Extending the Print Record

With each release of LaserWriter 8 and PSPrinter, there are several new feature requests that can not be fulfilled due to the restrictions of the current 120 byte print record. These requested features usually take the form of “can’t we make ‘x’ sticky to the document.” Due to the small number of free bits in the print record, the answer to this request is no. Other features, such as custom paper sizes and even multiple paper sizes with the same physical size, are restricted due to the print record constraints. This document describes an extended print record to enable the development of these new features.

Applications Need To Change

To maintain compatibility with existing applications, printer drivers that support the extended print record will not grow the print record unless the application indicates that it can handle the larger print records. This means that applications will need to be revised in order to invoke the extended print record features. The new PrGeneral opcode, ‘kExtendPrintRecOp’ (18) is implemented in drivers that support the extensible print record. Applications call this new PrGeneral routine passing in a print record that the driver will mark as extensible. If a printer driver returns an error from the new PrGeneral routine indicating that it doesn’t support extensible print records (OpNotImpl), then the application must ensure that the print record is exactly 120 bytes long.

The easiest manner to revise an application to work with the new extensible print record, is to change calls to PrDefault() to extendPrDefault() and the calls to PrValidate() to extendPrValidate(). These new, extend routines are listed below and should be linked in by the application. In order to make these extend calls, the application must meet the following conditions:

1. The application makes no assumptions about the size of the print record. This means that when writing a print record to a document, the application must be prepared for a print record that is larger than 120 bytes. Similarly, when reading a saved print record, the application must read the entire record even if it is longer than 120 bytes. The application’s internal handling of print records must avoid the use of the constant sizeof(TPrint) when working with extended print records, and can only use the TPrint structure for access to the fields which are documented in Inside Macintosh.

2. The application can not pass the printer driver a fake handle for a print record. In order to grow print records, the extensible drivers will call SetHandleSize(). If the application is using fake handles,then the SetHandleSize call will eventually cause a crash.

3. The application must use the new PrGeneral opcodes before passing a print record to any Printing Manager routine. If you use the sample routines extendPrDefault() or extendPrValidate(), you must call before passing a print record to PrStlDialog(), PrStlInit(), PrJobDialog(), PrJobInit(), PrOpenDoc(), or PrGeneral(). Applications should already be validating print records before making any of these calls. In addition, applications must call extendPrValidate() with both print records before calling prJobMerge(). If you call

prJobMerge with an extended print record and a print record that has not been extended, the results will depend on the driver. Drivers should handle this correctly and return the correct type of destination print record, but an application that depends on this behavior is likely to be disappointed.

An Application’s New Extend Routines

```
#define kExtendPrintRecordOp 18

/* Use this structure when making the 'kExtendPrintRecOp' PrGeneral call.
On entry, 'hPrint' is the handle for a standard or extended print
record. If 'hPrint' is an extended print record, then nothing is
done. If 'hPrint' is a standard print record, then it is extended.
*/

typedef struct{
short iOpCode;
short iError;
long lReserved;
TPrintH hPrint;
}TExtendPrintRecord;

TExtendPrintRecord extend;


Boolean extendPrValidate(THPrint hPrint)
/* The current printer driver’s PrValidate() routine is called.
After the print record ‘hPrint’ is validated or defaulted, this
routine allows the printer driver to mark the print record as
extensible. If the current printer driver does not support an
extensible print record, then this routine makes sure that the
the print record handle, ‘hPrint’, is the handle for a standard
size print record.
*/
{

/* In case ‘hPrint’ is extended and the current printer driver doesn’t
support the extensible print record, we let extendPrintRecord()
truncate the print record. The print record won’t be truncated
if the print record is already the standard size or if the current driver
supports the extensible print record.
*/
extendPrintRecord(hPrint);

/* Call the real PrValidate with a correctly sized print record.
*/
return PrValidate(hPrint);
}

void extendPrDefault(THPrint hPrint)
{

/* The default print record is the standard size for all drivers.
So default the 120 byte record.
*/
SetHandleSize((Handle)hPrint, sizeof(TPrint));
PrDefault(hPrint);

/* Tell the printer driver it is okay to extend this print record.
*/
extendPrintRecord(hPrint);

}

void extendPrintRecord(THPrint hPrint)
{
TExtendPrintRec extend;

/* Call the new PrGeneral opcode to see if the current printer driver
supports extended print records. If the driver does support extended
print records, then it will return noErr, it may also extend the print record
at this time. It will certainly mark the print record as being extensible.
If the driver does not support extended print records, then the driver
will return 'OpNotImpl'.
*/
extend.iOpCode = kExtendPrintRecordOp;
extend.lReserved = 0;
extend.hPrint = hPrint;
PrGeneral(&extend);

/* If the driver fails to make the print record extensible, then we make
sure the print record is the standard 120 bytes.
*/
if(extend.iError) SetHandleSize((Handle)hPrint, sizeof(TPrint));
}
```

Application Benefits

There are three benefits for applications that enable the use of extended print records by a printer driver.

Firstly, when running with an extended print record, a printer driver can add new features to the Page Setup dialog. In LaserWriter 8.4, the printer driver will allow more than two paper sizes with the same physical page size to be listed in the paper popup. Future versions of LaserWriter 8 will enable other features for applications that support extended print records. These features will be attached to the document via the print record, something users will appreciate.

The second use for the extended print record is to communicate information about a user's Page Setup and Print dialog selections. A printer driver that supports the extended print record will, in the future, be able to place information about the user's selections into defined tags in the collection that makes up the trailer of the larger print record. Version 8.4 of the LaserWriter driver will not offer application access to the fields of the extended print record. The API definition and standard tags are currently planned to be released with the 8.4.1 release of the driver.

Driver Implementation

In LaserWriter 8's implementation of extended print records, the standard 120 bytes of a TPrint structure are followed by a four byte signature field with the value 'grow'. This signature is then followed by a flattened Collection as described in *Inside Macintosh, QuickDraw GX, Environment and Utilities, Chapter 5* . If another print driver implements extended print records, it may have a different format for the extended portion of the print record.

Accessing the extended print record fields

The interface for accessing fields that are contained within the extended portion of the print record is not currently defined. The `kGetExtendedPrintRecOp` and `kSetExtendedPrintRecOp` opCodes will be used to access these fields, and documentation describing the structures used by these opCodes will be provided in the form of a Technote in the future.