Technical Note TN1129

LaserWriter 8.5.1: The Settings Library

CONTENTS

The Settings Library (`SettingsLib`), introduced in LaserWriter 8.5.1, allows applications to access and change the information created by LaserWriter 8 when a desktop printer is created. `SettingsLib` prevents contention over the printer database between different parts of a printer driver, different printer drivers, and other printer database clients such as the Desktop Print Monitor. The printer database is stored in the LaserWriter 8 preferences file. The Settings Library is especially useful for applications that previously relied on the format of the `PAPA` resource in the LaserWriter 8 driver. With the introduction of LaserWriter 8.5.1, the size of the `PAPA` resource changed, and applications that depended upon its size broke. Since the resource size may change again in the future, Apple has introduced this library to prevent such problems. This Technote overviews the Settings Library APIs.

Updated: [May 11 1998]

The Settings Library and PrintingLib

`SettingsLib` is a code fragment which lives within the `PrintingLib` file. `SettingsLib` is used by the other code fragments in `PrintingLib`, as well as by LaserWriter 8, in order to access stored printer information. You should link against `PrintingLib` in order to gain access to the `SettingsLib` APIs.

Back to top

The Settings Library Interface

In order to avoid printer database collisions, clients should never open the printer database. Instead, the routines in this section should be used to get individual printer information collections to enumerate over the printer entries in the database.

psGetPrintingPrefsFolder

When using the recommended Settings Interface, the client targets the printer database by a filename only. The Settings Library knows the volume and folder that contains the printer databases; thus, the database name uniquely targets the database. In the current implementation, the printer databases exist in the "Printing Prefs" folder in the system's "Preferences" folder. In order to get the location of the folder that holds the printer databases, use `psGetPrintingPrefsFolder()`. This API can also be used if you wish to determine where to place printing-related data.

```
OSStatus psGetPrintingPrefsFolder(short *vRef, long *folderId);
```

This function locates, and creates (if needed) the current system's printing preferences folder. On exit, the volume for the printing preferences folder is placed into `*vRef` and the directory ID is placed into `*folderId`. If the folder does not exist and cannot be created, then an error is returned.

psGetPrefsPrinterInfo

Once the client has the printer database name and a PAPA handle that identifies a printer, the `psGetPrefsPrinterInfo()` routine can be used to get a collection containing the printer's information. The client can then use the Collection Manager to access the individual items within the collection. It is the client's responsibility to dispose of the collection using the Collection Manager's `DisposeCollection()`. For more information on the Collections Manager, please see *Inside Macintosh: QuickDraw GX Environment and Utilities* .

```
Collection psGetPrefsPrinterInfo(StringPtr prefsName, Handle papa);
```

Returns a collection containing information about the printer specified by PAPA. The information is taken from the printing preferences file, whose name is `prefsName`. If the collection cannot be obtained, then NULL is returned.

While the filename is used to target the printer database, a handle containing a PAPA is used to target an individual printer in the database. The PAPA consists of three Pascal strings packed together, followed by data that is unique to the communications method for the given printer. In other words, the data after the three Pascal strings is not inspected by the `SettingsLib`, and its contents are up to the client.[1] The three Pascal strings at the start of the PAPA are the printer's name, the network object type, and the printer's zone. These three strings identify a given printer database entry; together, they form a unique key to the database. Please see Technote 1115: "The Extended PAPA Resource" for more information.

psUpdatePrefsPrinterInfo and psRemovePrefsPrinterInfo

Once the client has a collection describing a printer, the Collection Manager can be used to add, remove, or alter the printer information. When the client has made the necessary changes, the collection is handed to `psUpdatePrefsPrinterInfo()` to replace the existing collection for a given printer. Even after calling `psUpdatePrefsPrinterInfo()`, it is the client's responsibility to dispose of the collection. To remove a printer from the database completely, use `psRemovePrefsPrinterInfo()`.

```
OSErr psUpdatePrefsPrinterInfo(StringPtr prefsName, Handle papa, Collection prInfo);
```

Stores the information contained in the collection `prInfo` (for the printer specified by PAPA) into the preferences database named prefsName. All information associated with the printer specified by PAPA is replaced with the information contained in `prInfo`.

```
OSErr psRemovePrefsPrinterInfo(StringPtr prefsName, Handle papa, Collection prInfo);
```

Removes all information about the printer specified by PAPA from the printing preferences file named `prefsName`. If PAPA is NULL, then `prInfo` is searched for a printer name hint, which is then used to choose the entry to be deleted from the preferences database.

