Developing v4 Print Drivers v1.2

May 31, 2012

Abstract

This paper provides information about the version 4 print driver model for Windows operating systems. It provides guidelines for hardware manufacturers to build v4 print drivers. It assumes that the reader is familiar with the existing v3 print driver architecture.

This information applies to the following operating systems:
   - Windows 8 Release Preview
   - Windows Server 2012

The current version of this paper is maintained on the web at:
   - Developing v4 Print Drivers

Disclaimer. This document is provided “as-is”. Information and views expressed in this document, including URL and other Internet website references, may change without notice. You bear the risk of using it.

Some examples depicted herein are provided for illustration only and are fictitious. No real association or connection is intended or should be inferred.

This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes. You may modify this document for your internal, reference purposes.

© 2012 Microsoft. All rights reserved.
### Document History

<table>
<thead>
<tr>
<th>Date</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>May 31, 2012</td>
<td>Updated instructions on how to use Visual Studio to build and test drivers; Updated information on Standard XPS Filter features; Added information about the new API used by the XPS; Rasterization Service that enables high precision color support; Added information about disabling punctuation character substitutions; Added information about exceptions that could be encountered using property bags; Added information about how to extend PrinterExtensionLibrary project; Added information about how to share binaries between Metro style and desktop apps; Added information about support for USB Bidi over a secondary USB interface; Added guidance about how to determine if two drivers should share a PrinterDriverID; Added best practices about using GUIDs; Extended existing information about USB Bidi JavaScript; Extended existing information about printer sharing</td>
</tr>
<tr>
<td>February 28, 2012</td>
<td>Updated for Windows 8 Consumer Preview</td>
</tr>
<tr>
<td>September 13, 2011</td>
<td>First publication</td>
</tr>
</tbody>
</table>

### Contents

1. Change Log.......................................................................................................................... 4
2. Introduction............................................................................................................................ 5
   2.1 V4 Design Motivations........................................................................................................ 5
       2.1.1 Design for Metro style Scenarios........................................................................... 5
       2.1.2 Design for Windows on ARM................................................................................... 6
       2.1.3 Design for Ease of Driver Development................................................................... 6
       2.1.4 Design for Printer Sharing..................................................................................... 6
   2.2 High Level Architecture................................................................................................. 6
3. Developing v4 Print Drivers.................................................................................................. 7
   3.1 Using Visual Studio........................................................................................................... 8
       3.1.1 Create a v4 Driver Wizard...................................................................................... 8
       3.1.2 Files and Templates Generated................................................................................ 8
       3.1.3 Building and Testing Your Driver........................................................................... 10
   3.2 File Paths.......................................................................................................................... 11
4. Rendering............................................................................................................................... 11
   4.1 XPSDrv Rendering............................................................................................................ 11
       4.1.1 Architecture............................................................................................................ 11
       4.1.2 Print Filter Pipeline Configuration File................................................................. 11
       4.1.3 IHV Rendering Filter.............................................................................................. 11
       4.1.4 IHV Feature Filter................................................................................................... 12
       4.1.5 Standard XPS Filters............................................................................................... 12
1 Change Log

1.1 Version 1.1

- Updated architecture diagram in 2.2
- Removed section 3.1 – Initializing the environment
- Added section 3.1.3 Building and Testing Your driver.
- Added information about how to utilize the standard XPS filters for PCL6 and PS
- Added clarifications about the use of *Include:msxpsinc.ppd in section 5.
- Added required GPD/PPD attributes in 5.1.
- Clarified the expected usage of *PrintSchemaKeywordMap and *MSPrintSchemaKeywordMap in 5.1.1
- Updated section on JavaScript constraints
  - Updated JavaScript constraints: driverProperties parameter has been superceded by the scriptContext parameter.
  - Added information about debugging
  - Added comments about limitations of JavaScript constraints.
- Added information about the queue property bag.
- Added information about the expected localization behavior.
  - Customized User InterfacesUpdated the printer extension description to reference the PrinterExtensionSample and use the up-to-date versions of the APIs
  - Fixed the code snippet on 6.3.1.2 to remove duplicated "manager.OnDriverEvent +="
  - Clarified OS support for printer extensions
  - Updated diagram in 6
  - Removed all references to “device companion app” or “DCA”, except where they appear in descriptions of samples which are yet to be updated.
  - Removed the Prerequisites section which described workarounds to some fixed bugs
  - Removed the previous content on registering printer extensions and replaced it with the new registry format that does not require a system to have a v4 print driver installed for registration to succeed.
• Added information about how to perform printer maintenance tasks from a printer extension.
• Added information about how to enumerate print queues in printer extensions

• Setup
  • Added information about how to specify a USB soft reset on job cancellation
• Added references to the location of all relevant samples and XSD files.
• Added information about standard compatible IDs supported by Microsoft print class drivers.
• Reorganized Connectivity section for improved clarity.

1.2 Version 1.2
• Updated instructions on how to use Visual Studio to build and test drivers.
• Updated information on Standard XPS Filter features
• Added information about the new API used by the XPS Rasterization Service that enables high precision color support.
• Added information about disabling punctuation character substitutions.
• Added information about exceptions that could be encountered using property bags.
• Added information about how to extend PrinterExtensionLibrary project
• Added information about how to share binaries between Metro style and desktop apps.
• Added information about support for USB Bidi over a secondary USB interface
• Added guidance about how to determine if two drivers should share a PrinterDriverID
• Added best practices about using GUIDs
• Extended existing information about USB Bidi JavaScript
• Extended existing information about printer sharing
2 Introduction

The past 10 years have seen the evolution of the version 3 printer driver model from generic printer description (GPD)/PostScript printer description (PPD)-only configuration and rendering to XPSDrv, auto-config, and full UI replacement. These changes supported the demands of customers and partners, but also added complexity and management overhead. Printer drivers are a top cause of crashes and hangs while using Microsoft Office. Network administrators spend a lot of time managing print driver updates and consolidating print queues, and end users struggle with cross-architecture drivers. In order to help mitigate these issues, many partners have tried universal or class drivers, but the version 3 printer driver model falls short in a number of key scenarios that would enable partner success.

The v4 printer driver model aims to improve the quality of device experience for all users by mitigating known issues in the v3 driver model and developing easy-to-implement extensibility points. While v4 printer drivers continue to use GPDs, PPDs, Bidi, and Generic Descriptor Language (GDL)-based autoconfiguration, some of the layers available in the past have been removed or replaced.

One new aspect of the v4 print driver model is a focus on providing print class drivers. These special v4 print drivers, which implement only functionality that is common across a broad class of devices, are shipped with Microsoft Windows, and are marked as generic printer drivers. As a result, Windows can automatically replace the driver with any better result on Windows Update, providing the end user with a better experience as soon as it is available.

Note that many samples in this paper use the fictional company, Fabrikam, as an example.

2.1 V4 Design Motivations

2.1.1 Design for Metro style Scenarios
Metro style UI and the new Metro app models introduce new design considerations regarding UI behavior and security context. V4 print drivers need to be first class citizens in the Metro UI with deep integration, seamless behavior, and rich extensibility.

