Technote 1178

LaserWriter 8.7: Scriptable Printing

By Richard Blanchard and Dave Polaschek

Apple Worldwide Developer Technical Support

CONTENTS

Until the introduction of PrintingLib 8.7 (which is included with LaserWriter 8, Version 8.7), it has been impractical for an application to extend the scriptability of the Print Documents Apple event. The application receives the print event, but the printer driver controls and uses the settings for the print job. In addition, the print settings used by each driver have been different. This document details the extensions to the print event that allow scripters to control printing. This document also describes the changes applications and printer drivers must make to support these new scriptable printing features.

The New Print Event

Starting with PrintingLib 8.7, the traditional print event has been extended to include an optional print settings parameter that defines the settings for the print job or jobs, and an optional print dialog parameter that specifies whether the application should display the print dialog. By default, the application should show the Print dialog if the caller doesn't specify a print dialog parameter.

print: Print the specified object(s)

print reference -- Objects to print. Can be a list of files or an object specifier.

[**with properties** print settings] -- The print settings
[**print dialog** boolean] -- Should the application show the print dialog?

The `print settings` parameter allows a scripter to specify the following information:

Class print settings:

Properties:

copies integer [r/o] -- The number of copies of a document to be printed
collating boolean [r/o] -- Should printed copies be collated?
generating job copy boolean [r/o] -- Should a job copy be generated?
generating job log boolean [r/o] -- Should a job log be generated?
log folder alias [r/o] -- The folder in which the job log and copy should be created
starting page integer [r/o] -- The first page of the document to be printed
ending page integer [r/o] -- The last page of the document to be printed
pages across integer [r/o] -- Number of logical pages laid across a physical page

pages down integer [r/o] -- Number of logical pages laid out down a physical page
requested print time date [r/o] -- The time at which the desktop printer should print the document...
queue placement urgent/normal/hold/foreground [r/o] -- ... or, how the printed document should be placed into the desktop printer queue
cover page none/cover page before job/cover page after job [r/o] -- Should a cover page be generated for the job and where should it be placed?
error handling standard/summarized/detailed [r/o] -- How errors are handled
printer features a list of list [r/o] -- For printer specific features

Note:
Only one of **requested print time** or **queue placement** should be specified in a script.

The **printer features** list is designed to be used for extensions that other drivers or future versions of LaserWriter 8 might need. The **printer features** list is printer-driver dependent, or in the case of future versions of LaserWriter 8, may depend on the PostScript Printer Description (PPD) file.

For example, the following script prints three copies of the second page of a document without showing the print dialog:

```
set theDoc to alias "Macintosh HD:ReadMe"
set printConfig to {copies:3, starting page:2, ending page:2}
print theDoc with printConfig without print dialog
```

These optional parameters to the print event thus provide much finer control than was previously possible.

Back to top

Application Changes Required

Applications will need to be revised to support the optional print settings parameter for the print event. While every effort has been made to minimize the number of changes you will need to make to your application, the actual amount of work is dependent upon the current state of the application's printing code. This section steps through the changes needed to support the extended print event. In addition, there is sample code accompanying this Technote based on a SimpleText Sample which shows how to add scriptable printing support to an application.

Download Sample code

1. Update your application's `'aete'` resource to indicate support for the new, optional print parameter and for the print settings class. An example `'aete'` is included with the sample code for this Technote in the file "aete.r"
2. Revise your application to support the extended print record.

   If your application does not already support the extended print record as described in Technote 1161, "Extending the Print Record for LaserWriter 8", you will need to revise it as detailed in that Technote. Sample code for extending the print record is also included with the sample code accompanying this Technote in the "ExtendPrintRecord.c" file.

   We recommend that applications fully support extended print records if possible. However, if changing your application's print code to be fully compatible with the extended print record is too onerous, you can supply a short-term solution by supporting the extended print record only in your scriptable printing code (that is,

within your print documents Apple event handler).

3. Revise the application's print event handler to respond to the optional print settings parameter.

   The optional print settings parameter for the print event uses the key keyAEPropData. An application's print handler retrieves the keyAEPropData parameter and then coerces it to a `THPrint` print manager object. The coercion should normally be handled by calling `AECoerceDesc()`, but, if the coercion handler cannot be found, you can call the printer driver's `PrGeneral` routine directly. This is demonstrated in the sample below, which is included in the "ScriptablePrinting.c" file within the sample code.