For developers who want to add their own data, Apple recommends that you use the tag APPL along with an ID that is your registered creator type. For example, Adobe Acrobat should store its data as tag APPL and ID CARO. Thus, each application gets only a single slot in the database. We further recommend that the data stored in that single slot be a flattened collection in which you can use any tags and IDs you want.

```
#define kHintAppPrivateTag APPL
#define kHintAppPrivateId    // Application's creator type.
#define kHintAppPrivateVar   // A flattened collection.
```

Note:
Apple reserves the APPL tag so that applications and non-drivers can store information in the printer database and other PrintingLib collections. A non-driver should use its registered creator type as the ID for its data, and the data contents should be a flattened collection.

psCountPrefsPrinters and psGetPrefsIndName

If it is necessary to enumerate over all of the printers in a database, the client should first call psCountPrefsPrinters() to get the total number of printers in a database. Once the total number of printers is known, the client can enumerate over the printers by handing an index from 1 to the total number of printers (inclusive) with psGetPrefsIndName(). psGetPrefsIndName() then returns the printer name and zone name of the indexed printer info collection.[2]

```
long psCountPrefsPrinters(StringPtr prefsName);
```

Return the number of printers in the printing database specified by prefsName'.

```
OSErr psGetPrefsIndName(StringPtr prefsName, long index, StringPtr name, StringPtr
zone);
```

Use psGetPrefsIndName() to enumerate over the printer info collections in the printer database specified by prefsName. index should be between 1 and psCountPrefsPrinters(), inclusive. For each index, this function fills in name with the printers name and zone with the printers zone. On entry, name and zone must point to at least Str32 arrays of characters. The client can use the returned name and zone to build a PAPA that can be handed to the other settings routines.

Back to top

The Extended PAPA and SettingsLib

PrintingLib 8.5.1 supports the use of an extended PAPA record as described in Technote 1115: "The Extended PAPA Resource." In order to limit the number of clients who need to deal directly with the extended PAPA format, the Settings Library provides routines for creating extended PAPA handles, and for setting and retrieving tagged values in the PAPA. This section describes the APIs for this use.

psSetPapPapa, psSetInfraredPapa, psSetFilePapa, psSetHoldPapa, psSetLprPapa, and psSetCustomPapa.

To create a new extended PAPA record, SettingsLib clients allocate a new record via NewHandle() and then pass the handle, a PAP printer name, object type, zone, and optional AppleTalk address block to the various psSetXXXPapa() routines. The memory block is filled in as an extended PAPA record for the printer.

```
OSStatus psSetPapPapa(Handle papaH, const Byte * printer, const Byte * atType, const
Byte * zone, AddrBlock *addr);
```

Given the name of an AppleTalk PAP printer (pointed to by printer), an AppleTalk device type (pointed to by atType), and an AppleTalk zone (pointed to by zone), this builds a PAPA in the block whose handle is papaH. papaH must be the handle of a memory block of at least 103 bytes long. addr is a pointer to an AppleTalk address block. If addr is not NULL on entry, the address block is copied after the printer-type-zone strings in the PAPA.

If addr is NULL, then zeros are entered for the address in the PAPA.

```
OSStatus psSetInfraredPapa(Handle papaH, const Byte * printer, const Byte * zone);
```

This function will fill in the memory, whose handle is papaH, with an extended PAPA describing an infrared printer. Together, printer and zone should uniquely identify a printer since this pair is used as a key to store information in the printer database. papaH must be the handle of a memory block of at least 103 bytes long.

```
OSStatus psSetFilePapa(Handle papaH, const Byte * printer, const Byte * zone);
```

This function will fill in the memory whose handle is papaH with an extended PAPA describing a save to file desktop printer. Together, printer and zone should uniquely identify a printer since this pair is used as a key to store information in the printer database. papaH must be the handle of a memory block of at least 103 bytes long.

```
OSStatus psSetHoldPapa(Handle papaH, const Byte * printer, const Byte * zone);
```

This function will fill in the memory whose handle is papaH with an extended PAPA describing a hold desktop printer. Together, printer and zone should uniquely identify a printer since this pair is used as a key to store information in the printer database. papaH must be the handle of a memory block of at least 103 bytes long.

```
OSStatus psSetLprPapa(Handle papaH, const Byte * printer, const Byte * zone, const
Byte * tcpAddr, const Byte * qName);
```

This function will fill in the memory whose handle is papaH with an extended PAPA describing an lpr desktop printer. Together, printer and zone should uniquely identify a printer since this pair is used as a key to store information in the printer database. tcpAddr is the network address of the lpr printer.