2.1.2 Design for Windows on ARM
Running Windows on an ARM processor changes the paradigm with regard to power utilization and driver behaviors. The v4 print driver model supports printers on ARM with print class drivers while still allowing rich end user experiences thanks to a decoupled UI development model.

2.1.3 Design for Ease of Driver Development
The v4 driver model supports existing investments in v3 and the XPSDrv architectures, while making drivers easier to develop and test. The v4 driver model supports driver development using Visual Studio, and supports a number of templates that make drivers easier to build. V4 print drivers also include support for building customized UI as a simple Windows application, enabling developers to use the most up-to-date frameworks and toolsets to keep their experiences fresh.

2.1.4 Design for Printer Sharing
Printer sharing is a major value proposition of Windows print servers, and v4 print drivers are designed to make printer sharing even better. In particular, they reduce management costs, eliminate cross-
architecture driver management, and support all client operating systems from Windows XP to Windows 8.

2.2 High-Level Architecture

The following is a high-level representation of a v4 print driver. With the exception of the rendering filters and user interface applications, all code is Microsoft provided. V4 print drivers rely heavily on data files and JavaScript for extensibility. The blue boxes represent existing files that were used in the v3 driver model, and the green boxes represent new places to plug in.

![Architecture of a v4 Print Driver](image)

**Figure 1. Architecture of a v4 Print Driver**

3 Developing v4 Print Drivers

Because v4 print drivers reuse many technologies that were available in v3 print drivers, we recommend that partners first build out an inventory of existing assets and determine their compatibility with v4 print drivers by using the following table.
Table 1. Overview of Compatible Technologies in v4 Print Drivers

<table>
<thead>
<tr>
<th></th>
<th>V4-compatible</th>
<th>Not V4-compatible</th>
<th>Print class driver only</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rendering</td>
<td>XPSDrv</td>
<td>Monolithic GDI rendering</td>
<td>XPSDrv</td>
</tr>
<tr>
<td></td>
<td>• Includes XPS to PCL6, XPS to PostScript, provided by Microsoft Print class driver dependencies</td>
<td>• Includes XPS to PCL6, XPS to PostScript, provided by Microsoft Unidrv or PScript rendering</td>
<td>Unidrv/PScript rendering</td>
</tr>
<tr>
<td>Configuration</td>
<td>GPD-only configuration</td>
<td>Monolithic configuration GPD configuration with UnidrvUI plug-ins</td>
<td>GPD-only configuration</td>
</tr>
<tr>
<td></td>
<td>PPD-only configuration</td>
<td>PPD configuration with UnidrvUI plug-ins</td>
<td>String Resource DLL</td>
</tr>
<tr>
<td></td>
<td>String Resource DLL</td>
<td>String Resource DLL</td>
<td>String Resource DLL</td>
</tr>
<tr>
<td>Setup</td>
<td>v4 INFs + v4 manifest files</td>
<td>• v3 INFs: Co-installers</td>
<td>v4 INFs + v4 manifest files</td>
</tr>
<tr>
<td></td>
<td>Setup applications using INF based driver install</td>
<td>• Co-installers Setup applications not using INF based install (eg using spooler APIs)</td>
<td></td>
</tr>
<tr>
<td>Connectivity</td>
<td>Bidi extension files</td>
<td>Custom port monitors</td>
<td>Bidi extension files</td>
</tr>
<tr>
<td></td>
<td>Autoconfiguration GDL</td>
<td>Custom language monitors</td>
<td>Autoconfiguration GDL</td>
</tr>
</tbody>
</table>

3.1 Using Visual Studio

The Windows Driver Kit (WDK) now supports Visual Studio as its build environment and IDE. For developers creating v4 print drivers, we have included a new project wizard that builds an appropriate Visual Studio solution.

3.1.1 Create a v4 Driver Wizard

To create a new v4 print driver in Visual Studio, follow these steps:

1. Open the **File** menu and select **New Project**
2. In the **New Project** window, under the Installed templates, choose **Templates, Visual C++, Windows Driver**, and finally **Printer Driver V4**.

The wizard then asks a few basic questions to start the driver. Complete the wizard and then click **Finish** to generate a Visual Studio solution with the appropriate projects and files, as illustrated here.

### 3.1.2 Files and Templates Generated

#### 3.1.2.1 INF

The INF generated for the project requires manual editing prior to being a compliant, installable v4 INF. See the Version 4 INF section of this paper for more details.

#### 3.1.2.2 V4 Driver Manifest

This provides a manifest file based on the selections made in the Create a v4 Driver Wizard, but may also be manually edited using the guidance in this paper under the Version 4 Manifest section of this whitepaper.

#### 3.1.2.3 Render Filter

The Render filter project template provides a good foundation for building an XPS rendering filter and an XPS filter pipeline configuration file. It is based on the existing XPS Rasterization Service sample in the WDK. The following UML diagram provides an overview of the methods and fields implemented in this template.
3.1.2.4 Driver Property Bag

The driver property bag template creates an XML file that describes the contents of a driver property bag. When compiled, the properties specified here, as well as any data files added to the project’s ByteArray or IStream folders, will be compiled into a driver property bag.

The XML Schema for the driver property bag template is in the Windows Driver Kit at \Include\um\printdriverproperties.xml.

3.1.2.5 V4 Print Driver Configuration (GPD)

This template provides a basic GPD file for a v4 print driver. For more information on GPD syntax, see http://msdn.microsoft.com/en-us/library/ff551750(VS.85).aspx or the sample GPD files included in Windows Sample Gallery.
3.1.2.8 V4 Print Driver Configuration (PPD)
This template provides a basic PPD file for a v4 print driver. For more information on PPD syntax, see http://msdn.microsoft.com/en-us/library/ff561810(VS.85).aspx or the sample PPD files included in the Windows Sample Gallery.

PPD based drivers must not specify the *Include: msxpsinc.ppd directive as this is known to cause compatibility issues with some applications.

3.1.2.7 JavaScript Constraints File
This template provides the method headers for all of the supported JavaScript constraint entry points. More information on the JavaScript constraints syntax is available in the JavaScript Constraints section of this whitepaper.

3.1.2.8 Autoconfiguration File (GDL)
This provides a basic autoconfiguration file for a v4 print driver. More information on GDL syntax for autoconfiguration is available in the autoconfig sample on the Windows Sample Gallery and http://msdn.microsoft.com/en-us/library/ff544978(VS.85).aspx.

3.1.2.9 Bidi Extension File (TCP/IP)

3.1.2.10 Bidi Extension File (WSD)

3.1.2.11 Bidi Extension File (USB)
This provides a simple USB Bidi extension file. It is dependent on the existence of a matched USB Bidi Extender JavaScript. More information on this file is available in this whitepaper.

3.1.2.12 Bidi Extension JavaScript File
This provides a simple Bidi Extension JavaScript file to support USB devices. It is dependent on the existence of a matched USB Bidi Extension XML File. More information on this file is available in this whitepaper.

