X Print Service Sample Implementation

Version 1.0
X Window System
Version 11 Release 6.4

A. Deininger, T. Gilg, J. Miller, H. Phinney, C. Prince
Hewlett-Packard Company

Table of Contexts

1 Introduction

This document describes the implementation of the X Print Server distributed with Release 6.4 of the X Window System. The intended reader is the system administrator who needs to configure the X Print Server for a particular set of printers and a particular spooling subsystem.

The syntax and format of the configuration files read by the X Print Server are described.

This document is not an X Consortium standard.

2 X Print Configuration Databases

Configuration files provide the raw information that is used by the X Print Service components. Strictly speaking, the configuration files, print dialog manager, and DDX drivers of the print server form a matched set. The configuration files, though, have been designed to be as flexible as possible.

Most of the configuration files are in the form of an XRM resource file. This provides maximum flexibility. The hierarchical nature of the data base avoids name clashes, and wild cards can be used to identify characteristics that apply to many printers. Also, additional attributes can be added later.

This section documents the configuration directories and files used by the X Print Service.

$XPCONFIGDIR is an environment variable read by the X print server which defines the root of the configuration directory hierarchy. If $XPCONFIGDIR is not defined, the server will default to `<XRoot>/lib/X11`, where `<XRoot>` is the root of the X11 install tree. Configuration values are determined by performing these steps:

1. Search $XPCONFIGDIR`/C/print` to obtain default values from a configuration file.
2. If the configuration file is not found, server-defined defaults will be used.
3. For locales other than C, search $XPCONFIGDIR/$LANG`/print` and use the configuration file values to augment the defaults determined above.

One exception to this is the `Xprinters` file. This file indicates which printers will be managed by the X Print Server. The path and name of this file is indicated by the `-XpFile` command line option defined by the X Print Server. If the command line option is not present, the X Server will default to $XPCONFIGDIR`/C/print/Xprinters`. This file is optional.

There are several types of configuration files stored in several subdirectories:

- A file that indicates which printers will be managed by the X Print Server. It is referred to as the `Xprinters` file.
- Printer attribute files that define the capabilities of the printer model. The name of the file is typically all uppercase, and consists of the manufacturer and the model of printer. Examples of file names are: `HPDJ1600C`, `IBM-4039-16l`, and `SUN-NP20`.
- Printer attribute files that define the capabilities of printers installed on a particular X Print Server.
- Job and document attribute files that specify initial values for the print operation.
- Optional DDX driver configuration files. The format of each file is internal to the corresponding DDX driver.

Most of the configuration files documented in this section are encoded in COMPOUND_TEXT, as defined by the X Window System. The only exception is the optional DDX driver configuration files. These files are defined at the discretion of the driver developer.

2.1 Configuration Directories

2.1.1 Print Configuration Directory

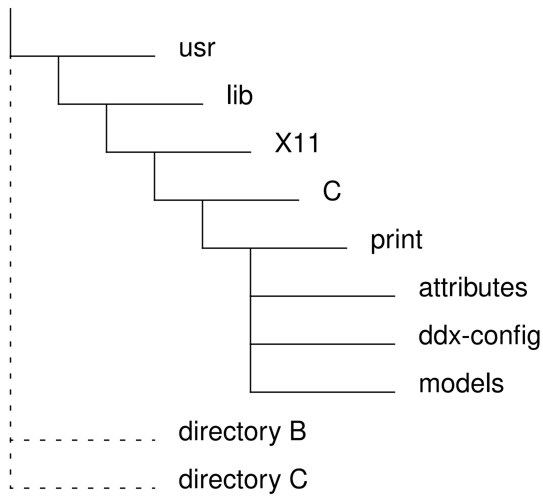

Figure 0-1.Example Print Configuration Directory

The X Print Service configuration directory is assumed to be `/usr/lib/X11/C/print` for the purposes of this discussion. During actual use, the configuration files will be distributed throughout the configuration hierarchy, as described in the “Configuration Directories” section.

At the top level of the locale-specific `print` directory, three subdirectories are defined. The `ddx-config` directory contains configuration information specific to X Print Server DDX drivers. The `models` directory defines default attributes and internal font metrics for various models of printers. The `attributes` directory defines attributes for the various X Printers defined on the host system. The following sections describe these directories in more detail.

2.1.2 Printer Model Configuration Directories

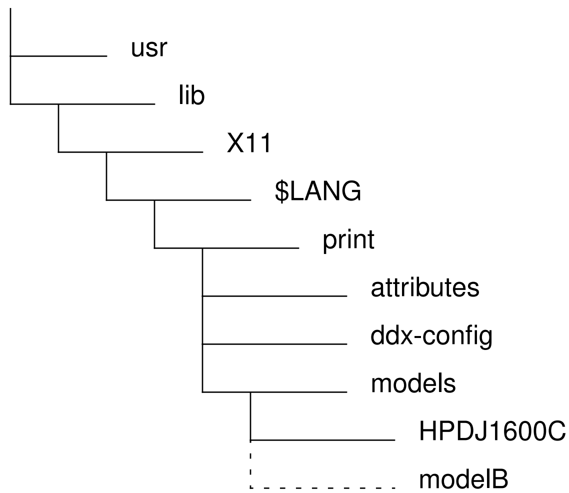

Figure 0-2.Example X Printer `models` Directory

The `models` directory contains subdirectories that define configuration information for various models of printers. Each subdirectory corresponds to a specific printer model or a specific class of printer models. The names of these model directories define valid values for the **`xp-model-identifier`** attribute in the printer attributes file. See the “Printer Attributes” section.

It is recommended that only uppercase characters be used for the names of model configuration directories. This will help avoid namespace collisions between model names and printer names when they are used as qualifiers in the attribute files. See the “Printer Attributes”, “Document Attributes”, and the “Job Attributes” sections for information on the format of these files.

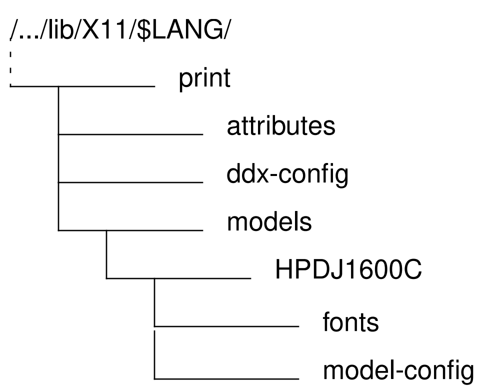

Figure 0-3.Example Printer Model Configuration Directory

The printer model configuration directory contains a `model-config` file and a `fonts` directory. The `model-config` file defines a set of default attributes for a specific printer model or a specific class of printer models. See “Printer Model Attributes File” for details on the format of this file.

The `fonts` directory defines font metrics for the printer’s internal fonts. If any fonts are defined under a locale-specific subdirectory, they obscure all fonts defined under the default C locale subdirectory.

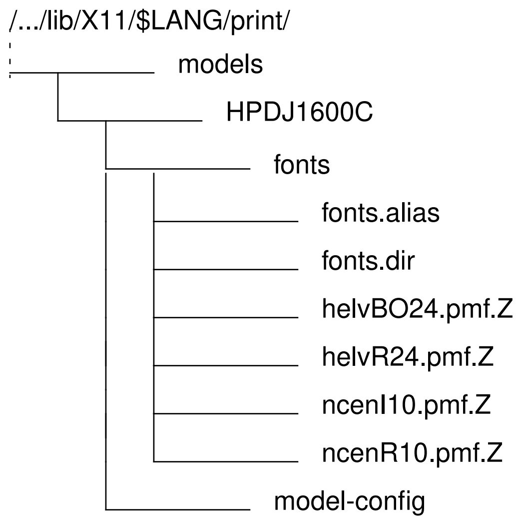

Figure 0-4.Example X Printer Internal Fonts Directory

The `fonts` directory is read by the X Print Server. See “Fonts” on page 13 for more information.

2.1.3 Printing Attributes Configuration Directory

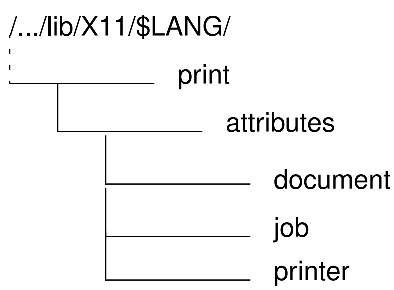

Figure 0-5.Printing Attributes Configuration Directory

The files in the `attributes` directory contain initial values for the X Print Service attributes. These attributes define print setup options (`document` and `job`) and provide printer capabilities (`printer`). See the “Printer Attributes File”, “Document Attributes File”, and the “Job Attributes File” sections for information on the format of these files.

