to the files produced by OmnisPDF and Bullzip.

PDFDevice and printing to memory: There appears to be only a small difference in speed between PDFDevice printing to a disk compared to printing to memory. However, it must be considered that when printing to memory the document data is held in memory ready to be returned to the client, whereas when returning a disk file to the client, the file has still to be read into memory at some point. Thus the performance benefit for the server when printing direct to memory is probably greater than is apparent from the figures above.

Timing (c): As already mentioned, timing (c) will be most relevant to fat-client implementations that have to consecutively print and store PDF files and are thus unable to take advantage of multi-threading. As the figures show, PDFDevice vastly outperforms both OmnisPDF and Bullzip in such conditions. Depending on the report used, PDFDevice is about 9 to 37 times faster compared with printing via OmnisPDF and 33 to 68 times faster when compared with printing via Bullzip.

Conclusion

The above figures suggest that Omnis in conjunction with PDFDevice can produce 25 to 27 reports of type (1) per second, compared with only 8 reports per second when using OmnisPDF, and fewer still when printing via Bullzip. Thus PDFDevice provides the greatest scalability for the most common type of report, which is especially important to server implementations.

3. Customer Feedback

Below we have listed some feedback from an Omnis developer who had started using OmnisPDF and returned to using PDFDevice after encountering issues.

"... I observed speed and size differences ... the main problems were as follows ... OmnisPDF passed control back to Omnis before the PDF creation finished. This meant I was emailing incomplete PDF files. I had to write a routine to check the file was finished before continuing which was a nuisance ... On roll out to clients we had a lot of problems with Macs. Documents which had been printing fine with our Omnis 4 and PDFDevice application kept crashing with message about fonts not being available. This seemed mostly to do with bold, light and other variations on fonts where only the generic font was in the OS Suitcase. Messing with Mac fonts is a nightmare at the best of times and we were wasting hours resolving these problems. By using PDF Device the majority of these problems went away. The output is fine by the way there isn't any unacceptable font substitution ... the overhead of the python folder ... slows down upgrading and I will now remove the files as they won't be needed ..."

Document History

09 June 2015: first publication