This address can be in either name or dot format, i.e., either "\plaser.rbi.com" or "\p204.188.109.155". qName is the name of the print queue associated with a spooler at tcpAddr. If qName is NULL, then the default queue for the printer/spooler is used. papaH must be the handle of a memory block of at least 1024 bytes long.

```
OSStatus psSetCustomPapa(Handle papaH, const Byte * printer, const Byte * zone);
```

This function will fill in the memory, whose handle is papaH, with an extended PAPA describing a custom desktop printer. Together, printer and zone; should uniquely identify a printer since this pair is used as a key to store information in the printer database. See Technote 1113: "Customizing Desktop Printer Utility" for more information.

psPapaToCollection and **psCollectionToPapa**

If a client has a handle to an extended PAPA record, the routines psPapaToCollection() and psCollectionToPapa() can be used to access and set extended PAPA tag block. The routine psPapaToCollection() takes a handle to an extended PAPA record and a collection, and copies the tag values from the extended PAPA into the collection. The client can then use the Collection Manager to enumerate and alter the tag values. The call psCollectionToPapa() takes the tags from a collection and copies them into an extended PAPA record, replacing any tags that may already be there.

```
OSStatus psPapaToCollection(Handle papaH, Collection coll);
```

Given a handle to a PAPA record, papaH, and a collection, coll, psPapaToCollection() creates a set of collection items in coll, all of which have the tag PAPA and whose ID is the tag from the extended PAPA record. For example, if the

extended PAPA record papaH has a tag with type TCP, then the data for that tag will be placed into coll with tag PAPA and ID TCP. The TAGS value from the extended PAPA is not copied into the collection.

```
OSStatus psCollectionToPapa(Collection coll, Handle papaH);
```

Given a collection, coll, this routine enumerates the collection items with the tag PAPA and makes them part of the extended PAPA record whose handle is papaH. The ID of the collection item becomes the tag in the PAPA. Any existing tags in the PAPA record papaH are replaced. The error errNoRoomInPapa (-8941) is returned if all of the tags do not fit. All of the tags that do fit are added into the PAPA.

psGetDTPType

If a client has an extended PAPA and needs to know the type of printer described by that PAPA, psGetDTPType() can be used.

The psGetDTPType() function will classify a desktop printer described by a PAPA handle as being one of the following DTPType's:

```
typedef enum{
 kInvalidDTP =  0,     // An invalid DTP type.
 kHoldDTP = '=Hld',     // Hold DTP's spool job to a print queue which
                 // is always on hold.
 kFileDTP = '=Fil',     // Save to file DTP's can't print; they just
             // create PS/PDF files.
 kLprDTP = '=LPR',     // LPR printers print over TCP to LPR
             // spoolers/printers.
 kCustomDTP = '=Cst',   // Custom DTP's launch an app to post-process
                   // the PostScript.
 kPapaDTP = 'PAP ',     // An AppleTalk PAP network printer.
 kInfraRedDTP = '=Ird'  // A printer connected via an infrared link.
} DTPType;
```

OSStatus psGetDTPType(Handle papaH, DTPType *dtpTypeP);

Given a valid PAPA handle, papaH, this routine will place the DTP type into *dtpTypeP. If there is an error, then an error value is returned and *dtpTypeP is set to kInvalidDTP.

psIsValidPapaHandle

Useful when debugging, psIsValidPapaHandle() tries to determine if a PAPA handle is valid. It will return false if the PAPA seems to be invalid.

```
Boolean psIsValidPapaHandle(Handle papaH);
```

Return true if papaH looks to be a valid PAPA handle.

Back to top

Summary

In order to avoid printer database collisions, Apple recommends that every developer who needs access to certain parts of the LaserWriter 8 driver should use the Settings Library as documented here.

Back to top

References

Technote 1115: "LaserWriter 8.5.1: The Extended PAPA Resource"

Inside Macintosh: QuickDraw GX Environment and Utilities, Chapter 5

1

There are printer driver limitations on the PAPA that do not affect the `SettingsLib`. In particular, due to the "MultiFinder Friendly" bit, printer drivers are not allowed to change the size of the PAPA resource stored in the driver. Thus, the driver's PAPA is a fixed size.

2

Notably lacking from the return values for `psGetPrefsIndName()` is the network object name. Today, all of the printer database entries have a network object name of "LaserWriter." A new routine will be added in the future to deal with this limitation.

Back to top

Downloadables

Acrobat version of this Note (K). Download

Back to top