2.1.4 DDX Driver Configuration Directories

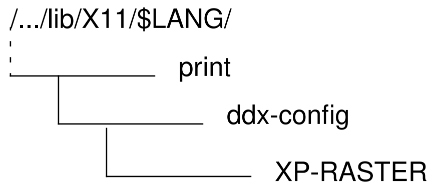

Figure 0-6.Example `ddx-config` Directory

The `ddx-config` directory contains DDX driver configuration directories. A DDX driver may or may not require one of these directories. The contents of each directory is specific to the corresponding driver. The name of the directory is the same as the driver name provided by the DDX driver to the X Print Server, and is also used as the value of the **`xp-ddx-identifier`** printer attribute.

Figure 0-7. shows an example of the DDX driver configuration directory for the raster driver.

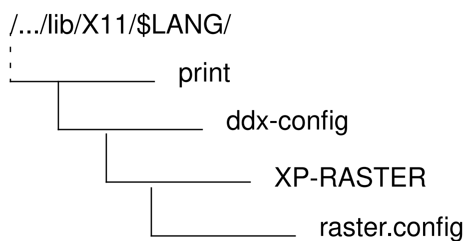

Figure 0-7.Example DDX Driver Configuration Directory

Driver configuration files in this directory may be assigned on a per-printer basis by using the **`xp-ddx-config-file-name`** printer attribute. Whether or not this attribute is utilized is determined by each individual driver.

2.2 Xprinters File

NAME

`Xprinters file:`Identifies the printers to be managed by an X Print Server

DESCRIPTION

The `Xprinters` file is read by an X Print Server during initialization to determine which printers it will manage.

Lines in the file consist of a keyword followed by a value. Keyword recognition is case-sensitive. Any data following the comment character “`#`” on a given line is ignored.

The `Xprinters` file is encoded in COMPOUND_TEXT as defined by the X Window System.

KEYWORDS

Augment_Printer_List

This keyword is used to generate a list of printer names that will be added to the list of printers the server will manage. If this line is not specified, or if the Xprinters file does not exist, the server will generate a list of printers by utilizing the output of *lpstat(1)*. Predefined values for the `Augment_Printer_List` keyword are:

%default

Explictly invoke the default behavior, i.e. augment the list of printers by utilizing the output of *lpstat(1)*.

%none%

Do not augment the list of printers. This provides a way to override the default behavior of calling *lpstat(1)* when no `Augment_Printer_List` line is present.

In addition, the value may be specified as a POSIX shell command pipeline that generates a list of printers on **`stdout`**. This generated list is added to the list of printers managed by the server.

Printer A whitespace delimited list of one or more printer names to add to the list of printers managed by the server.

Map Attributes configuration files utilize a printer qualifier, defined by the X Print Server, that is the printer name by default, provided the characters comprising the printer name conform to the restricted set of characters allowed for the printer qualifier, that is, the set of characters allowed for Xrm resource names. The `Map` keyword is provided to allow specification of a printer qualifier when a default printer qualifier is not generated by the server, or if an override of the default qualifier is desired.

The `Map` value is in the form *<printer name> <printer qualifier>,* for example:

```
Map könig koenig
```

EXAMPLE

```
########################################################################
#
# Xprinters sample configuration file
#
# The Xprinters file is read by an X Print Server during initialization in
# order to determine which printers it will manage. The actual file name and
# path is given to the X Print Server via the -XpFile command
# line option.
########################################################################

########################################################################
# Use lpstat to augment the list of printers managed by the
# server. (This is the default behavior if the Xprinters file is
# not specified, or if an "Augment_Printer_List" line is not specified.)
########################################################################
Augment_Printer_List %default%

########################################################################
# Use the specified command pipeline to augment the list of printers
# managed by the server.
########################################################################
#Augment_Printer_List lpstat -a | cut -d " " -f 1  #equivalent to default

########################################################################
# Do not augment the list of printers managed by the server.
########################################################################
#Augment_Printer_List %none%

########################################################################
# Add individual printers to the list of printers managed by the
# server.
########################################################################
#Printer laser_1 laser_2 laser_c4
#Printer deskJet_1 deskJet_2
#Printer xpress

########################################################################
# Provide printer qualifiers for non-conforming printer names
########################################################################
Map könig koenig
```

SEE ALSO

- lpstat(1)

2.3 Printer Model Attributes File

NAME

`Printer model attributes file:` Printer model capabilities

DESCRIPTION

The printer model attributes file consists of printer attributes for a specific printer model or a specific class of printer models. This file is delivered by a printer vendor or DDX printer driver developer in order to provide default configuration information for a printer.

Valid attributes are based on a subset of the POSIX 1387.4 Printer Object attribute definitions (*note:* the X Print Service is *not* an implementation of POSIX 1387.4). See the “Printer Attributes” section for the complete list.

The printer model attributes file is encoded in COMPOUND_TEXT as defined by the X Window System.

Attribute names must be qualified using either the **`xp-model-identifier`** or an asterisk (`*`). For example, if `HPDJ1600C` is the **`xp-model-identifier`**, then to initialize the **`plexes-supported`** attribute to **`simplex`**, use: `HPDJ1600C.plexes-supported: simplex`. For the asterisk, use: `*.plexes-supported: simplex`. If the same attribute is specified using each method, the **`xp-model-identifier`** qualified entry takes precedence.

EXAMPLE

```
! This is the configuration file for the HP DeskJet 1600C printer.
! It is designed for use with the CDEnext Sample Implementation
! PCL, raster drivers, and print dialog manager.

HPDJ1600C.printer-model: Hewlett-Packard DeskJet 1600C
HPDJ1600C.descriptor: Hewlett-Packard DeskJet 1600C
HPDJ1600C.printer-resolutions-supported: 300
HPDJ1600C.content-orientations-supported: portrait landscape
HPDJ1600C.document-formats-supported: {PCL 5}
HPDJ1600C.plexes-supported: simplex
HPDJ1600C.xp-ddx-identifier: XP-PCL
HPDJ1600C.xp-embedded-formats-supported: {PCL 5} {HPGL 2}
HPDJ1600C.dt-pdm-command: dtpdm

! na-letter, iso-a4, na-legal, na-number-10-envelope, more?
! assumes 1/4” unprintable margins for all media
HPDJ1600C.medium-source-sizes-supported: \
{‘’ \
  {na-letter FALSE {6.35 209.55 6.35 273.05}} \
  {iso-a4 FALSE {6.35 203.65 6.35 290.65}} \
  {na-legal FALSE {6.35 209.55 6.35 349.25}} \
  {na-number-10-envelope FALSE {6.35 222.25 6.35 98.425}} \
}
```

SEE ALSO

- “Printer Attributes”.

2.4 Printer Attributes File

NAME

`printer attributes file:` Printer configuration

DESCRIPTION

The printer attributes file identifies capabilities and defaults for an X printer on the host system. This file is defined by the system administrator. Definitions in this file override attributes defined in the Printer Model Attributes file.

Valid attributes are based on a subset of the POSIX 1387.4 Printer Object attribute definitions. See the “Printer Attributes” section for the complete list.

The printer attributes file is encoded in COMPOUND_TEXT as defined by the X Window System.

Attribute names must be qualified by using one of the following (listed in order of precedence):

printer qualifier

Set this attribute for the printer indicated by the printer qualifier. The set of valid printer qualifiers is defined as the list of printer qualifiers managed by the X Print Server (the server typically generates this list by reading the `Xprinters` file).

Example: `dj_1.document-formats-ready: {PCL 5}`

xp-model-identifier

Set this attribute for all printers of a specific model:

Example: `HPDJ1600C.document-formats-ready: {PCL 5}`

Set this attribute for all printers:

Example: `*.document-formats-ready: {PCL 5}`

EXAMPLE

```
*.xp-model-identifier: HPLJ4SI

HPDJ1600C.input-trays-medium: { main na-letter }

deskJet_1.descriptor: DeskJet 1600C in Bob’s Cubicle
deskJet_1.xp-model-identifier: HPDJ1600C

laser_1.descriptor: 4si in Brock’s Bay
laser_1.input-trays-medium: {top na-letter} {bottom na-legal} \
    {large-capacity na-letter}
laser_2.descriptor: laserjet in test area
laser_2.plexes-supported: simplex
laser_2.input-trays-medium: {top iso-a4} {bottom iso-a4}
```

SEE ALSO

- “Printer Model Attributes File”.
- The “Printer Attributes” section.

2.5 Job Attributes File

NAME

`job attributes file`:Print job initial values

DESCRIPTION

The job attributes file is encoded in COMPOUND_TEXT as defined by the X Window System.

Attribute names must be qualified by using one of the following (listed in order of precedence):

printer qualifier
: Set this attribute for the printer indicated by the printer qualifier. The set of valid printer qualifiers is defined as the list of printer qualifiers managed by the X Print Server (the server typically generates this list by reading the `Xprinters` file).