3.1.3 Building and Testing Your Driver
Once you have created your v4 print driver project using the Create a v4 Print Driver Wizard, there are several steps to get the driver built and installable. Please note that the file names and project names used below are following the default names. Your files SHOULD have different names that follow the best practices laid out in this document.

3.1.3.13 Create a v4 Print Driver INF
The INF that is created by the wizard code is incomplete. To make it installable, you must add several items. Configure the [Version] section.

   a. In the [Version] section, set **DriverVer** with format that is similar to this, but using a current date:

      DriverVer=04/01/2012,1.0.0.0

May 14, 2012
© 2012 Microsoft. All rights reserved.
b. Ensure that **ClassVer** is set to 4.0

c. Ensure that **Signature** is set to “$WINDOWS NT$”

2. Configure the v4 print driver’s files

   a. Add files to the [SourceDisksFiles] section as shown below. You may have more files to reference, depending on your driver. By default, all files will be output to the directory referenced by 1 from the [SourceDisksNames] section.

      v4PrinterDriver1.gpd=1

      v4PrinterDriver1-manifest.ini=1

   b. Create a new section called [DriverInstall] at the bottom of the INF

   c. Inside the [DriverInstall] section, create the following entry:

      CopyFiles=DriverFiles

   d. Create a new section called [DriverFiles] at the bottom of the INF

   e. Inside the [DriverFiles] section, list the files as shown below. You may have more file to reference, depending on your driver.

      v4PrinterDriver1.gpd

      v4PrinterDriver1-manifest.ini

3. Configure the driver to target a device

   a. In the [Standard.NT$ARCH$] section, create driver entries in the following format:

      “Model name” = DriverInstall, USBPRINT\Fabrikam1234

      “Model name” = DriverInstall, WSDPRINT\Fabrikam1234

   b. Find the **PrinterDriverID** and add it into the INF.

      i. In **Solution Explorer**, open **v4PrinterDriver-manifest.ini**.

      ii. Find the line starting with **PrinterDriverID**. Select everything to the right of the = sign, right click and choose **Copy**.

      iii. In **Solution Explorer**, open **v4PrinterDriver1.inf**

      iv. In the [Standard.NT$ARCH$] section, add one more driver entry, pasting the value from the manifest file where {GUID} is shown below.

         “Model name”=DriverInstall, {GUID}

4. Provide a **Manufacturer** name

   a. In the [String] section, set the following string value, where “My Company” is your company’s name.

      ManufacturerName = “My Company”
Configure the Visual Studio Solution
Visual Studio supports automatic signing and deployment of drivers. The following steps outline the project and solution configuration necessary to make this happen automatically on each build.

1. Configure remote debugging
   a. Under Solution Explorer, right click on the driver package project v4 Printer Driver 1 Package. Choose Properties.
   b. In the left pane, choose Configuration Properties, then Debugging
   c. Click the Debugger to launch combo box and choose Debugging Tools For Windows – Remote Debugger
   d. Under Remote Computer Name, enter either your local computer’s name to do local installation or a remote test machine’s name.
   e. Click Apply

2. Configure driver installation logic
In the left pane, choose Configuration Properties, then Driver Install, then Deployment
Check Enable deployment, Import into driver store
Under remote computer name, click ...
In the Computer Configuration dialog, choose Add new computer.
Type your target computer’s name, and then click Next.
When the installation has completed, click Next.
Click Finish.
Now in the Remote Computer Name combo box, choose the local machine name.
Choose Install and Verify to have the driver installed to the spooler. If you have any print queues at the time you build your driver, they will be updated to use the driver you just created automatically. This will also run several WDK tests against the print queue you created.
In the Optional Arguments field, enter the name of the print driver you just created.
Alternatively, you could choose Custom Command Line and specify a PowerShell script to install and create a print queue.
Click Apply

3. Configure Driver Signing
   a. In the left pane, choose Driver Signing, then General
   b. In the right pane, confirm that Sign Mode is set to Test Sign
   c. Select Test Certificate, then Create Test Certificate.
   d. Select TimeStampServer, then choose Verisign.
   e. Click Ok.
4. Build your driver
   a. In **Solution Explorer**, right click on the Solution ‘v4 Printer Driver1’ and choose **Build Solution**.

5. Create a print queue using either plug-and-play or the Add Printer Wizard.

### 3.2 File Paths

Since all v4 print drivers execute directly from the DriverStore, and enhanced Point and Print doesn't download the entire driver package to client machines, we recommend that driver binaries not try to open any other binary in the driver and instead use a driver property bag to encapsulate any common, proprietary data.

### 4 Rendering

The v4 driver model uses only the XPSDrv architecture in order to support rendering into device PDL. XPS direct devices need not include any filters, but all others must either include filters that render into the device PDL or take a dependency on an existing print class driver using the RequiredClass directive in the v4 manifest file.

#### 4.1 XPSDrv Rendering

The XPSDrv architecture and the XPS Filter Pipeline follow the same architecture as prior Windows OS releases, with a few additional additions.

##### 4.1.1 Architecture

![Diagram of Rendering Architecture Choices for v4 Print Drivers](image)

**Figure 3. Rendering Architecture Choices for v4 Print Drivers**

##### 4.1.2 Print Filter Pipeline Configuration File

The print filter pipeline configuration file is unchanged in format.

**Recommended naming convention**: `vv<PDL>-pipelineconfig.xml`, where `vv` is a placeholder for your manufacturer code. Example `fapcl6-pipelineconfig.xml`. All print filter pipeline configuration files must end with `.pipelineconfig.xml` in order to be compatible with legacy applications that print XPS.
4.1.3 IHV Rendering Filter
This filter completes the rendering from XPS to the device PDL output. It may use the XPS Rasterization Service or a third-party RIP as necessary.

   **Recommended input type:** `IXpsDocumentProvider`. Using the `IXpsDocumentProvider` interface is faster than using stream interfaces because serialization steps are avoided at a number of points through the rendering process.

   **Recommended output type:** `IWriteStream`. After this filter is complete, the device PDL should be output as a stream.

   **Recommended naming convention:** Use `v<vendor>.dll`, where `v` is a placeholder for your manufacturer code. Example: `faps.dll` for a Fabrikam-provided PostScript renderer.

4.1.3.15 XPS Direct Devices
Devices that are capable of consuming XPS as a PDL may be supported without any rendering filters. However, some devices may require PrintTickets that do not work well with the Microsoft standard UI. In these cases, we recommend that manufacturers convert to a device-compatible PrintTicket in an XPS rendering filter. This ensures the best compatibility with the standard UI and with devices.

4.1.4 IHV Feature Filter
IHV Feature Filters enable the processing of features like N-up, watermarking, or page reordering. These are a convenient way to add features to a driver without changing the underlying PDL rendering.

   **Recommended input type:** `IXpsDocumentProvider`

   **Recommended output type:** `IXpsDocumentConsumer`

For manufacturers with multiple IHV Feature Filters, we recommend that these filters are implemented into the same DLL as separate logical filters. This encourages code sharing and can decrease the overall working set during printing.

