Technical Note TN1113

Customizing Desktop Printer Utility

CONTENTS

In the current LaserWriter 8 driver, the Chooser lists printers which are available on the AppleTalk network (either LocalTalk or AppleTalk) with an NBP type of "LaserWriter". When the user selects one of these printers, LaserWriter 8 creates a desktop printer by sending an Apple Event to the Finder. In the past, this mechanism of setting up printers has served the needs of most users. However, with LaserWriter 8.5.1, Apple is responding to several requests for more flexible desktop printing functionality. To this end, Apple has introduced a new application called Desktop Printer Utility which enables users to create additional types of desktop printers. Users can now create desktop printers which use the Unix *lpr* protocol for printing, in addition to the regular AppleTalk PAP printers. There are also "hold" printers which represent local print queues and "virtual" printers which represent printers which are not available on the network.

Along with the expanded capabilities, Desktop Printer Utility is also customizable. LaserWriter 8.5.1 and Desktop Printer Utility (together) support the creation and use of desktop printers known as Custom DTPs. When printed to, Custom DTPs cause the LaserWriter 8.5.1 driver to create a PostScript file and to launch an application that can post-process the PostScript. This post-processing application can do anything it likes with the PostScript file, such as converting the PostScript into another file format, transferring the file to another location

using a modem or a network connection, or displaying the PostScript file to the user. This Technote describes how a developer might customize Desktop Printer Utility for use with an application.

Updated: [Feb 06 1998]

Customizing the DTP Utility

In order to add a new custom DTP to the list of supported DTPs in Desktop Printer Utility, you must add resources to the utility's resource fork. Desktop Printer Utility contains templates to ease the creation of these resources.

Back to top

'CsDs'(CustomDTPResource) Resource

In order to customize Desktop Printer Utility, a 'CsDs' (CustomDTPResource) resource must be added to Desktop Printer Utility. The LaserWriter uses this information to find the appropriate application to launch and to customize your DTP interface.

The resource's format is defined by the structure 'CustomAppDesc:'

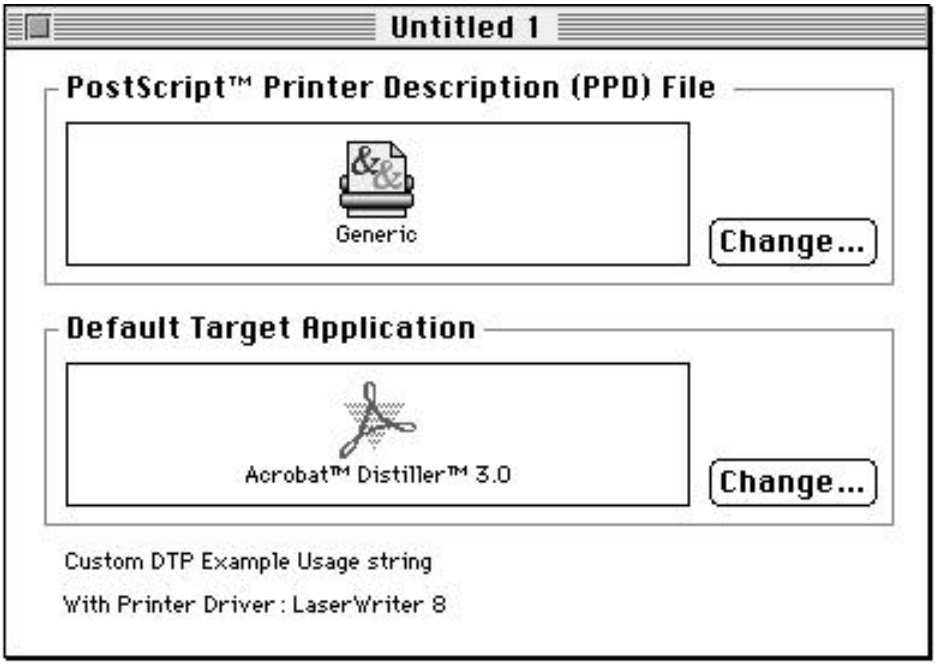
```
#define kVariableLen 1

struct CustomAppDesc{
    OSType appSignature;
    Str255 docType;
    Str255 helpText;
    Str255 usage;
    Str255 appFileName;
    short numOfHintsFollow;
    HintRsrcSpec hintRsrc[kVariableLen];
};
typedef struct CustomAppDesc CustomAppDesc;
```

- **appSignature**: The application signature of the application to launch. The LaserWriter driver uses this information to find the application on all mounted volumes which are available to the user.
- **docType**: A string that describes the type of custom DTP. This string appears in the list of document types when "New" menu is selected. This string is highlighted in Figure 1.
- **helpText**: This string appears when your custom DTP is selected. This string will appear in the "New" dialog field just below the DTP choices. This string appears as 'Custom DTP Example Help Text' in Figure 1.
- **usage**: A string that describes the usage of your custom DTP. This string appears as "Custom DTP Example Usage string" in Figure 2.
- **appFileName**: The name of the application to launch. The LaserWriter driver uses this information to find the application on disks which are available to the user as well as to provide error messages to the user (e.g. "SurfWriter could not be launched due to insufficient memory").