  Example: `laser_1.job-name: Payroll Reports`

xp-model-identifier
: Set this attribute for all printers of a specific model:

  Example: `HPDJ1600C.job-name: Payroll Reports`

*
: Set this attribute for all printers:

  Example: `*.job-name: Payroll Reports`

EXAMPLE

```
! defaults
*.job-name:
*.notification-profile: {}

! Printer laser_1 prints paychecks - always send email on
completion
laser_1.notification-profile: {{event-report-job-completed}
electronic-mail}
laser_1.job-name: Payroll Reports
```

2.6 Document Attributes File

NAME

`document attributes file:` Print document initial values

DESCRIPTION

The document attributes file is encoded in COMPOUND_TEXT as defined by the X Window System.

Attribute names must be qualified by using one of the following (listed in order of precedence):

printer qualifier

Set this attribute for the printer indicated by the printer qualifier. The set of valid printer qualifiers is defined as the list of printer qualifiers managed by the X Print Server (the server typically generates this list by reading the `Xprinters` file).

Example: `dj_1.plex: duplex`

xp-model-identifier

Set this attribute for all printers of a specific model.

Example: `HPDJ1600C.plex: duplex`

*Set this attribute for all printers.

Example: `*.plex: duplex`

EXAMPLE

```
*.default-input-tray: top
*.default-printer-resolution: 300
*.plex: duplex
*.content-orientation: portrait
*.copy-count: 1
*.document-format: {PCL 5}
HPLJ4SI.default-printer-resolution: 600
printer_1.default-input-tray: large-capacity
deskJet_1.plex: simplex
```

2.7 DDX Driver Configuration Files

NAME

`DDX configuration file:` DDX driver defined configuration

DESCRIPTION

The DDX configuration file is defined at the discretion of the DDX driver developer. The format of the information defined in the file is internal to the DDX driver. The developer may choose to publish the format of this file to allow for customization by system administrators.

EXAMPLES

The Raster driver supplied with the X Print Service utilizes a DDX configuration file. Here is an example of how it is defined:

```
! Raster DDX print driver configuration file
*PageCommand: command -o option
```

SEE ALSO

The “X Printer Driver Interface” on page 15.

3 Fonts

Fonts play an important role in the printing environment. The basic tenet of the DtPrint X Server is to act like a regular X server. X programmers will find the interface familiar.

Fonts provide the ability to render text. They may come from several sources:

- Fonts built into the printer (both bitmapped and scalable)
- Bitmapped fonts on the server's local disk
- Scalable and bitmapped fonts in a format compatible with the printer
- Fonts from a font server

From a printing application's point of view, the LoadFont, QueryFont, and ListFonts requests work as usual after the creation and setting of a print context. If the document-formats-supported attribute contains multiple document formats, then the client must set the document-format attribute prior to performing any font requests. All fonts must be on the font path for the print context. In the sample implementation, that font path is identical to the server's font path. That is to say, in the sample implementation there is one server-wide font path. ListFonts returns a list of fonts available along the font path. The X Logical Font Description (XLFD) 1.5 standard is supported.

Font Path Handling. In the sample print server there is one server-wide font path. At server initialization time the font path element corresponding to each printer model configured into the server is added at the front of the server's font path. This means that the font path elements for the printer internal fonts precede the font path elements for other font types. The font renderer for the printer internal font path elements inspects the client performing any font-related request, and responds differently based on whether or not the client has set a print context, and if so, then depending on the model of printer specified in the print context. If the client has not set a print context, or if the client's print context specifies a printer model other than that associated with the particular font path element, then the renderer will not find or return any fonts. If the client has set a print context and the printer specified by that print context matches the model associated with the font path element, then the renderer responds to the font request with information derived from the ".pmf" and other files (e.g. fonts.alias) in the fonts directory within that printer model's configuration directory.

Fonts built into the printer (both bitmapped and scalable). Users will generally prefer to use internal fonts for performance reasons: they already reside in the printer and do not have to be downloaded. The configuration directory for each printer containing internal fonts has a subdirectory named "fonts". This directory contains ".pmf" files defining the metrics for all glyphs in the font. The ".pmf" file format is analogous to that of a ".pcf" file with the glyphs omitted.

PCF bitmapped fonts on the server's local disk. The print server treats these fonts like ordinary X fonts. In response to a LoadFont request, the server will scale the font as required and, in the case of the PCL driver, will convert the font into a format appropriate to the printer and download the font.
QueryFont will return an X Font Structure containing metrics for the font.

Fonts from an X font server. These fonts are analogous to having PCF fonts on disk. The difference is that these scale inside the font server.

DEPENDENCIES

Print properties rely on X standard mechanisms:

- Xlib
- X protocol
- Font server technology

3.1 Systems Administration Considerations

3.1.1 Related Information

FILES

The print driver's configuration directory stores the metrics for the printer's internal fonts. It contains the font metrics in *pmf* files. The *pmf* files are identical to pcf files, but with glyphs removed. A *fonts.dir* is an index to the fonts. A *fonts.alias* file provides font names consistent with the X Logical Font Description (XLFD) Version 1.5.

4 X Printer Driver Interface

4.1 Xp Print Driver Overview

This section describes the interfaces used to integrate the print drivers into a server with the Xp extension. This section includes descriptions of the functions a driver is required to implement in order to cooperate with the Xp extension, and descriptions of some utility functions available for the convenience of driver writers. Not covered here are normal DDX driver interfaces for core X functionality.

The X Print server is simply an X server with the Xp extension. The drivers effectively provide a mapping from most X protocol rendering operations to a form understandable by a particular class of printer. The drivers are much like the hardware-specific display drivers in any other X server, but need to have some slightly different and extended capabilities in order to cooperate with the Xp extension, and with the configuration capabilities exposed via the Print Dialog Manager and its associated setup dialogs.

The print drivers are tightly coupled with the X server itself, and the initial sample print server will be based on the X11-R6 server as supplied by the X Consortium.

4.2 X Print Driver Initialization

4.2.1 Information Available during Initialization

The driver has the following practical sources of information during its initialization:

- Command line arguments - The driver's initialization routine is passed argc and argv corresponding to the arguments passed on the command line to the server.
- Information in the ScreenRec - The driver's initialization routine is passed a pointer to a ScreenRec containing potentially useful information. In particular the width, height, mmWidth, and mmHeight fields are filled in with the maximum potential dimensions prior to the calling of the driver's initialization routine.
- Driver-specific configuration files - The driver can attempt to read information from on-disk files it may expect to be in place on the system.

4.2.2 Xp Extension Initialization Interface

The Xp extension is a bit abnormal relative to other X server extensions. In particular, it is possible to have this extension be applicable on a subset of the screens of a given server. This enables a workstation with an attached printer to utilize a single process for both the X display and the Xp functions. Another somewhat unusual aspect of this extension is that the implementation of its functionality is highly device dependent, and thus each driver must support a set of entry points beyond those provided by normal DDX-compatible drivers. To these ends the driver's initialization routine (i.e. the function which might be called from dix:addScreen) must call a function to provide a pointer to the driver's InitContext function.

4.2.3 XpRegisterInitFunc

Short Description

Provides the printer-independent print server code with a pointer to the driver's routine to be called when a print context is being initialized for a printer associated with this driver.

Long Description

NAME

`XpRegisterInitFunc` :Register an InitContext function with the device-independent print server code.

SYNOPSIS

```
void XpRegisterInitFunc(ScreenPtr pScreen, int(*InitContext)(),
                        char *driverName);
```

ARGUMENTS

pScreen	Specifies a pointer to a ScreenRec indicating a screen which is prepared to support the Xp extension.
initContext	Specifies a pointer to the function to be called when a print context is initialized.
driverName	Specifies the name of the driver. The names defined in the CDE sample are: XP-RASTER, XP-PCL, and XP-POSTSCRIPT.

RETURN VALUE

None.

DESCRIPTION

The XpRegisterInitFunc provides to the printer-independent portion of the X print server a pointer to the routine to be called during the creation and initialization of a print context associated with a printer which this driver supports.

4.3 Attribute Concepts

Much of the functionality of the Xp system is controlled via the setting of various `attributes`. The attributes both describe the capabilities of the printer, and allow the user and/or the application to control many aspects of the printed output. Most of the attributes are defined in the ISO 10175 and POSIX 1387.4 standards, and are broken into a few different pools.

4.3.1 Server Attributes

These attributes are read-only to the driver. They are created and initialized when the server is initialized, and remain unchanged until the server recycles or is restarted.

Table 0-1: Server Attribute Usage

Attribute	Configuration	DDX Driver
document-attributes-supported		X
job-attributes-supported		X
locale		X
multiple-documents-supported		X

document-attributes-supported

A list of document attributes supported by the X Print Server. This list is returned as a set of whitespace-delimited attribute names.

The list of document attributes includes only attributes that are handled by the X Print Server. The full set of supported document attributes for a given printer is determined by the printer DDX driver. The driver augments the value of this server attribute, and presents the full set of supported document attributes as the value of the Printer object **document-attributes-supported** attribute. As such, applications can only query the Printer attribute and not this Server attribute in order to determine which document attributes can be used.

job-attributes-supported

A list of the job attributes supported by the X Print Server. This list is comprised of a set of whitespace-delimited attribute names.