Note:
The coercion handler is only loaded if Desktop Printing is enabled. This is because the coercion handler is loaded by the initialization routine for the SettingsLib shared library which is loaded by Desktop Print Spooler.

Note:
You must call `PrOpen()` before calling `AECoerceDesc()`. The reason for this is that `AECoerceDesc()` calls into the current printer driver, which will not be loaded if `PrOpen()` has not been called.

The retrieval and coercion is performed as follows:

```
/* These constants are defined in "PrintAETypes.h" */
#define kPrintRecordAEType  'prec'
#define kPrintSettingsAEType  'prST'

OSStatus getPrintRecordFromEvent(const AppleEvent *inAppleEvent,
        THPrint  *hPrintP)
/* Given a print event Apple Event descriptor inAppleEvent,
 look for the optional print setting parameter.
 If this parameter exists, then coerce it into a print
 record and place the handle to the print record into
 *hPrintP. If the optional parameter does not exist
 or it cannot be converted, then a non-zero error code
 is returned and *hPrintP is set to NULL
*/
{
 OSStatus err = noErr;
 AEDesc  optionalDesc = {};
 AEDesc  printRecordDesc = {};

 AECoercionHandlerUPP handler = NULL;
 long refCon;
 Boolean typeIsDesc;

 PrOpen();
 err = PrError();

 if(err == noErr) {
  err = AEGetParamDesc(inAppleEvent, keyAEPropData,
      typeAERecord, &optionalDesc);
  if (err == noErr) {
   err = AEGetCoercionHandler(typeAERecord,
        kPrintRecordAEType,
        &handler, &refCon,
```

```
        &typeIsDesc, true);

   if (err == noErr) {
    err = AECoerceDesc(&optionalDesc,
        kPrintRecordAEType,
        &printRecordDesc);
   } else if (err == errAEHandlerNotFound) {
    /* If desktop printing is not enabled,
     the handler won't be installed. This
     is not fatal, since we can call the
     driver's PrGeneral call directly.
     It's better to use the coercion handler,
     but if we can't find it, this is a
     good fallback to use
     */
    PrCoerceStruct coerceData;

    coerceData.iOpCode = kPrCoerceOp;
    coerceData.iError = noErr;
    coerceData.lReserved = 0;
    coerceData.fromDesc = &optionalDesc;
    coerceData.toType = kPrintRecordAEType;
    coerceData.toDesc = &printRecordDesc;

    PrGeneral(&coerceData);

    err = coerceData.iError;
   }
  }
 }

 if (err == noErr) {
  OSErr tempErr;

  *hPrintP = printRecordDesc.dataHandle;
  err = HandToHand(hPrintP);
  tempErr = AEDisposeDesc(&printRecordDesc);
  if (err == noErr) err = tempErr;
 } else {
  *hPrintP = NULL;
 }

 err = AEDisposeDesc(&optionalDesc);

 PrClose();

 return err;
}
```

4. If the optional parameter was successfully coerced into a `THPrint` handle, it can now be used to set the print settings for the current print job. The formatting information for the document to be printed comes from the print record stored with this document or from a newly defaulted print record. This formatting information is not altered by the print event. Instead, the print event supplies print time settings that must be merged with the stored formatting information. For example, the document's print record may specify that the document is laid out on a ledger size page in landscape orientation, but it is the print settings print record which sets the pages to be printed and the number of copies. The two print records must be merged. Use the classic Print Manager routine `PrJobMerge` to perform the merge. You then pass the print record that results from `PrJobMerge()` to `PrOpenDoc()`. It is extremely important that you do not store this merged print record with the document, but rather use it only for the duration of the print job.