- **numOfHintsFollow**: The number of hints provided in the hintRsrc array. See the "Hints Resources" section for more details.
- **hintRsrc**: An array of hints that outlines the kind of PostScript that should be generated by the driver. See the "Hints Resources" section for more details.

Figure 1

Figure 2



Back to top

Hints Resources (optional)

The CustomDTPResource can provide "hints" about the PostScript the driver should generate. Each

"CustomDTPResource" contains the field "numOfHintsFollow" which represents the number of "hints" provided. The "hints" data is placed in an array that contains "numOfHintsFollow" number of copies of the structure:

```
struct HintRsrcSpec{
       OSType rsrcType;
       short rsrcID
};
typedef struct HintRsrcSpec HintRsrcSpec;
```

Each entry in the array points to a resource which is converted to a hint in the driver's preferences. The hint resource types, which are currently supported, have the following formats:

/* PostScript Level */

The application that postprocesses the PostScript for a custom DTP may require or prefer a certain PostScript level. The 'pslv' hint specifies the desired PostScript level.

Resource type 'PSlv'

```
typedef struct{
    OSType hintType;        //'pslv' hint type
    long hintId;            // set to 1
    long psLevel;
}PSlvResource;

#define kHintLanguageLevelTag   'pslv'
#define kHintLanguageLevelId    1
```

The possible(current) values for psLevel which can be set are:

-3	Level 2 and 3	
-2	Level 1 and 2	
-1	Unknown Level	
0	Other Level	//do not use
1	Level 1	
2	Level 2	
3	Level 3	

Note:
-3 will create PostScript that takes advantage of PostScript level 2 and level 3 printers, but the data may fail on a level 1 printer. -2 will create PostScript that will succeed on level 1, 2, and 3 printers.

/* Binary OK - 'bnok' */

This hint allows or disallows the use of binary data in the PostScript file.

Resource type 'BNok'

```
typedef struct{
    OSType hintType;        //'bnok' hint type
    long hintId;            // set to 1
    TriState binaryOK;
}BNokResource;

#define kHintADOIsBinaryOKTag   'bnok'
#define kHintADOIsBinaryOKId    1
```

There are three acceptable settings for binaryOK which answer the question, "Is binary data ok?":

0	False	
1	True	
2	Unknown	//do not use

Note:
This hint should not be created in the DTP with a value of unknown; however, it is available for future use.

/* Job Type - 'jobt' */

This hint specifies the type of PostScript to generate.

Resource type 'JObt'

```
typedef struct{
    OSType hintType;        //'jobt' hint type
    long hintId;            // set to 1
    char jobType;
}JObtResource;

#define kHintJobTypeTag     'jobt'
#define kHintJobTypeId      1
```

Currently only two PostScript forms for `jobType` are available:

0	`psJobPostScript`
1	`psJobEPSNoPreview`

/* Font Handling - 'font' */

This hint specifies the font inclusion. The font handling hint consists of a four-byte flag followed by a list of NULL terminated font names.

Resource type 'FOnt'

```
typedef struct{
    OSType hintType;         //'font' hint type
    long hintId;             //set to 1
    long tag;
    unsigned char name[1];
}FOntResource;

#define kHintIncludeFontsTag        'font'
#define kHintIncludeFontsId         1
```

The possible values for tag are:

0 kIncludeNoFontsOtherThan

1 kIncludeAllFontsBut

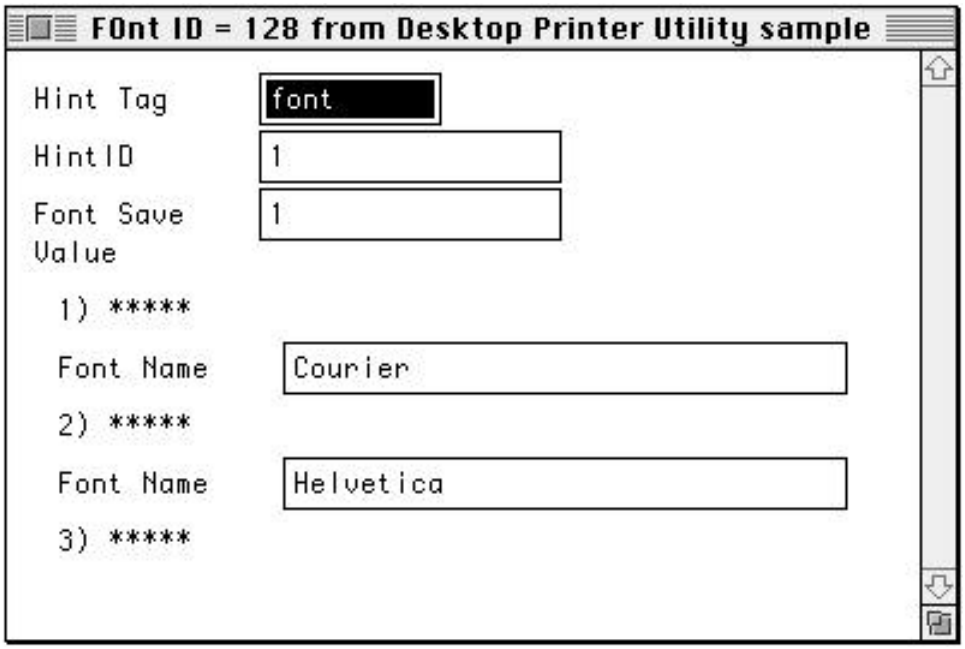
If the flag is 'kIncludeNoFontsOtherThan', the utility will only include the font definitions of fonts that are listed after the flag. If there are no font names after the flag, no font resources are included in the output. If the flag is 'kIncludeAllFontsBut', font resources are included in the output unless the font names are listed. To include all fonts, use the 'kIncludeAllFontsBut' flag with no fonts listed afterwards.

Back to top

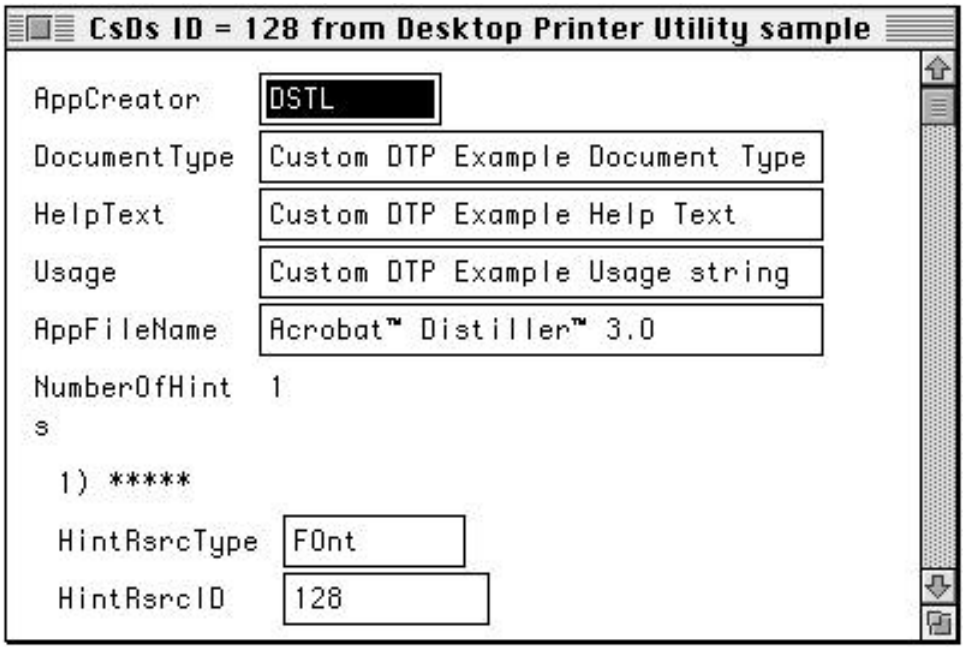
Sample Custom DTP

Following is a sample of a custom DTP that excludes Courier and Helvetica from the PostScript file via a 'CsDs' and 'FOnt' resource. This sample is intended to give developers an idea of how easy it is to customize the DTP Utility.

The first step towards adding the Custom DTP is to create a hint resource of the correct type. The example shown is the 'FOnt' resource which provides data about font inclusion.



The second step is to associate the hint with a particular DTP by adding hint resource information to the 'CsDs' resource for a particular DTP.



And that's all that is required to get the DTP Utility to create a custom DTP that produces a PostScript file without Courier and Helvetica.

Limitations of the DTP Utility

Due to the parallel development schedules of Mac OS 8.0 and LaserWriter 8.5.1, DTP Utility 1.0 (which ships with LaserWriter 8.5.1) will not install on Mac OS 8.0. It will, however, work with the upcoming Mac OS 8.1.

Licensing the DTP Utility

In order to ship a customized DTP Utility with your application, you must license the DTP Utility from Apple. Please contact Apple's Software Licensing group (sw.license@apple.com or 512-919-2645) for more information.

Back to top

Summary

This Technote has described and demonstrated all that is necessary for you to customize Desktop Printer Utility. We encourage all developers to give this a try.

Back to top

References

TN 1112: Introducing the LaserWriter Driver Version 8.5.1

Back to top

Downloadables

Acrobat version of this Note (K). Download

Technical Notes by API | Date | Number | Technology | Title
Developer Documentation | Technical Q&As | Development Kits | Sample Code