The list of job attributes shall include only attributes that are handled by the X Print Server. The full set of supported job attributes for a given printer is determined by the printer DDX driver. The driver augments the value of this server attribute, and presents the full set of supported job attributes as the value of the Printer object **job-attributes-supported** attribute. As such, applications can only query the Printer attribute and not this Server attribute in order to determine which job attributes can be used.

locale

The value of this attribute is the locale in which the X Print Server is running. ***Check this description for use in sample implementation.***

multiple-documents-supported

The sample implementation does not support multiple documents, so this value will always be **False** in the sample implementation.

4.3.2 Printer Attributes

These attributes can only be written by the print driver. An application can only read these values, as they are a description of the capabilities of the printer and driver combination. These attributes include a description of the available and supported media types, and the supported page description languages among others.

Table 0-2: Printer Attribute Usage

Attribute	Configuration	DDX Driver
content-orientations-supported	X	X
descriptor	X	X
document-attributes-supported		X
document-formats-supported	X	X
input-trays-medium	X	X
job-attributes-supported		X
medium-source-sizes-supported	X	X
plexes-supported	X	X
printer-model	X	X
printer-name		X
printer-resolutions-supported	X	X
xp-ddx-config-file-name	X	X
xp-ddx-identifier	X	X
xp-embedded-formats-supported	X	X
xp-listfonts-modes-supported	X	X
xp-model-identifier	X	
xp-page-attributes-supported		X
xp-raw-formats-supported	X	X
xp-setup-proviso	X	X
xp-spooler-command	X	X

content-orientations-supported

The default value is determined by the DDX, and is explicitly set in the printer pool. Validation for this attribute is as described in the protocol document.

The initial value of the **content-orientations-supported** attribute is typically set by the printer vendor in the model-config file.

descriptor No default is provided for this attribute. No validation of the attribute value is performed.

The initial value of the **descriptor** attribute is typically set by the system administrator in the printer attributes file.

document-attributes-supported

The value of the **document-attributes-supported** attribute is determined by the print DDX driver.

document-formats-supported
Valid values in the sample implementation are **{ PCL 5 }** and **{ PostScript 2 }**.

The default value is determined by the DDX, and is explicitly set in the printer pool. Validation for this attribute is as described for multi-valued attributes in the protocol document. The actual set of valid document-format values varies based on the DDX.

The initial value of the **document-formats-supported** attribute is typically set by the printer vendor in the model-config file.

input-trays-medium The default value is implicitly determined to be an empty list. Validation for this attribute is as described for multi-valued attributes in the protocol document.

The initial value of the **input-trays-medium** attribute is typically specified by the system administrator in the printer attributes file.

job-attributes-supported The value of the **job-attributes-supported** attribute is determined by the print DDX driver.

medium-source-sizes-supported
The X Print Service requires that each position be specified as an integer.

The default value is explicitly set with an omitted input tray, a single medium size of **na-letter**, short edge feed direction, and a reproducible area based on 1/4 inch margins. Validation for this attribute is as described for multi-valued attributes in the protocol document. Syntax errors may cause the entire value to be considered invalid.

The initial value of the **medium-source-sizes-supported** attribute is typically set by the printer vendor in the model-config file.

plexes-supported The default value is determined by the DDX, and is explicitly set in the printer pool. Validation for this attribute is as described for multi-valued attributes in the protocol document.

The initial value of the **plexes-supported** attribute is typically set by the printer vendor in the model-config file.

printer-model The initial value of the **printer-model** attribute is typically set by the printer vendor in the model-config file.

printer-name This attribute uniquely identifies a printer on a given X Print Server. ***Needs to be edited for this sample implementation document.***

printer-resolutions-supported
The default value is determined by the DDX, and is explicitly set in the printer pool. Validation for this attribute is as described for multi-valued attributes in the protocol document.

The initial value of the **printer-resolutions-supported** attribute is typically set by the printer vendor in the model-config file.

xp-ddx-config-file-name

The name of a DDX driver-defined configuration file. Whether or not this attribute is utilized is determined by each individual driver. The file name is taken relative to the DDX configuration directory for the driver.

A default value may be assumed depending on the individual driver.

The initial value of the **xp-ddx-config-file-name** attribute is typically set by the printer vendor in the model-config file.

xp-ddx-identifier

This attribute identifies which printer DDX driver should be used for this printer. The value is a driver name provided by the DDX driver to the server, and is determined by the printer driver developer.Valid values in the sample implementation are **XP-PCL**, **XP-POSTSCRIPT**, and **XP-RASTER**.

The default value in the sample implementing is implicitly taken to be **XP-POSTSCRIPT** by the X Print Server. Validation for this attribute is as described for single valued attributes in the protocol document.

The initial value of the **xp-ddx-identifier** attribute is typically set by the printer vendor in the model-config file.

xp-embedded-formats-supported

For the sample implementation, valid values may include **{EPS}**, **{PostScript 2}**, **{PCL 5}**, and **{HPGL 2}**.

The default value is determined by the DDX, and is explicitly set in the printer pool. Validation for this attribute is as described for multi-valued attributes in the protocol document. The actual set of valid document-format values varies based on the DDX.

The initial value of the **xp-embedded-formats-supported** attribute is typically set by the printer vendor in the model-config file.

xp-listfonts-modes-supported

Valid listfonts mode values in the sample implementation are **xp-list-internal-printer-fonts** and **xp-list-glyph-fonts**.

The default value is determined by the DDX, and is explicitly set in the printer pool. Validation for this attribute is as described for multi-valued attributes in the protocol document.

The initial value of the **xp-listfonts-modes-supported** attribute is typically set by the printer vendor in the model-config file.

xp-model-identifier

The X Print Service allows specification of DPA Printer object attribute definitions across two configuration files: the attributes/printer file and the models/*/model-config file. The **xp-model-identifier** is defined to provide an association between these two files.

The **xp-model-identifier** attribute value is specified in the attributes/printer file. This value corresponds to the name of a model subdirectory under the models configuration directory. The X Print Service obtains initial printer attributes from the model-config file in the named model subdirectory.

The value consists of the manufacturer and model of the printer. It is recommended that the value consist of only uppercase characters, since either the model identifier or the printer name (typically lowercase) may function as a qualifier for attribute definitions within the configuration files. Valid characters for the value of **xp-model-identifier** are
a-**z**, **A**-**Z**, **0**-**9**, **_**, and **-**.

There is no default value for this attribute. Validation for this attribute is as described for single valued attributes in the protocol document. If a model-config file cannot be found based on the value, the value is considered invalid.

The initial value of the **xp-model-identifier** attribute is typically specified by the system administrator in the printer attributes file.

xp-page-attributes-supported

The value of the **xp-page-attributes-supported** attribute is determined by the print DDX driver.

xp-raw-formats-supported

The default value is determined by the DDX, and is explicitly set in the printer pool. Validation entails syntax checking only.

The initial value of the **xp-raw-formats-supported** attribute is typically set by the printer vendor in the model-config file.

xp-setup-proviso ***Hadn't marked up any information to include in sample implementations doc.***

This attribute indicates whether or not a required attribute or set of attributes must be set (typically via user interaction with Print Dialog Manager) prior to commencing the print job.

Valid values for this attribute are **xp-setup-mandatory** and **xp-setup-optional**. If this attribute is not specified, **xp-setup-optional** is assumed.

The initial value of the **xp-setup-proviso** attribute is typically set by the printer vendor in the model-config file.

xp-spooler-command

This attribute can be used to override the default spooling operation performed by the X Print Server. The value consists of a command plus any command line options. The resulting print file is passed to the command via stdin.

The command line may contain references to a predefined set of variables, that will be expanded by the server. The variables are:

%printer-name%	The name of the printer
%copy-count%	The value of the **copy-count** attribute
%job-name%	The value of the **job-name** attribute
%options%	The value of the **xp-spooler-command-options** attribute

The initial value of the **xp-spooler-command** attribute is typically not specified.

4.3.3 Document Attributes

These attributes describe such things as the media to use for the document, the “plex” to use, and the orientation (i.e. portrait or landscape). These attributes can be read and written by both the application and the driver. Default values for these attributes are set by the driver (possibly using the provided utility routines) when a new print context is initialized. The user or application can modify these attributes to communicate such choices to the driver. It is the driver’s responsibility to communicate these attributes to the specific printer, presumably by embedding the appropriate page description language strings in the output. Changes in these attributes may cause the driver to perform operations such as resizing a window referenced by a subsequent **`StartPage`** to fit the specified media size or orientation.

The following table shows where querying and / or setting a document attribute value is supported within the X Print Service.