4.1.5 Standard XPS Filters
Windows supports built-in conversion from XPS to PCL6 and PostScript level 3. These filters are available for both print class drivers and model-specific v4 print drivers. These can be combined with IHV Feature Filters as well as IHV Post-processing filters, as needed in order to ensure compatibility with existing firmware implementations.

   **Note:** These filters are not redistributable, and are not available to v3 print drivers.

4.1.5.16 Manifest
To use either of these filters, the v4 driver manifest file must specify the filters using the `RequiredFiles` directive under the `DriverConfig` section. The names of the filters are below:
### Table 2. Standard XPS Filter Names

<table>
<thead>
<tr>
<th>PCL6</th>
<th>PS</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSxpsPCL6.dll</td>
<td>MSxpsPS.dll</td>
</tr>
</tbody>
</table>

No INF updates are required to utilize one of these filters, and redistribution is not supported.

#### 4.1.5.17 Print Filter Pipeline Configuration

Create a print filter pipeline configuration file as shown below. You may optionally add filters prior to the filter or after it as needed.

##### 4.1.5.17.1 PCL6

```xml
<?xml version="1.0" encoding="utf-8"?>
<Filters>
  <Filter dll="MSxpsPCL6.dll" clsid="{3821E518-33AF-4d17-92B3-28EB410D46B6}" name="Microsoft XPS to PCL6">
    <Input guid="{4d47a67c-66cc-4430-850e-daf466fe5bc4}" comment="IID_IPrintReadStream" />
    <Output guid="{65bb7f1b-371e-4571-8ac7-912f510c1a38}" comment="IID_IPrintWriteStream" />
  </Filter>
</Filters>
```

##### 4.1.5.17.2 PostScript

```xml
<?xml version="1.0" encoding="utf-8"?>
<Filters>
  <Filter dll="MSxpsPS.dll" clsid="{8636D90A-5E03-4d62-9269-E06493C57473}" name="Microsoft XPS to PS">
    <Input guid="{4d47a67c-66cc-4430-850e-daf466fe5bc4}" comment="IID_IPrintReadStream" />
    <Output guid="{65bb7f1b-371e-4571-8ac7-912f510c1a38}" comment="IID_IPrintWriteStream" />
  </Filter>
</Filters>
```

#### 4.1.5.18 Features Supported

The standard XPS filters support many common features. All feature definitions use the GPD or PPD file for the driver. The MSxpsPCL6.dll filter requires that a GPD file is used for configuration and the
MSxpsPS.dll filter requires that a PPD file is used for configuration. Unless otherwise noted, if a custom PDL command is specified for a feature, it will be used.

Specific to GPD, if injection strings exist under a particular section (specified with the *Order command) the filter will make a number of assumptions about the content of those strings and will avoid sending default commands which may cause collisions. Thus, the creator of a GPD must follow these guidelines:

- JOB_SETUP.#
  - A PCL6 Binary Stream Header (for example: “”<HP-PCL XL;1;<CR><LF>”) must exist.
  - A BeginSession operator must exist, including all required attributes.
  - An OpenDataSource operator must exist, including all required attributes.
- PAGE_SETUP.#
  - A BeginPage operator must exist, including all required attributes.
- PAGE_FINISH.#
  - An EndPage operator must exist.
- JOB_FINISH.#
  - A CloseDataSource operator must exist.
  - An EndSession operator must exist
  - An EndPJLCommands operator must exist.

GPD and PPD files should not specify a *PageOrigin other than (0,0), nor should they specify CMD strings that add further offsets to page content. The filters will generate PDL that places the page within the printable area as specified by the following directives, on a per media size basis.

<table>
<thead>
<tr>
<th>Printable Area</th>
<th>Printable Origin</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPD</td>
<td>*PrintableArea</td>
</tr>
<tr>
<td></td>
<td>*PrintableOrigin</td>
</tr>
<tr>
<td>PPD</td>
<td>*ImageableArea</td>
</tr>
<tr>
<td></td>
<td>N/A Included in the existing</td>
</tr>
<tr>
<td></td>
<td>*ImageableArea directive</td>
</tr>
</tbody>
</table>

The following PrintTicket features are supported by the filters. All elements (Features, Options, ScoredProperties, Parameters) listed are from the print schema keywords (psk) namespace. All features will cause the filters to vary the PDL generated whether the PDL commands are generated by the filter itself or specified by the device GPD/PPD.

4.1.5.18.1 PageMediaSize
This Feature describes the dimensions of the media sheet used for the printed output. In addition to the name, each Option can contain two Scored Properties: MediaSizeWidth and MediaSizeHeight. These describe the physical size of the media. Supported Options are any with a corresponding GPD/PPD entry.

For PCL6/GPD, if the PrintTicket Option is CustomMediaSize then the Parameters PageMediaSizeMediaSizeWith and PageMediaSizeMediaSizeHeight are used to obtain the dimensions of the media.

For PostScript/PPD, if the PrintTicket Option is PSCustomMediaSize then the Parameters PageMediaSizePSWith and PageMediaSizePSHeight are used to obtain the dimensions of the media.
The PCL6 generated for the selected media type is specified by the GPD PageSize Feature. The Option in the GPD to use is selected in the following order:

1. If PrintSchemaKeywordMap is specified and matches the name attribute of the PageMediaSize Option.

2. The default mappings are used:
   
   - PrintTicket CustomMediaSize -> GPD CUSTOMSIZE
   - PrintTicket NorthAmerica10x11 -> GPD 10X11
   - PrintTicket NorthAmerica10x14 -> GPD 10X14
   - PrintTicket NorthAmerica11x17 -> GPD 11X17
   - PrintTicket NorthAmerica9x11 -> GPD 9X11
   - PrintTicket NorthAmericaSuperA -> GPD A_PLUS
   - PrintTicket ISOA2 -> GPD A2
   - PrintTicket ISOA3 -> GPD A3
   - PrintTicket ISOA3Extra -> GPD A3_EXTRA
   - PrintTicket ISOA4 -> GPD A4
   - PrintTicket ISOA4Extra -> GPD A4_EXTRA
   - PrintTicket OtherMetricA4Plus -> GPD A4_PLUS
   - PrintTicket ISOA5 -> GPD A5
   - PrintTicket ISOA5Extra -> GPD A5_EXTRA
   - PrintTicket ISOA6 -> GPD A6
   - PrintTicket NorthAmericaSuperB -> GPD B_PLUS
   - PrintTicket JSB4 -> GPD B4
   - PrintTicket JSB5 -> GPD B5
   - PrintTicket ISOB5Extra -> GPD B5_EXTRA
   - PrintTicket JSB6 -> GPD B6_JIS
   - PrintTicket NorthAmericaCSheet -> GPD CSHEET
- PrintTicket JapanDoubleHagakiPostcard -> GPD DBL_JAPANESE_POSTCARD
- PrintTicket NorthAmericaDSheet -> GPD DSHEET
- PrintTicket NorthAmericaNumber10Envelope -> GPD ENV_10
- PrintTicket NorthAmericaNumber11Envelope -> GPD ENV_11
- PrintTicket NorthAmericaNumber12Envelope -> GPD ENV_12
- PrintTicket NorthAmericaNumber14Envelope -> GPD ENV_14
- PrintTicket NorthAmericaNumber9Envelope -> GPD ENV_9
- PrintTicket ISOB4Envelope -> GPD ENV_B4
- PrintTicket ISOB5Envelope -> GPD ENV_B5
- PrintTicket ISOC3Envelope -> GPD ENV_C3
- PrintTicket ISOC4Envelope -> GPD ENV_C4
- PrintTicket ISOC5Envelope -> GPD ENV_C5
- PrintTicket ISOC6Envelope -> GPD ENV_C6
- PrintTicket ISOC65Envelope -> GPD ENV_C65
- PrintTicket ISODLEEnvelope -> GPD ENV_DL
- PrintTicket OtherMetricInviteEnvelope -> GPD ENV_INVITE
- PrintTicket OtherMetricItalianEnvelope -> GPD ENV_ITALY
- PrintTicket NorthAmericaMonarchEnvelope -> GPD ENV_MONARCH
- PrintTicket NorthAmericaPersonalEnvelope -> GPD ENV_PERSONAL
- PrintTicket NorthAmericaESheet -> GPD ESHEET
- PrintTicket NorthAmericaExecutive -> GPD EXECUTIVE
- PrintTicket NorthAmericaGermanLegalFanfold -> GPD FANFOLD_LGL_GERMAN
- PrintTicket NorthAmericaGermanStandardFanfold -> GPD FANFOLD_STD_GERMAN
- PrintTicket OtherMetricFolio -> GPD FOLIO
- PrintTicket ISOB4 -> GPD ISO_B4
- PrintTicket JapanHagakiPostcard -> GPD JAPANESE_POSTCARD
- PrintTicket JapanChou3Envelope -> GPD JENV_CHOU3
- PrintTicket JapanChou4Envelope -> GPD JENV_CHOU4
- PrintTicket JapanKaku2Envelope -> GPD JENV_KAKU2
- PrintTicket JapanKaku3Envelope -> GPD JENV_KAKU3
- PrintTicket JapanYou4Envelope -> GPD JENV_YOU4
- PrintTicket NorthAmericaLegal -> GPD LEGAL
- PrintTicket NorthAmericaLegalExtra -> GPD LEGAL_EXTRA
- PrintTicket NorthAmericaLetter -> GPD LETTER
- PrintTicket NorthAmericaLetterExtra -> GPD LETTER_EXTRA
- PrintTicket NorthAmericaLetterPlus -> GPD LETTER_PLUS
- PrintTicket NorthAmericaNote -> GPD NOTE
- PrintTicket PRC16K -> GPD P16K
- PrintTicket PRC32K -> GPD P32K
- PrintTicket PRC32KBig -> GPD P32KBIG
- PrintTicket PRC1Envelope -> GPD PENV_1
- PrintTicket PRC10Envelope -> GPD PENV_10
- PrintTicket PRC2Envelope -> GPD PENV_2
- PrintTicket PRC3Envelope -> GPD PENV_3
- PrintTicket PRC4Envelope -> GPD PENV_4
- PrintTicket PRC5Envelope -> GPD PENV_5
- PrintTicket PRC6Envelope -> GPD PENV_6
- PrintTicket PRC7Envelope -> GPD PENV_7
- PrintTicket PRC8Envelope -> GPD PENV_8
- PrintTicket PRC9Envelope -> GPD PENV_9
- PrintTicket NorthAmericaQuarto -> GPD QUARTO
- PrintTicket NorthAmericaStatement -> GPD STATEMENT
- PrintTicket NorthAmericaTabloid -> GPD TABLOID
- PrintTicket NorthAmericaTabloidExtra -> GPD TABLOID_EXTRA
- PrintTicket CustomMediaSize -> GPD CUSTOMSIZE

3. The name attribute of the PageSize Option matches the name of the Option in the GPD.

The filter will replace PhysPaperWidth in any GPD command with the width of the paper specified by the MediaSizeWidth ScoredProperty or the PageMediaSizeMediaSizeWidth Parameter.

The filter will replace PhysPaperLength in any GPD command with the length of the paper specified by the MediaSizeHeight ScoredProperty or the PageMediaSizeMediaSizeHeight Parameter.

The PostScript generated for the selected media type is specified by the PPD PageSize Feature. The Option in the PPD to use is selected in the following order:

1. If MSPrintSchemaKeywordMap is specified and matches the name attribute of the PageMediaSize Option.

2. The name attribute of the PageSize Option matches the name of the Option in the PPD

4.1.5.18.2 PageMediaType
This Feature describes characteristics of the media sheet available to the device, such as coatings, media material, and media weight. Supported Options are any with a corresponding GPD/PPD entry.

The PCL6 generated for the selected media type is specified by the GPD MediaType Feature. The Option in the GPD to use is selected in the following order:

1. If PrintSchemaKeywordMap is specified and matches the name attribute of the PageMediaType Option.

2. The default mappings are used:

   - PrintTicket PhotographicGlossy -> GPD GLOSSY
   - PrintTicket Plain -> GPD STANDARD
   - PrintTicket Transparency -> GPD TRANSPARENCY

3. The name attribute of the PageMediaType Option matches the name of the Option in the GPD.

The PostScript generated for the selected media type is specified by the PPD MediaType Feature. The Option in the PPD to use is selected in the following order:
1. If MSPrintSchemaKeywordMap is specified and matches the name attribute of the PageMediaType Option.

2. The name attribute of the PageMediaType Option matches the name of the Option in the PPD.

4.1.5.18.3 PageMediaColor

This Feature describes the color of the media sheet. Supported Options are any with a corresponding GPD/PPD entry.

The PCL6 generated for the selected media color is specified by the GPD Feature containing *PrintSchemaKeywordMap: “PageMediaColor”. The Option in the GPD to use is selected in the following order:

1. If PrintSchemaKeywordMap is specified and matches the name attribute of the PageMediaColor Option.

2. The name attribute of the PageMediaColor Option matches the name of the Option in the PPD.

The PostScript generated for the selected media color is specified by the PPD MediaColor feature. The Option in the PPD to use is selected in the following order:

1. If MSPrintSchemaKeywordMap is specified and matches the name attribute of the PageMediaColor Option.

2. The name attribute of the PageMediaColor Option matches the name of the Option in the PPD.

4.1.5.18.4 JobInputBin

This Feature describes where media is to be taken into the device. Supported Options are any with a corresponding GPD/PPD entry.

