Technote 1161

Extending the Print Record for LaserWriter 8

CONTENTS

This Technote describes what changes need to be made in your application to support drivers that use extended print records in LaserWriter 8.

Overview

With each release of LaserWriter 8, there are several new feature requests that cannot be fulfilled due to the restrictions of the 120-byte print record. These requests 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 has always been "no." Other features--such as custom paper sizes and even multiple paper sizes with the same physical size--have also been restricted due to the print record constraints.

This method of extending the print record applies only to LaserWriter 8. You should also read the Carbon Printing API to learn how the print record is extended in Carbon. (It's been split into two structures, among other changes.) Using the Carbon APIs is the recommended way to support extended print records. You should use the method described in this Technote for versions of your application that do not support Carbon, since drivers which use the extended print record are available on MacOS 7.6 and later.

By supporting extended print records, applications can/may profit in many/several ways. The first benefit is that the `PrJobMerge` function actually works. This is a significant advantage, since it allows your application to print multiple documents (in response to a `'pdoc'` Apple Event, for example) with a single print dialog, even if the documents use different Page Setups. Secondly, when you use the Extended Print Record, the driver can better support custom paper sizes, or different page sizes that use the same physical paper size. This offers a significant benefit to your users since it allows them to do things like set up a custom paper size for corporate letterhead and save it with their documents in your application.

Back to top

Applications

Applications Need to Change

Many existing applications expect the print record (size) to be 120 bytes. To maintain compatibility with existing applications, printer drivers cannot grow the print record unless the application indicates that it can handle larger print records. As an application writer, you need to first ensure that you don't require the print record size to be exactly `sizeof(TPrint)`. Second, to allow a driver to use an extended print record, you must tell it that you are compatible. 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 the error from the new `PrGeneral` routine that it doesn't support extensible print records (`OpNotImpl`), the application must ensure that the print record is exactly 120 bytes long.

The easiest way to revise an application to work with the new extensible print record is to change calls from `PrDefault()` to `extendPrDefault()` and the calls from `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 must make 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)`, except for drivers which do not support `kExtendPrintRecOp`. Applications can still access fields in the first 120 bytes of the print record, but should be aware that the data within the first 120 bytes may be replicated in an extension to the print record.
2. The applicationcannot pass the printer driver a fake handle for a print record. To grow print records, the extensible drivers call `SetHandleSize()`. If the application is using fake handles, `SetHandleSize()` will cause a crash.
3. The application must call `extendPrDefault()` or `extendPrValidate()` 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()`.

An Application's New Extend Routines

Here are the new routines that applications should use to support the extended print record:

```
#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 driver is asked to extend the print record.
If this fails, the print record handle is sized to the standard
size print record (120 bytes).
The current printer driver's PrValidate() routine is called to
validate the print record whether it has been successfully
extended or not.
*/
{

/*
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, it returns noErr. It marks the print record as
extensible and may also extend it at this time.
If the driver does not support extended print records,
it returns 'OpNotImpl'.
*/
extend.iOpCode = kExtendPrintRecordOp;
extend.lReserved = 0;
extend.hPrint = hPrint;
PrGeneral(&extend);

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

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* : Collection Manager.

Back to top

PrJobMerge

The LaserWriter 8 series of drivers has always offered users more options in the Print Dialog than can be stored in the 120 byte print record. With the introduction of the extended print record, all of the user's print features can now be stored in the larger print record. If an application uses extended print records, `PrJobMerge` can be used to copy all of a user's feature requests into a new print record. In other words, `PrJobMerge` finally works with all of the options from the Print Dialog, beginning with LaserWriter 8.6.

Note:
You must extend both the source and destination print records for `PrJobMerge` to work correctly with extended print records.

Back to top

Summary

There is now a mechanism for print records larger than 120 bytes. Applications need to change to take advantage of this larger print record, but the changes are small. With more extensive changes, applications can control more aspects of printing, such as page orientation, layout, error handling, etc., in a clean manner. That information will be presented in a separate document describing the extended print record data used by LaserWriter 8.

Further References

- *Inside Macintosh* , Imaging With QuickDraw, chapter 9.
- Print Hints: The All-New LaserWriter Driver Version 8.4, *develop* 27.
- Technote 1112: "Introducing the LaserWriter Driver Version 8.5.1"
- *Inside Macintosh* , The Collection Manager

Back to top

Change History

- Originally written in May 1996.
- In April 1999, this document was updated to comply with Technote format, and to include the latest information available.
- In July 1999, this Technote was updated to include sample code.

Downloadables

Acrobat version of this Technote (K).

Binhexed Header Files (147K).

Back to top

To contact us, please use the Contact Us page.
Updated: 5-July-1999

Technotes | Contents
Previous Technote | Next Technote