Note:
Print records that result from `PrJobMerge` or even from `PrJobDialog` hold print time settings. Applications should not store these print records with a document.

```
OSStatus getPrintJobPrintRec(THPrint docPrintRec,
       THPrint settingsPrintRec,
       THPrint *jobPrintRecP)
/* The caller passes in the print record stored with a
 document, docPrintRec, as well as the print record
 obtained from the print event, settingsPrintRec.
 This function creates a new print record that combines
 the formatting information from docPrintRec with
 the print time settings in settingsPrintRec and
 places the new print record into *jobPrintRecP.
 On entry, settingsPrintRec can be NULL in which
 case docPrintRec is duplicated and returned in
 *jobPrintRecP. In either case, if a new print
 record can not be created, this function sets
 *jobPrintRecP to NULL and returns a non-zero error
 code.
*/
{
 OSStatus err = noErr;
 THPrint jobPrintRec = docPrintRec;

 err = HandToHand(&jobPrintRec);
 if ((err == noErr) && (settingsPrintRec != NULL)) {
  /* Both print records must be extended when
   calling PrJobMerge in order to get a full
   merge. The scripted print settings print
   record is extended by the coercion so
   nothing need be done to it here.
  */
  err = extendPrValidate(docPrintRec); // from TN 1161

  if (err == noErr) {
   PrJobMerge(settingsPrintRec, jobPrintRec);
   err = PrError();
  }
 } else {
  jobPrintRec = NULL;
 }

 if (err != noErr) {
  if (jobPrintRec != NULL) {
   DisposeHandle((Handle) jobPrintRec);
   jobPrintRec = NULL;
```

```
  }
 }

 *jobPrintRecP = jobPrintRec;

 return err;
}
```

5. Finally, check for the presence of the optional parameter that specifies whether or not your application should display the print dialog to the user. If this parameter is missing, you should call `PrJobDialog` to display the print dialog.

```
/* This constant is defined in "PrintAETypes.h" */
#define kPrintDialogAEType   'pdlg'

OSStatus getPrintJobShowDialog(const AppleEvent *inAppleEvent,
        Boolean *showDialog)
/* Given a print Apple Event, look for the optional parameter
 saying whether or not the application should show the
 PrJobDialog. While an apple event with the "print settings"
 parameter COULD have most of the items that will be needed
 to print the job, it's possible that the user or scripter
 will  want to present the dialog to handle printer-specific
 settings. An application should not prevent the user from
 doing so, so this function is needed. The application should
 default to showing the dialog if the parameter is not
 specified.
*/
{
 OSStatus err = noErr;
 AEDesc showDesc = {};

 err = AEGetParamDesc(inAppleEvent, kPrintDialogAEType,
      typeBoolean, &showDesc);
 if (err == noErr) {
  *showDialog = **(showDesc.dataHandle);
 }
 if (err == errAEDescNotFound) {
  /* not having the descriptor is okay.
   It just means we default to showing.
  */
  *showDialog = true;
  err = noErr;
 }

 return err;
}
```

Note:
The code described above is contained within the sample code accompanying this document in the "ScriptablePrinting.c" file.

Back to top

Printer Driver Changes Required

This section describes changes required in a printer driver in order to support scripting of print settings. LaserWriter 8.7 is the first printer driver to support scripted print settings, but other printer drivers can add this support by implementing the following recommendations.

1. Implement the new `PrGeneral` call, `kPrCoerceOp`.

The SettingsLib library within PrintingLib 8.7 installs an AppleScript coercion handler capable of coercing a print settings record into a print record.

Note:
The coercion handler is only loaded if Desktop Printing is enabled. This is because the coercion handler is loaded by the initialization routine for the SettingsLib shared library which is loaded by Desktop Print Spooler.

PrintingLib performs this coercion by calling the current printer driver through the `PrGeneral` call. The new `PrGeneral` call, `kPrCoerceOp` (`opcode` 27) takes the following `PrGeneral` parameter structure:

```
/* This structure holds a repackaging of the AppleScript
 coercion handler parameters. See the AppleScript
 documentation for a complete description of the
 fromDesc, toType and toDesc fields. When the print
 system's coercion handler is invoked to convert between
 an AppleScript print settings record and a print record
 (or the other way), the print system invokes the current
 driver's PrGeneral routine with the kPrCoerceOp opcode in
 the following structure. If the printer driver supports
 scripting  then it will convert the input data. If the
 printer driver does not support scripting then it will
 return OpNotImpl and the print system will decide the
 level of support available for the driver.
*/

typedef struct {
 short iOpCode;
 short iError;
 long lReserved;
 const AEDesc *fromDesc;
 DescType toType;
 AEDesc *toDesc;
} PrCoerceStruct;
```

As described in the comment above, this `PrGeneral` call is just a repackaging of the AppleScript coercion handler parameters. The call requests the driver to convert the descriptor pointed to by `fromDesc` into the type `toType`. The driver places the resulting type in `toDesc`. If the driver cannot perform the conversion, it places `OpNotImpl` in the `iError` field of the structure.