The PCL6 generated for the selected input tray is specified by the GPD InputBin feature. The Option in the GPD to use is selected in the following order:

1. If PrintSchemaKeywordMap is specified and matches the name attribute of the JobInputBin Option.

2. The default mappings are used:

   - PrintTicket Cassette -> GPD AUTO,CASSETTE,ENVFEED,ENVMANUAL
   - PrintTicket AutoSelect -> GPD FORMSOURCE
   - PrintTicket High -> GPD LARGECAPACITY,LARGEFMT,LOWER
• PrintTicket Manual -> GPD MANUAL,MIDDLE,SMALLFMT

• PrintTicket Tractor -> GPD TRACTOR,UPPER

3. The name attribute of the PageMediaTypeInfo Option matches the name of the Option in the GPD.

The PostScript generated for the selected input tray is specified by the PPD InputSlot feature. The Option in the PPD to use is selected in the following order:

1. If MSPrintSchemaKeywordMap is specified and matches the name attribute of the JobInputBin Option.

2. The name attribute of the JobInputBin Option matches the name of the Option in the PPD.

4.1.5.18.5 PageOrientation

This Feature indicates the rotation transform to use when converting from the content coordinate space to the media coordinate space for the sheet. Supported Options are Portrait, Landscape, ReversePortrait, and ReverseLandscape.

The PCL6 generated for the selected orientation is specified by the GPD Orientation Feature. The Option in the GPD to use is selected in the following order:

1. If PrintSchemaKeywordMap is specified and matches the name attribute of the PageOrientation Option.

2. The default mappings are used:

   a. PrintTicket Portrait -> GPD PORTRAIT

   b. PrintTicket Landscape -> GPD LANDSCAPE_CC90

   c. PrintTicket ReverseLandscape -> GPD LANDSCAPE_CC270

3. The name attribute of the PageOrientation Option matches the name of the Option in the GPD.

The PostScript generated for the selected orientation is determined by the filter.

4.1.5.18.6 PageOutputColor

This Feature controls the color characteristics (color, monochrome) of the printed output for the destination document page. Supported Options are Color, Grayscale, and Monochrome.

The PCL6 generated for the selected output color is specified by the GPD ColorMode Feature. The Option in the GPD to use is selected in the following order:
1. If PrintSchemaKeywordMap is specified and matches the name attribute of the PageOutputColor Option.

2. The name attribute of the PageOutputColor Option matches the name of the Option in the GPD.

The PostScript generated for the selected output color is determined by the filter.

4.1.5.18.7 PageResolution

This Feature defines the available resolutions (in dots per inch) at which the device can produce output. The Print Schema does not specify any standard names for Options of this Feature; however, we do support two ScoredProperties regardless of the Option name: ResolutionX, and ResolutionY. Supported Options are any with a corresponding GPD/PPD entry.

The PCL6 generated for the selected resolution is specified by the GPD Resolution Feature. The Option in the GPD to use is selected in the following order:

1. If PrintSchemaKeywordMap is specified and matches the name attribute of the PageResolution Option.

2. The name attribute of the PageResolution Option matches the name of the Option in the GPD.

The filter will replace GraphicsXRes and TextXRes in and GPD command with the horizontal resolution specified by ResolutionX.

The filter will replace GraphicsYRes and TextYRes in any GPD command with the vertical resolution specified by ResolutionY.

The PostScript generated for the selected resolution is specified by the PPD Resolution or JCLResolution Feature. The Option in the PPD to use is selected in the following order:

1. If MSPrintSchemaKeywordMap is specified and matches the name attribute of the PageResolution Option.

2. The name attribute of the PageResolution Option matches the name of the Option in the PPD.

4.1.5.18.8 PageOutputQuality

This Feature defines the print quality for document page. Supported Options are any with a corresponding GPD/PPD entry.

The PCL6 generated for the selected quality is specified by the GPD Feature with a PrintSchemaKeywordMap value of PageOutputQuality. The Option in the GPD to use is selected in the following order:
1. If PrintSchemaKeywordMap is specified and matches the name attribute of the PageOutputQuality Option.

2. The name attribute of the PageOutputQuality Option matches the name of the Option in the GPD.

The PostScript generated for the selected quality is specified by the PPD Feature with an MSPrintSchemaKeywordMap value of PageOutputQuality. The Option in the PPD to use is selected in the following order:

1. If MSPrintSchemaKeywordMap is specified and matches the name attribute of the PageOutputQuality Option.

2. The name attribute of the PageOutputQuality Option matches the name of the Option in the PPD.

4.1.5.18.9 JobCopiesAllDocuments

This Parameter specifies the number of times that all documents within the print job are output.

The PCL6 generated for the selected copies is determined by the filter. See the JobCollateAllDocuments Feature for interaction with this Parameter.

The PostScript generated for the selected copies is determined by the filter. See the JobCollateAllDocuments Feature for interaction with this Parameter.

4.1.5.18.10DocumentCopiesAllPages

This Parameter specifies the number of page copies the associated document in the print job should output.

The PCL6 generated for the selected copies is determined by the filter. See the DocumentCollate Feature for interaction with this Parameter.

The PostScript generated for the selected copies is determined by the filter. See the DocumentCollate Feature for interaction with this Parameter.

4.1.5.18.11PageCopies

This Parameter specifies how many copies of an individual source document page within a document should be output. Since the copy count only applies to the current page there is no collation.

The PCL6 generated for the selected copies is determined by the filter.

The PostScript generated for the selected copies is determined by the filter.
4.1.5.18.12 DocumentCollate

This Feature specifies the order in which the pages of the associated document in the print job appear in the printed output. Supported Options are Collated and Uncollated.

The PCL6 generated for the selected collation is specified by the GPD Collate Feature. The Option in the GPD to use is selected in the following order:

1. If PrintSchemaKeywordMap is specified and matches the name attribute of the DocumentCollate Option.
2. The default mappings are used:
   a. PrintTicket Uncollated -> GPD OFF
   b. PrintTicket Collated -> GPD ON
3. The name attribute of the DocumentCollate Option matches the name of the Option in the GPD.

NOTE: When DocumentCollate is set to Collated and the GPD Collate option contains a command then it is assumed that the device can generate the collated copies. The XPS.PCL6 filter will only generate 1 copy of the job and use the GPD command to instruct the device to generate the collated copies. The filter will replace NumOfCopies in the GPD command with the number of copies specified by JobCopiesAllDocuments.

The PostScript generated for the selected collation is specified by the PPD Collate feature. The Option in the PPD to use is selected in the following order:

1. If MSPrintSchemaKeywordMap is specified and matches the name attribute of the DocumentCollate Option.
2. The default mappings are used:
   • Print Ticket Uncollated -> PPD False
   • Print Ticket Collated -> PPD True
3. The name attribute of the DocumentCollate Option matches the name of the Option in the PPD.

NOTE: When DocumentCollate is set to Collated and the PPD contains the Collate Feature, or a Feature which is keyword mapped to DocumentCollate, then it is assumed that the device can generate the collated copies. The XPS.PS filter will only generate 1 copy of the job and use the PPD command to instruct the device to generate the collated copies.
4.1.5.18.13 JobDuplexAllDocumentsContiguously

This Feature specifies the duplex printing of the print job without consideration for document boundaries. If duplex printing is specified, all pages of all documents in the print job are duplex printed continuously without the insertion of blank pages between documents. Supported Options are OneSided, TwoSidedShortEdge, and TwoSidedLongEdge.