Attribute	Configuration	DDX Driver
content-orientation	X	X
copy-count	X	X
default-printer-resolution	X	X
default-input-tray	X	X
default-medium	X	X
document-format	X	X
plex	X	X
xp-listfonts-modes	X	X

content-orientation The default value is implicitly determined by the DDX driver to be the first entry in the value of the **content-orientations-supported** printer attribute. Validation for this attribute is as described for single valued attributes in the protocol document. The value must appear in the **content-orientations-supported** attribute value to be considered valid.

copy-count Specifies the number of copies of this document to print.

No information indicated for this attribute in sample implementation in original edit.

The default value is implicitly taken to be **1** by the X Print Server. Validation for this attribute is as described for single valued attributes in the protocol document. The value must be a positive integer.

default-printer-resolution

The default value is implicitly determined by the DDX driver to be the first entry in the value of the **printer-resolutions-supported** printer attribute. Validation for this attribute is as described for single valued attributes in the protocol document. The value must appear in the **printer-resolutions-supported** attribute value to be considered valid.

default-input-tray No default is assumed for this attribute, since the **default-medium** attribute takes precedence. Validation for this attribute is as described for single valued attributes in the protocol document. The input tray must be included in the **medium-source-sizes-supported** printer attribute value (note: if an entry in **medium-source-sizes-supported** omits the input tray specifier, then the input tray value specified for **default-input-tray** will be considered valid, provided of course that it is listed as one of the valid values above).

default-medium The default value is implicitly determined by the DDX driver, provided the **default-input-tray** attribute is unspecified. The default will correspond to the first medium size found in the value of the **medium-source-sizes-supported** printer attribute. Validation for this attribute is as described for single valued attributes in the protocol document. The value must appear in the **medium-source-sizes-supported** attribute value to be considered valid.

document-format Valid values in the sample implementation are **{ PCL 5 }** and **{ PostScript 2 }**.

The default value is determined by the DDX, and is explicitly set in the printer pool. Validation for this attribute is as described for single valued attributes in the protocol document. The value must appear in the **document-formats-supported** printer attribute value to be considered valid.

plex The default value is implicitly determined by the DDX driver to be the first entry in the value of the **plexes-supported** printer attribute. Validation for this attribute is as described for single valued attributes in the protocol document. The value must appear in the **plexes-supported** attribute value to be considered valid.

xp-listfonts-modes The default value is implicitly determined by the DDX driver to be the all of the listfonts modes specified in the **xp-listfonts-modes-supported** printer attribute. Validation for this attribute is as described for multi-valued attributes in the protocol document. Each listfonts mode value must appear in the **xp-listfonts-modes-supported** attribute value to be considered valid.

4.3.4 Page Attributes

These are a subset of the document attributes which can be varied on a page-by-page basis. This allows, for example, an application to print a particular page in landscape orientation in the middle of a document which is otherwise in portrait orientation. These attributes can be read and written by both the application and the driver. It is the driver's responsibility to communicate these attributes to the specific printer, typically by embedding the appropriate page description language strings in the output. Changes in these attributes may cause the driver to perform operations such as resizing a window to fit the specified media size or orientation when **`StartPage`** is executed.

Validation of page attributes is the same as for document attributes.

The following table shows where querying and / or setting a page attribute value is supported within the X Print Service.

Attribute	Configuration	DDX Driver
content-orientation		X
default-printer-resolution		X
default-input-tray		X
default-medium		X
plex		X
xp-listfonts-modes	X	X

content-orientation Specifies the orientation to be used for this page. Valid values are: **portrait**, **landscape**, **reverse-portrait**, and **reverse-landscape**.

default-printer-resolution
Specifies the resolution in dots per inch to be used for this page.

default-input-tray No default is assumed for this attribute, since the **default-medium** attribute takes precedence. Validation for this attribute is as described for single valued attributes in the protocol document. The input tray must be included in the **medium-source-sizes-supported** printer attribute value (note: if an entry in **medium-source-sizes-supported** omits the input tray specifier, then the input tray value specified for **default-input-tray** will be considered valid, provided of course that it is listed as one of the valid values above).

default-medium The default value is implicitly determined by the DDX driver, provided the **default-input-tray** attribute is unspecified. The default will correspond to the first medium size found in the value of the **medium-source-sizes-supported** printer attribute. Validation for this attribute is as described for single valued attributes in the protocol document. The value must appear in the **medium-source-sizes-supported** attribute value to be considered valid.

plex The default value is implicitly determined by the DDX driver to be the first entry in the value of the **plexes-supported** printer attribute. Validation for this attribute is as described for single valued attributes in the protocol document. The value must appear in the **plexes-supported** attribute value to be considered valid.

xp-listfonts-modes The default value is implicitly determined by the DDX driver to be the all of the listfonts modes specified in the **xp-listfonts-modes-supported** printer attribute. Validation for this attribute is as described for multi-valued attributes in the protocol document. Each listfonts mode value must appear in the **xp-listfonts-modes-supported** attribute value to be considered valid.

4.3.5 Job Attributes

These control the functioning of the spooler itself, allowing the specification of items such as the banner page contents. These attributes can be read and written by both the application and the driver, however the driver should be able to be blissfully unaware of these attributes if the driver chooses to utilize the **`XpSub-`**

`mitJob` call documented below. These attributes are ignored if the client specifies the *`save_data`* field to be **`XPGetData`** in its call to **`StartJob`**.

The following table shows where querying and/or setting a job attribute value is supported within the X Print Service.

Table 0-3: Job Attribute Usage

Attribute	Configuration	DDX Driver
job-name	X	X
job-owner		X
notification-profile	X	X
xp-setup-state	X	X
xp-spooler-command-options	X	X
xp-spooler-command-results	X	X

job-name No default is provided for this attribute. No validation of the attribute value is performed.

job-owner This attribute identifies the human owner of the print job. ***Anything else or anything different here?***

notification-profile The default value is implicitly taken to indicate that no message be sent. No validation of the attribute value is performed.

xp-setup-state The initial value of **xp-setup-state** is typically not specified in the job attributes configuration file. If **xp-setup-state** is unspecified, the default value is **xp-setup-incomplete**.

xp-spooler-command-options
No default is provided for this attribute. No validation of the attribute value is performed.

xp-spooler-command-results
Nothing was marked for the sample implementation document...

A free form text string that will contain the spooler command output that would otherwise appear on a terminal (e.g. stderr and stdout). This text may be useful to present to the user to allow tracking of the resulting spooler job. Applications should retrieve this value following receipt of the **XPEndJobNotify** event.

4.4 Attribute Store and Spooler Interface Functions

The functions described in this section are intended as conveniences for the drivers in implementing their **`GetAttribute`**, **`SetAttribute`**, and **`EndJob`** functions. The DDX drivers are *not required* to use the functions described in this section, but it is *strongly recommended* that they do so. These functions insulate the driver from the underlying print spooling system, and are intended to allow drivers developed for the initial sample server to function in environments where more capable printing systems (e.g. Palladium) are in place. These functions do not attempt to mirror the API afforded by the Palladium system, but should provide sufficient capabilities to allow a driver access to all attributes accessible via a Palladium based-system. A driver which chooses not to use these functions is unlikely to integrate smoothly into a Palladium-based environment. Note that the Attribute Store functions do no error checking of Printer, Document, or Page attributes, as such checking is left entirely in the hands of the driver. Driver writers are advised to write their context initialization code such that it gets the attributes, and edits them prior to responding to the first **`GetAttributes`** request from a client. A store of Job attributes is maintained and error-checked internally by the attribute store.

From a driver's perspective the Attribute Store consists of four distinct collections of attributes: Printer Attributes, Document Attributes, Job Attributes, and PageAttributes. The driver has write access to all of these attributes, even though the protocol specifies that the Printer Attributes are read-only. This write access allows the driver to modify the attributes to describe the capabilities it possesses more accurately. For example, immediately after initialization of the Attribute Store the Printer Attributes may contain an entry stating that the document-formats-supported include both PCL and PostScript (e.g. for a HP-DeskJet 1600C). If the driver only supports a single document-format then the driver should change the document-formats-supported attribute to reflect the fact that it only supports its single document-format. There are separate attribute stores maintained on a per-print-context basis. All strings are in the form accepted by **`XrmGetStringDatabase`**().

4.4.1 XpInitAttributes

Short Description

Causes the Attribute Store to be initialized. In the initial sample implementation, this causes the Attribute Store to read the initial attribute values from the on-disk configuration files if they have not been read previously. The driver typically calls this routine in the function invoked by **`InitPrintContext`**. The attributes are expected to carry forward unchanged between jobs within the same print context, but the closing and re-initializing of a client's print context should result in freshly initialized attributes. To effect this, a driver should call **`XpInitAttributes`** once and only once for each **`InitPrintContext`** request. After the Attribute Store is initialized for a client, any changes made to the attribute store for the client should remain intact until the print context is destroyed.

Long Description

NAME

XpInitAttributes - Initialize the attributes for a particular print context.