The most likely use of the `kPrCoerceOp` call to `PrGeneral()` will be a request to convert an AppleScript record containing print settings into a classic print record. In this case `fromDesc->descriptorType` is `typeAERecord` and `toType` is `kPrintRecordAEType` as defined above. The printer driver should create a print record containing the settings specified by the caller and place the handle to the new print record into `toDesc->dataHandle`. If there are settings that do not apply to your driver, you should ignore them, rather than returning an error, since a user may use the same script to print via multiple printer drivers. A complete list of the settings and their types is contained in the "PrintAETypes.h" file.

Note:
If your driver supports extended print records, you should return an extended print record from the `kPrCoerceOp` call to `PrGeneral()`. You must also correctly handle extended print records within your `PrJobMerge()` call.

An example of a dispatch function that would be called by a driver's `PrGeneral` routine is shown below:

```
#include "CoercePrGeneral.h"

OSErr prCoerce(TGnlData *general)
/* This routine implements the 'kPrCoerceOp' opcode.
 This routine is a PrGeneral remapping of the standard
 AppleScript coercion handlers.
 Called by PrintingLib's SettingsLib, this PrGeneral
 is a printer driver's chance to convert between an
 AppleScript print settings record and a print record
 and vice versa.
*/
{
 PrCoerceStruct  *prCoerceData =  NULL;
 AECoerceDescProcPtr coercionHandler = NULL;
 OSStatus   err = noErr;

 /* check parameters */
 if (!isValidAddr(general))
  return OpNotImpl;
 if (((PrCoerceStruct *)general)->iOpCode != kPrCoerceOp)
  return OpNotImpl;
 if (!isValidAddr(((PrCoerceStruct *)general)->fromDesc))
  return OpNotImpl;
 if (!isValidAddr(((PrCoerceStruct *)general)->toDesc))
  return OpNotImpl;

 prCoerceData = (PrCoerceStruct *) general;

 switch (prCoerceData->fromDesc->descriptorType) {
  case typeAERecord:
   if (prCoerceData->toType == kPrintRecordAEType) {
    coercionHandler = coercePrintSettings;
   }
  break;

  case kPrintRecordAEType:
   if (prCoerceData->toType == typeAERecord) {
    coercionHandler = coercePrintRecord;
   }
   break;

  default:
   coercionHandler = NULL;
 }

 if (coercionHandler) {
  err = (*coercionHandler)(prCoerceData->fromDesc,
       prCoerceData->toType,
       NULL,
       prCoerceData->toDesc);
 } else {
```

```
  err = OpNotImpl;
 }

 return err;
}
```

The handlers called from this function have the same signature as an Apple event coercion handler.

Note:
Printer drivers should not install or remove their own coercion handlers. The coercion handler is installed by the initialization routine for the SettingsLib shared library which is loaded by Desktop Print Spooler.

As the use of this call by an application requires application support for the extended print record, writers of printer drivers are free to assume that the print record parameters passed to the `kPrCoerceOp` call are extended, as detailed in Technote 1161, "Extending the Print Record for LaserWriter 8."

Back to top

Summary

The introduction of scriptable printing with PrintingLib 8.7 allows users to script the printing process. Supporting this scriptability requires that applications change. If your application already supports the Extended Print Record as described in Technote 1161, "Extending the Print Record for LaserWriter 8," the changes you will need to make to support scripting should be minimal. The rewards for AppleScript users will be large, since these changes will provide much more flexibility in printing. Developers of printer drivers are encouraged to add support for scriptable printing so users can use the same scripts with all of their printers.

Further References

- Technote 1161: Extending the Print Record for LaserWriter 8
- Technote 1177: Introducing LaserWriter 8, Version 8.7.
- AppleScript Language Guide for AppleScript 1.3.7

Back to top

Downloadables


Acrobat version of this Note (???K).


Sample Code

Back to top

Acknowledgments

Thanks to John Blanchard, Jose Carlos Colon, Paul Danbold, Chris Espinosa, Steve Evangelou, David Gelphman, Bill Hastings, Ingrid Kelly, and Cal Simone.

To contact us, please use the Contact Us page.
Updated: 05-October-1999

Technotes | Contents
Previous Technote | Next Technote