The PCL6 generated for the selected duplex is specified by the GPD Duplex Feature. The Option in the GPD to use is selected in the following order:

1. If PrintSchemaKeywordMap is specified and matches the name attribute of the JobDuplexAllDocumentsContiguously Option.

2. The default mappings are used:
   - PrintTicket OneSided -> GPD NONE
   - PrintTicket TwoSidedShortEdge -> GPD HORIZONTAL
   - PrintTicket TwoSidedLongEdge -> GPD VERTICAL

3. The name attribute of the JobDuplexAllDocumentsContiguously Option matches the name of the Option in the GPD.

The PostScript generated for the selected duplex is specified by the PPD Duplex Feature. The Option in the PPD to use is selected in the following order:

1. If MSPrintSchemaKeywordMap is specified and matches the name attribute of the JobDuplexAllDocumentsContiguously Option.

2. The default mappings are used:
   - PrintTicket OneSided -> PPD None
   - PrintTicket TwoSidedShortEdge -> PPD DuplexTumble
   - PrintTicket TwoSidedLongEdge -> PPD DuplexNoTumble

3. The name attribute of the JobDuplexAllDocumentsContiguously Option matches the name of the Option in the PPD.

4.1.5.18.14 DocumentDuplex

This Feature controls duplex printing of the associated document in the print job. If this is specified, the printed output begins on the front side of a new sheet of media. Supported Options are OneSided, TwoSidedShortEdge, and TwoSidedLongEdge.
The PCL6 generated for the selected duplex is specified by the GPD Duplex Feature. The Option in the GPD to use is selected in the following order:

1. If PrintSchemaKeywordMap is specified and matches the name attribute of the DocumentDuplex Option.

2. The default mappings are used:
   a. PrintTicket OneSided -> GPD NONE
   b. PrintTicket TwoSidedShortEdge -> GPD HORIZONTAL
   c. PrintTicket TwoSidedLongEdge -> GPD VERTICAL

3. The name attribute of the DocumentDuplex Option matches the name of the Option in the GPD.

The PostScript generated for the selected duplex is specified by the PPD Duplex Feature. The Option in the PPD to use is selected in the following order:

1. If MSPrintSchemaKeywordMap is specified and matches the name attribute of the DocumentDuplex Option.

2. The default mappings are used:
   - PrintTicket OneSided -> PPD None
   - PrintTicket TwoSidedShortEdge -> PPD DuplexTumble
   - PrintTicket TwoSidedLongEdge -> PPD DuplexNoTumble

3. The name attribute of the DocumentDuplex Option matches the name of the Option in the PPD.

4.1.5.18.15 DocumentNUp

This Feature specifies that the content of multiple pages should be output on each sheet of the physical media such that the content from different documents is not output on the same sheet. The Print Schema Specification does not specify a name for this Option; however the Option does support the ScoredProperty PagesPerSheet to specify how many pages will be placed on one side of the physical media. Supported values of PagesPerSheet are 1, 2, 4, 6, 8, 9, 12, 16, 25 and 32 with the physical page orientation being rotated for 2, 6, 8, 12 and 32.

The PCL6 generated for the selected N-Up is determined by the filter.

The PostScript generated for the selected N-Up is determined by the filter.
4.1.5.18.16 JobOutputBin

This Feature describes where media is deposited after it has been printed. Supported Options are any with a corresponding GPD/PPD entry.

The PCL6 generated for the selected output bin is specified by the GPD OutputBin Feature. The Option in the GPD to use is selected in the following order:

1. If PrintSchemaKeywordMap is specified and matches the name attribute of the [Job|Document|Page]OutputBin Option.

2. The name attribute of the [Job|Document|Page]OutputBin Option matches the name of the Option in the GPD.

The PostScript generated for the selected duplex is specified by the PPD OutputBin Feature. The Option in the PPD to use is selected in the following order:

1. If MSPrintSchemaKeywordMap is specified and matches the name attribute of the [Job|Document|Page]OutputBin Option.

2. The name attribute of the [Job|Document|Page]OutputBin Option matches the name of the Option in the PPD.

4.1.5.18.17 JobBindAllDocuments

This Feature describes the method of binding for the printed sheets in the print job. All documents in the print job should be bound together. Supported Options include: None, BindBottom, BindLeft, BindRight, BindTop, Booklet, EdgeStitchBottom, EdgeStitchLeft, EdgeStitchRight, EdgeStitchTop.

When Booklet is selected the filter output will be formatted 2-Up with the pages re-ordered such that when the stack of sheets for the job is folded in half the pages are in the proper order for a book.

When the BindingGutter ScoredProperty is specified for Booklet, the filter will enforce a center margin (from the center of the paper to the edge of the scaled printable area) that is at least as large as specified by the JobBindAllDocumentsGutter Parameter.

When the BindingGutter ScoredProperty is specified for BindLeft, EdgeStitchLeft the filter will shift the front side of the sheet to the right as specified by the JobBindAllDocumentsGutter Parameter. Content on the right side that now falls outside the printable area will be clipped. Content on the back side of the sheet will be clipped on the right edge as specified by the JobBindAllDocumentsGutter Parameter.

When the BindingGutter ScoredProperty is specified for BindTop, EdgeStitchTop the filter will shift the content of both the front and back sides of the sheet toward the bottom as specified by
the JobBindAllDocumentsGutter Parameter. Content on the bottom that now falls outside the printable area will be clipped.

When the BindingGutter ScoredProperty is specified for BindRight, EdgeStitchRight the filter will clip the content on the front side of the sheet on the right as specified by the JobBindAllDocumentsGutter Parameter. Content on the back side of the sheet will be shifted to the left as specified by the JobBindAllDocumentsGutter Parameter. Content on the left side that now falls outside the printable area will be clipped.

When the BindingGutter ScoredProperty is specified for BindBottom, EdgeStitchBottom the filter will shift the content of both the front and back sides of the sheet toward the top as specified by the JobBindAllDocumentsGutter Parameter. Content on the top that now falls outside the printable area will be clipped.

Note that the binding edge is the specified edge based on the orientation of the first page of the first document in the job. For all other Options, BindingGutter is ignored.

If the GPD does not specify a command for the selected option then the PCL6 generated for the selected binding is determined by the filter.

If the PPD does not specify an invocation command for the selected option then the PostScript generated for the selected binding is determined by the filter.

4.1.5.18.18DocumentBinding

This Feature describes the method to use when binding the printed sheets of the associated document in the print job. All pages in the document should be bound together. Supported Options include: None, BindBottom, BindLeft, BindRight, BindTop, Booklet, EdgeStitchBottom, EdgeStitchLeft, EdgeStitchRight, and EdgeStitchTop.

When Booklet is selected the filter output will be formatted 2-Up with the pages re-ordered such that when the stack of sheets for the document is folded in half the pages are in the proper order for a book.

When the BindingGutter ScoredProperty is specified for Booklet, the filter will enforce a center margin (from the center of the paper to the edge of the scaled printable area) that is at least as large as specified by the DocumentBindingGutter Parameter.

When the BindingGutter ScoredProperty is specified for BindLeft, EdgeStitchLeft the filter will shift the front side of the sheet to the right as specified by the DocumentBindingGutter Parameter. Content on the right side that now falls outside the printable area will be clipped. Content on the back side of the sheet will be clipped on the right edge as specified by the DocumentBindingGutter Parameter.