SYNOPSIS

```
void XpInitAttributes(PrintContextPtr pContext);
```

ARGUMENTS

`pContext`	Specifies a pointer to the print context for which the attributes are desired.

RETURN VALUE

None.

DESCRIPTION

`XpInitAttributes` initializes the attribute store associated with a particular print context. It is expected that a driver will call this function upon receipt of an **`InitContext`** request. A driver must call **`XpInitAttributes`** prior to calling either **`XpGetAttributes`**, **`XpGetOneAttribute`**, **`XpAugmentAttributes`** or **`XpSetAttributes`** for a given context.

4.4.2 XpGetOneAttribute

Short Description

Retrieves the current value of a specified attribute from the Attribute Store.

Long Description

NAME

XpGetOneAttribute - Get the current value of the specified attribute for a given print context.

SYNOPSIS

```
char *XpGetOneAttribute(PrintContextPtr pContext, XpAttrType pool,
                        char *attributeName);
```

ARGUMENTS

pContext	Specifies a pointer to the print context for which the attribute value is desired.
pool	Specifies the pool of the attribute which is desired. Possible values are XPJobAttr, XPDocAttr, XPPrinterAttr, XPPageAttr, XPServerAttr.
attributeName	Specifies the name of the attribute for which the value is desired.

RETURN VALUE

A pointer to a character string containing the value of the specified attribute, or NULL if the attribute does not exist in the attribute store. The returned string must not be freed.

DESCRIPTION

The XpGetOneAttribute function returns a string containing the value of the specified attribute as a string.

4.4.3 XpGetAttributes

Short Description

Retrieves the current contents of the specified set of attributes in its entirety from the Attribute Store.

Long Description

NAME

XpGetAttributes - Obtain the current contents of the specified attribute set for a given print context.

SYNOPSIS

```
char *XpGetAttributes(PrintContextPtr pContext, XpAttrType pool);
```

ARGUMENTS

pContext	Specifies a pointer to the print context for which the attributes are desired.
pool	Specifies the pool of the attribute which is desired. Possible values are XPJobAttr, XPDocAttr, XPPrinterAttr, XPPageAttr, XPServerAttr.

RETURN VALUE

A pointer to a character string containing the current set of attributes for the print context. It is the caller's responsibility to free the string when it is no longer needed.

DESCRIPTION

The XpGetAttributes function returns a string containing the current attribute names and values. It is expected that drivers will use this function in order to implement the GetAttributes function.

4.4.4 XpGetMediumDimensions

Short Description

Retrieves the dimensions of the medium currently selected for the document associated with a particular print context from the Attribute Store.

Long Description

NAME

XpGetMediumDimensions - Obtain the dimensions of the medium for a document associated with a particular print context.

SYNOPSIS

```
void XpGetMediumDimensions(PrintContextPtr pContext, CARD16 *width,
   CARD16 *height);
```

ARGUMENTS

pContext	Specifies a pointer to the print context for which the attributes are desired.
width	Returns the width of the medium.
height	Returns the height of the medium.

RETURN VALUE

None.

DESCRIPTION

The XpGetMediumDimensions function provides a convenient means for the driver to determine the overall dimensions of the medium specified for the current (or first) page of the document in the job associated with the specified print context. The medium dimensions returned are computed from the value of the **`default-medium`** attribute, or if it is not specified, from the **`default-input-tray`** and **`input-trays-medium`** attributes. If neither of these attribute sets is valid, then XpGetMediumDimensions returns values corresponding to the first entry in the list of **`medium-source-sizes-supported`**. The returned dimensions are in pixel units, through the use of the **`content-orientation`** and **`default-resolution`** document attributes.

4.4.5 XpGetReproductionArea

Short Description

Retrieves from the Attribute Store the net reproducible area for the document associated with a particular print context.

Long Description

NAME

XpGetReproductionArea - Obtain the dimensions and position of the reproducible area for a document associated with a particular print context.

SYNOPSIS

`void XpGetReproductionArea(PrintContextPtr pContext, xRectangle *pRect);`

ARGUMENTS

`pContext`	Specifies a pointer to the print context for which the attributes are desired.
`pRect`	Specifies a pointer to a rectangle which will return the reproducible area.

RETURN VALUE

None.

DESCRIPTION

The XpGetReproductionArea function provides a convenient means for the driver to determine the dimensions of the reproducible area for the current (or first) page of the document in the job associated with the specified print context. The reproducible area differs from the medium dimensions in that the reproducible area has had subtracted from it any regions of the medium which cannot be printed on, and all regions which the printer mechanism cannot mark. The returned dimensions are in units of pixels, through the use of the **`content-orientation`** and **`default-resolution`** document attributes. The relevant medium is determined from the contents of either the **`default-medium`** attribute, or if that is not defined the **`default-input-tray`** and **`input-trays-medium`** attributes. The **`medium-source-sizes-supported`** attribute is used to determine the reproducible area for this medium. If insufficient information is available from the attributes then the values returned will correspond to North American Letter media with a one-quarter-inch non-reproducible border.

4.4.6 XpAugmentAttributes

Short Description

Augments the values of the specified attribute class.

Long Description

NAME

XpAugmentAttributes - Augment the value of a particular attribute class for a given print context.

SYNOPSIS

```
void XpAugmentAttributes(PrintContextPtr pContext, XpAttrType pool,
                         char *attributes);
```

ARGUMENTS

pContext	Specifies a pointer to the print context for which the attributes are desired.
pool	Specifies the pool of the attribute which is desired. Possible values are XPJobAttr, XPDocAttr, XPPrinterAttr, XPPageAttr.
attributes	Specifies the names and values of the attributes.

RETURN VALUE

None.

DESCRIPTION

The XpAugmentAttributes function adds the supplied attributes to the specified attribute class. If a supplied attribute already exists, then its new value is taken from the supplied list of attributes.

4.4.7 XpSetAttributes

Short Description

Stores a new set of attributes for a particular class of attributes.

Long Description

NAME

XpSetAttributes - Set new attributes and values for a given print context.

SYNOPSIS

```
void XpSetAttributes(PrintContextPtr pContext, XpAttrType pool, char
   *attributes);
```

ARGUMENTS

pContext	Specifies a pointer to the print context for which the attributes are to be set.

pool	Specifies the pool of the attribute which is desired. This is one of XPJobAttr, XPDocAttr, XPPrinterAttr, XPPageAttr, XPServerAttr.
attributes	A string of all the attributes and values for the specified class.

RETURN VALUE

None.

DESCRIPTION

The XpSetPrintAttributes function accepts a string containing the new attribute names and values. It is expected that drivers will use this function in order to implement the SetAttributes function.

4.4.8 XpSubmitJob

Short Description

Requests that a particular job file be submitted to the spooler with an associated set of job attributes.

Long Description

NAME

XpSubmitJob - Submit a file to the print spooler.

SYNOPSIS

```
void XpSubmitJob(char *fileName, PrintContextPtr pContext);
```

ARGUMENTS

fileName	Specifies the name of the file to be submitted for printing.
pContext	Specifies the print context associated with the print job.

RETURN VALUE

None.

DESCRIPTION

XpSubmitJob takes whatever steps are necessary to submit the specified file to the underlying spooling system with the specified job attributes. It is expected that drivers will call this function from within their EndJob functions. In the initial sample implementation this function invokes the lp command to spool the job.

4.4.9 XpFreeAttributes

Short Description

Frees the storage associated with the attributes for a specified XpContext.

Long Description

NAME

XpFreeAttributes - Free the memory associated with the attributes for a particular XpContext.

SYNOPSIS

`void XpFreeAttributes(PrintContextPtr pContext);`

ARGUMENTS

`pContext`	Specifies the print context associated with the print job.

RETURN VALUE

None.

DESCRIPTION

`XpFreeAttributes` frees the memory associated with the attributes for the specified print context. **`XpFreeAttributes`** should be called from the driver's **`DestroyContext`** function.

4.5 Xp Extension Functions

A print driver *must* implement the following set of functions which provide the underpinnings for the extension requests defined by the Xp extension. The InitContext call is the function which was passed to XpRegisterInitFunc, while the other functions are called via function pointers stored in each PrintContext. A pointer to a PrintContext is passed to each of these routines, and has the following structure:

```
typedef struct _xpprintfuncs {
int versionNumber;
int (*StartJob)(); /* pPrintContext, saveData */
int (*EndJob)(); /* pPrintContext, cancel */
int (*StartDoc)(); /* pPrintContext */
int (*EndDoc)(); /* pPrintContext, cancel */
int (*StartPage)(); /* pPrintContext, pWin */
int (*EndPage)(); /* pPrintContext, pWin, cancel */
int (*PutDocumentData)(); /* pPrintContext, pWin, pData,len_data, pFmt,
 pOpt */
int (*GetDocumentData)(); /* pPrintContext, client, maxBufferSize */
int (*DestroyContext)(); /* pPrintContext */
char *(*GetAttributes)(); /* pPrintContext, class */
char *(*GetOneAttribute)(); /* pPrintContext, class, attribute */
int (*SetAttributes)(); /* pPrintContext, class, pData */
int (*AugmentAttributes)(); /* pPrintContext, class, pData */
int (*GetMediumDimensions(); /* pPrintContext, pWidth, pHeight */
int (*GetReproducibleArea(); /* pPrintContext, pRect */
} XpDriverFuncs, *XpDriverFuncsPtr;

typedef struct _XpContext {
        XID contextID;
        char *printerName;
        int screenNum;
        struct _XpClient *clientHead; /* list of clients */
```

```
        CARD32 state;
        VisualID pageWin;
        DevUnion *devPrivates;
        XpDriverFuncs funcs;
} XpContextRec, *XpContextPtr;
```

When the driver's InitContext function is called it is free to inspect the printerName field of the XpContext, and is required to fill in all of the function pointers in the embedded XpDriversFuncs structure.

4.5.1 InitContext

Short Description

Provides pointers to functions implementing the various printing related operations for the specified context.

Long Description

NAME

InitContext - Initialize the contents of the supplied XpContext.

SYNOPSIS

`int InitContext(PrintContextPtr pContext);`

ARGUMENTS

pContext	Specifies a pointer to the print context in which the print data will be generated.

RETURN VALUE

`Success`, or a value indicating the error (e.g. **`BadAlloc`**).

DESCRIPTION

The **`InitContext`** function supplies the driver with the name of the printer to be used in subsequent print jobs in the specified print context. The driver is expected to fill in the function pointers within the XpContext, and to initialize the attribute store for the print context at the time this function is called. This enables an application to then query the printer attributes and receive accurate information. The driver should also initialize any per-context data it wishes to maintain.

4.5.2 DestroyContext

Short Description

Notifies the driver that the context is no longer in use, and any associated data should be freed.

Long Description

NAME

DestroyPrintContext - Release any driver resources allocated for the specified print context.

SYNOPSIS

```
int DestroyContext(PrintContextPtr pContext);
```

ARGUMENTS

pContext	Specifies a pointer to the print context which is being destroyed.

RETURN VALUE

Success, or a value indicating the error (e.g. **BadAlloc**).

DESCRIPTION

The **DestroyContext** function provides the driver an opportunity to clean up any state or resources it has allocated in support of the specified print context. **XpFreeAttributes** should be called from this function if the attribute storage facilities have been used to create the attributes store for this context.

4.5.3 StartJob

Short Description

Implements the driver level functionality of the XpStartJob extension request.

Long Description

NAME

StartJob - Begin a new print job associated with a particular window.

SYNOPSIS

```
int StartJob(PrintContextPtr pContext, XPSaveData sendData);
```

ARGUMENTS

pContext	Specifies a pointer to the print context for which the print job is starting.
sendData	Specifies whether the resulting print data is to be sent to a client, and if so, the driver must be prepared to call XpWriteClientData when there is print output data available to be sent.

RETURN VALUE

Success if no errors are encountered, otherwise a value indicating the error (e.g. **BadAlloc**).

DESCRIPTION

The **StartJob** function will typically check for and delete any previously created print data associated with the print context, and will create storage space for the new print job. The *sendData* parameter indicates that a client will receive the data, rather than having the data submitted to the spooling system. The driver is then required to call XpSendClientData() when there is print data available. The driver may assume that there will be no changes to the Job attributes for this context after the StartJob function has been called.

4.5.4 EndJob

Short Description

Implements the driver level functionality of the XpEndJob extension request.

Long Description

NAME

EndJob - Ends the print job associated with a particular window, and submits the job to the printer.

SYNOPSIS

```
int EndJob(PrintContextPtr pContext, Boolean cancel);
```

ARGUMENTS

pContext	Specifies a pointer to the print context for which the print job is ending.
cancel	A TRUE value indicates that the job is to be canceled, and any remaining print data discarded rather than submitted to the spooler or returned to the client.

RETURN VALUE

Success if no errors are encountered, otherwise a value indicating the error (e.g. **BadAlloc**).

DESCRIPTION

The **EndJob** function typically submits the job to the spooler. If *cancel* is TRUE then any remaining print data is discarded, and if necessary the print job is canceled. If print data has been sent to a client via XpSend-ClientData(), then XpSendClientData should be called with the "status" parameter set to either END or CANCEL, depending on the value of the *cancel* flag. At that point the driver should be able to properly accept a StartJob request on the same print context.

4.5.5 StartDoc

Short Description

Implements the driver level functionality of the XpStartDoc extension request.

Long Description

NAME

StartDoc - Begins a new document within the print job associated with a window.

SYNOPSIS

```
int StartDoc(PrintContextPtr pContext, XPDocumentType type);
```

ARGUMENTS

pContext	Specifies a pointer to the print context for which a new document is starting.

type	Specifies the type of the document. Possible values are XPDocRaw or XPDocNormal. A value of XPDocRaw indicates that the data is to be passed through unmodified by the print server, and only PutDocumentData calls will be accepted after such a StartDoc.

RETURN VALUE

Success if no errors are encountered, otherwise a value indicating the error (e.g. **BadAlloc**).

DESCRIPTION

The **StartDoc** function is primarily a place holder for any necessary functionality needed when and if the Xp Service is implemented on top of a print spooling system which supports multiple documents in a job, such as one compliant with POSIX 1387.4. The driver is guaranteed to receive a **StartDoc** call with *type* equal to **XpDocNormal** prior to receiving a **StartPage**. If the driver receives a **StartDoc** call with *type* equal to **XpDocRaw** it can assume it will not receive a **StartPage** prior to the **EndDoc** for that document. The driver may assume that there will be no changes to the document attributes for the specified context after this function has been called.

4.5.6 EndDoc

Short Description

Implements the driver level functionality of the XpEndDoc extension request.

Long Description

NAME

EndDoc - Ends a document within the print job associated with a print context.

SYNOPSIS

```
int EndDoc(PrintContextPtr pContext, Boolean cancel);
```

ARGUMENTS

pContext	Specifies a pointer to the print context for which a document is ending.
cancel	Indicates whether the current document is to be canceled, and any remaining (i.e. buffered) data for this document discarded.

RETURN VALUE

Success if no errors are encountered, otherwise a value indicating the error (e.g. **BadAlloc**).

DESCRIPTION

The **EndDoc** function is essentially a place holder for any necessary functionality needed when and if the Xp Service is implemented on top of a print spooling system capable of supporting multiple documents in a single job, such as one compliant with POSIX 1387.4. A driver is guaranteed to receive an **EndDoc** prior to an **EndJob**.

4.5.7 StartPage

Short Description

Implements the driver level functionality of the XpStartPage extension request.

Long Description

NAME

StartPage - Begins a new page within the print job, and associates the print context with a window.

SYNOPSIS

```
int StartPage(PrintContextPtr pContext, Window pWin);
```

ARGUMENTS

pContext	Specifies a pointer to the print context for which a new page is starting.
pWin	Specifies a pointer to the window to be used as the top-most window in the printed page.

RETURN VALUE

`Success` if no errors are encountered, otherwise a value indicating the error (e.g. **`BadAlloc`**).

DESCRIPTION

The **`StartPage`** function discards any previously created data for any previous page, allocates any storage which may be necessary for a new page, resizes the window to match the size of the medium, clears the window and all descendent windows to their backgrounds, and adds any necessary page header data to the contents of this page. Such header data is generally determined by the values of the page attributes. The driver may assume that there will be no changes to the Page attributes for the specified context after this call.

4.5.8 EndPage

Short Description

Implements the driver level functionality of the XpEndPage extension request.

Long Description

NAME

EndPage - Ends a page within the print job associated with a window.

SYNOPSIS

```
int EndPage(PrintContextPtr pContext, Window pWin, Boolean cancel);
```

ARGUMENTS

pContext	Specifies a pointer to the print context for which a new page is starting.
pWin	Specifies a pointer to the top-most window for the page.

cancel	A value of TRUE indicates that any remaining page data should be discarded rather than being submitted as part of the current document.

RETURN VALUE

Success if no errors are encountered, otherwise a value indicating the error (e.g. **BadAlloc**).

DESCRIPTION

The **EndPage** function adds any necessary trailing information for the page, and adds the page data to the print job associated with the print context. The trailer data is determined by the values of the page attributes. If *cancel* is TRUE, then any buffered page data should be discarded rather than being included in the current document and job.

4.5.9 PutDocumentData

Short Description

Implements the driver level functionality of the XpPutDocumentData extension request.

Long Description

NAME

PutDocumentData - Adds application supplied data to the print document associated with a print context.

SYNOPSIS

```
int PutDocumentData(
   PrintContextPtr pContext;
   Window pWin;
   char *pData;
   int len_data;
   char *pFmt,
   char *pOpt);
```

ARGUMENTS

pContext	Specifies a pointer to the print context defining the print job.
pWin	Specifies a pointer to the window into which the data is to be placed.
pData	Points to the data to be added to the print job.
len_data	Specifies the length in bytes of the data to be added to the print job.
pFmt	Points to a string describing the format of the data (e.g. PCL5).
pOpt	Points to a string describing driver-specific options for the data.

RETURN VALUE

Success if no errors are encountered, otherwise a value indicating the error (e.g. **BadAlloc**).

DESCRIPTION

The **PutDocumentData** function provides a means for an application to supply printer device dependent data of its own creation. The data is added to the document associated with the specified print context. The driver may, if it desires, modify or interpret the data based on the specified format, options, and known printer characteristics. As an example, a driver may choose to support DeviceData formats other than those which are supported by the printer itself by translating the data into a format understood by the printer. If the **PutDocumentData** is sent following a **StartDoc**(*printContext*, **XPDocNormal)**, then the driver is expected to provide any generally needed page description language header data necessary to embed the supplied data within the boundaries of the specified window. However, if the **PutDocument-Data** is sent after a **StartDoc**(*printContext*, **XPDocRaw**), then the driver is expected to pass the data straight through to the spooler with no additions or modifications.

The window given to the PutDocumentData function specifies the size and location of the embedded data. It may not be possible for the driver to clip the embedded data to take into account other windows which occlude the given window.

4.5.10 GetDocumentData

Short Description

Informs the driver which client should receive the generated document data for the print job associated with the specified print context.

Long Description

NAME

GetDocumentData - Establish which client should receive data generated by print jobs in a print context.

SYNOPSIS

```
int GetDocumentData(PrintContextPtr pContext, ClientPtr client, int
   maxBufferSize);
```

ARGUMENTS

pContext	Specifies a pointer to the print context for which the print data is desired.
client	Specifies the client which is to receive all generated document data for the job associated with the specified print context.
maxBufferSize	Specifies the maximum amount of data the client wishes to receive in a single reply.

RETURN VALUE

Success if the driver was able to set its state in preparation for returning the document data, else a code indicating the problem (e.g. BadAlloc).

DESCRIPTION

The GetDocumentData function allows the driver to prepare for sending document data for a job to the specified client. If the receiving client is unable to read back the generated data quickly enough to keep up with the rate of data generation the driver is free to suspend the processing of further requests from clients making rendering requests within a print context.

4.5.11 GetAttributes

Short Description

Returns the current contents of the specified set of attributes.

Long Description

NAME

GetAttributes - Obtain the current contents of the specified attribute set for a given print context.

SYNOPSIS

```
char *GetAttributes(PrintContextPtr pContext, XpAttrType pool);
```

ARGUMENTS

pContext	Specifies a pointer to the print context for which the attributes are desired.
pool	Specifies the pool of the attribute which is desired. This is one of XPJobAttr, XPDocAttr, XPPrinterAttr, XPPageAttr, XPServerAttr.

RETURN VALUE

A pointer to a character string containing the current set of attributes for the print context. It is the caller's responsibility to free the string when it is no longer needed. GetAttributes returns a NULL pointer in the case of an allocation error (i.e. **BadAlloc**), and returns a pointer to an empty string if the requested attribute store is empty.

DESCRIPTION

The GetAttributes function returns a string containing the current attribute names and values for the specified attribute class. It is expected that drivers will use the **XpGetAttributes** function to implement this function.

4.5.12 GetOneAttribute

Short Description

Returns the value of the specified attribute within a particular attribute pool for a print context.

Long Description

NAME

GetOneAttribute - Obtain the current value of a particular attribute.

SYNOPSIS

```
char *GetOneAttributes(PrintContextPtr pContext, XpAttrType pool, char
   *attr);
```

ARGUMENTS

pContext	Specifies a pointer to the print context for which the attributes are desired.

pool — Specifies the pool of the attribute which is desired. Possible values are XPJobAttr, XPDocAttr, XPPrinterAttr, XPPageAttr, XPServerAttr.

attr — Specifies the attribute for which the value is desired.

RETURN VALUE

A pointer to a character string containing the value of the attribute for the print context. The caller must not free the returned string. GetOneAttribute returns a NULL pointer in the case of an allocation error (i.e. **BadAlloc**), and returns a pointer to an empty string if the requested attribute is not defined.

DESCRIPTION

The GetOneAttribute function returns a string containing the values for the specified attribute class and attribute within the specified print context. It is expected that drivers will use the **XpGetOneAttribute** function to implement this function.

4.5.13 AugmentAttributes

Short Description

Augments the contents of the specified set of attributes.

Long Description

NAME

AugmentAttributes - Augment the contents of the specified attribute set for a given print context.

SYNOPSIS

```
int AugmentAttributes(PrintContextPtr pContext, XpAttrType pool, char
   *attributes);
```

ARGUMENTS

pContext — Specifies a pointer to the print context for which the attributes are to be augmented.

pool — Specifies the pool of the attribute which is desired. Possible values are XPJobAttr, XPDocAttr, XPPrinterAttr, XPPageAttr.

attributes — Specifies the names and values of some attributes for the above-specified class.

RETURN VALUE

Success if no error is detected, otherwise a value indicating the error (e.g. **BadAlloc**, **BadAttribute**).

DESCRIPTION

The **AugmentAttributes** function adds the specified attributes to the store of the specified attribute class. If a supplied attribute already exists in the store, then the value supplied in this call will become the value of that attribute.

4.5.14 SetAttributes

Short Description

Sets the contents of the specified set of attributes.

Long Description

NAME

SetAttributes - Set the contents of the specified attribute set for a given print context.

SYNOPSIS

`int SetAttributes(PrintContextPtr pContext, XpAttrType pool, char *attributes);`

ARGUMENTS

`pContext`	Specifies a pointer to the print context for which the attributes are desired.
`pool`	Specifies the pool of the attribute which is desired. Possible values are XPJobAttr, XPDocAttr, XPPrinterAttr, XPPageAttr.
`attributes`	Specifies the names and values of all the attributes for the above-specified class.

RETURN VALUE

`Success` if no error is detected, otherwise a value indicating the error (e.g. **`BadAlloc`**, **`BadAttribute`**).

DESCRIPTION

The SetAttributes function replaces the existing attributes and values (if any) with those contained in the *`attributes`*. It is expected that drivers will use the **`XpSetAttributes`** function to implement this function.

4.6 Xp Utility and Convenience Functions

The functions described in this section are intended as conveniences for the drivers.

4.6.1 XpSendData

Short Description

Send printer data to any client which has performed an XpGetDocumentData call for a specific print context.

Long Description

NAME

XpSendData - Send print data to any interested client.

SYNOPSIS

```
int XpSendData(PrintContextPtr pContext, ClientPtr client, char *data,
   int len_data);
```

ARGUMENTS

pContext	Specifies a pointer to the print context for which the attributes are desired.
client	A pointer to the client which is to receive the data.
data	A pointer to the print data to be sent to the interested client.
len_data	Specifies the length in bytes of the print data.

RETURN VALUE

Success if no error is detected, otherwise a value indicating the error (e.g. **BadAlloc**, **BadAttribute**).

DESCRIPTION

The XpSendData function sends the supplied data to the specified client. The client should be that which has performed an XpGetDocumentData call for the specific print context. The returned value indicates any error which occurred during the sending of the data. This function takes care of formatting the data into GetDocumentDataReply structures including byte-swapping reply header information.

4.6.2 XpAllocateContextPrivateIndex

Short Description

Allocate a context-private index for use by the driver.

Long Description

NAME

XpAllocateContextPrivateIndex - Allocate a context-private index for use by the driver.

SYNOPSIS

```
int XpAllocateContextPrivateIndex();
```

ARGUMENTS

RETURN VALUE

An index value which can be used in a subsequent call to XpAllocateContextPrivate.

DESCRIPTION

The XpAllocateContextPrivateIndex function returns an index into the context devPrivates array for use by the caller. This index may be passed to XpAllocateContextPrivate to have the printer-independent portion of the server automatically allocate a fixed amount of memory with each context.

4.6.3 XpAllocateContextPrivate

Short Description

Inform the printer-independent code of the amount of memory to be allocated with each context for use by the caller.

Long Description

NAME

XpAllocateContextPrivate - Allocate an amount of memory with each context for use by the caller.

SYNOPSIS

```
int XpAllocateContextPrivate(int index, int amount);
```

ARGUMENTS

index	Specifies an index returned by XpAllocateContextPrivateIndex.
amount	The amount of memory to be allocated with each context for use by the caller.

RETURN VALUE

`Success` if no error is detected, otherwise a value indicating the error (e.g. **`BadAlloc`**).

DESCRIPTION

The XpAllocateContextPrivate function informs the printer-independent portion of the server how much memory to allocate with each context for the use of the caller.

Symbols

A

C

D

E

F

G

I

J

L

M

N

P

S

X