When the BindingGutter ScoredProperty is specified for BindTop, EdgeStitchTop the filter will shift the content of both the front and back sides of the sheet toward the bottom as specified by
the DocumentBindingGutter Parameter. Content on the bottom that now falls outside the printable area will be clipped.

When the BindingGutter ScoredProperty is specified for BindRight, EdgeStitchRight the filter will clip the content on the front side of the sheet on the right as specified by the DocumentBindingGutter Parameter. Content on the back side of the sheet will be shifted to the left as specified by the DocumentBindingGutter Parameter. Content on the left side that now falls outside the printable area will be clipped.

When the BindingGutter ScoredProperty is specified for BindBottom, EdgeStitchBottom the filter will shift the content of both the front and back sides of the sheet toward the top as specified by the DocumentBindingGutter Parameter. Content on the top that now falls outside the printable area will be clipped.

Note that the binding edge is the specified edge based on the orientation of the first page of the document. For all other Options, BindingGutter is ignored.

If the GPD does not specify a command for the selected option then the PCL6 generated for the selected binding is determined by the filter.

If the PPD does not specify an invocation command for the selected option then the PostScript generated for the selected binding is determined by the filter.

4.1.5.18.19 JobStapleAllDocuments

This Feature describes the method to use when stapling the printed sheets in the print job. All documents in the job should be stapled together. Supported Options are any with a corresponding GPD/PPD entry.

The PCL6 generated for the selected stapling is specified by the GPD Staple Feature. The Option in the GPD to use is selected in the following order:

1. If PrintSchemaKeywordMap is specified and matches the name attribute of the JobStapleAllDocuments Option.

2. The name attribute of the JobStapleAllDocuments Option matches the name of the Option in the GPD.

The PostScript generated for the selected stapling is specified by the PPD Feature with an MSPrintSchemaKeywordMap value of JobStapleAllDocuments or DocumentStaple. The Option in the PPD to use is selected in the following order:

1. If MSPrintSchemaKeywordMap is specified and matches the name attribute of the JobStapleAllDocuments Option.

2. The name attribute of the JobStapleAllDocuments Option matches the name of the Option in the PPD.
4.1.5.18.20 JobHolePunch
This Feature describes the method to use when hole punching the printed sheets in the print job. All documents in the job should be hole punched together. Supported Options are any with a corresponding GPD/PPD entry.

The PCL6 generated for the selected hole punching is specified by the GPD Feature with a PrintSchemaKeywordMap value of JobHolePunch or DocumentHolePunch. The Option in the GPD to use is selected in the following order:

1. If PrintSchemaKeywordMap is specified and matches the name attribute of the JobHolePunch Option.
2. The name attribute of the JobHolePunch Option matches the name of the Option in the GPD.

The PostScript generated for the selected hole punching is specified by the PPD Feature with an MSPrintSchemaKeywordMap value of JobHolePunch or DocumentHolePunch. The Option in the PPD to use is selected in the following order:

1. If MSPrintSchemaKeywordMap is specified and matches the name attribute of the JobHolePunch Option.
2. The name attribute of the JobHolePunch Option matches the name of the Option in the PPD.

4.1.5.18.21 DocumentHolePunch
This Feature describes the method to use when hole punching the printed sheets of the associated document in the print job. All pages in the document should be hole punched together. Supported Options are any with a corresponding GPD/PPD entry.

The PCL6 generated for the selected hole punching is specified by the GPD Feature with a PrintSchemaKeywordMap value of JobHolePunch or DocumentHolePunch. The Option in the GPD to use is selected in the following order:

1. If PrintSchemaKeywordMap is specified and matches the name attribute of the DocumentHolePunch Option.
2. The name attribute of the DocumentHolePunch Option matches the name of the Option in the GPD.

The PostScript generated for the selected hole punching is specified by the PPD Feature with an MSPrintSchemaKeywordMap value of JobHolePunch or DocumentHolePunch. The Option in the PPD to use is selected in the following order:

1. If MSPrintSchemaKeywordMap is specified and matches the name attribute of the DocumentHolePunch Option.
2. The name attribute of the DocumentHolePunch Option matches the name of the Option in the PPD.

4.1.5.18.22PageMirrorImage

This Feature specifies if the page content should be mirrored. Supported Options are None and MirrorImageWidth.

The PCL6 generated for the selected mirroring is specified by the GPD Feature with a PrintSchemaKeywordMap value of PageMirrorImage. The Option in the GPD to use is selected in the following order:

1. If PrintSchemaKeywordMap is specified and matches the name attribute of the PageMirrorImage Option.

2. The name attribute of the PageMirrorImage Option matches the name of the Option in the GPD.

The PostScript generated for the selected mirroring is specified by the PPD MirrorPrint Feature. The Option in the PPD to use is selected in the following order:

1. If MSPrintSchemaKeywordMap is specified and matches the name attribute of the PageMirrorImage Option.

2. The default mappings are used:
   - PrintTicket None -> PPD False
   - PrintTicket MirrorImageWidth -> PPD True

3. The name attribute of the PageMirrorImage Option matches the name of the Option in the PPD.

4.1.5.18.23PageNegativeImage

This Feature specifies if the page content should be a negative image. Supported Options are None and Negative.

The PCL6 generated for the selected negative printing is specified by the GPD Feature with a PrintSchemaKeywordMap value of PageNegativeImage. The Option in the GPD to use is selected in the following order:

1. If PrintSchemaKeywordMap is specified and matches the name attribute of the PageNegativeImage Option.

2. The name attribute of the PageNegativeImage Option matches the name of the Option in the GPD.
The PostScript generated for the selected negative printing is specified by the PPD NegativePrint Feature. The Option in the PPD to use is selected in the following order:

1. If MSPrintSchemaKeywordMap is specified and matches the name attribute of the PageNegativeImage Option.
2. The default mappings are used:
   - PrintTicket None -> PPD False
   - PrintTicket Negative -> PPD True
3. The name attribute of the PageNegativeImage Option matches the name of the Option in the PPD.

4.1.5.19 Other Features
The filters will check all PrintTicket members to see if they are referenced in the GPD/PPD and specify commands to be output. If so, the specified commands will be generated.

GPD Features are mapped in the following order:

1. A PrintSchemaKeywordMap value is specified and matches the PrintTicket Feature name.
2. The PrintSchemaPrivateNamespaceURI attribute is specified, and the GPD Feature name matches the PrintTicket Feature name. Matching Feature names is not straightforward, and follows a number of rules:
   a. If the *Order section of the first option is PAGE_SETUP or PAGE_FINISH, and the GPD Feature does not begin with “Page”, “Page” will be prepended to the GPD Feature before attempting to match.
   b. If the *Order section of the first option is DOC_SETUP or DOC_FINISH, and the GPD Feature does not begin with “Document”, “Document” will be prepended to the GPD Feature before attempting to match.
   c. If the *Order section of the first option is JOB_SETUP or JOB_FINISH, and the GPD Feature does not begin with “Job”, “Job” will be prepended to the GPD Feature before attempting to match.
   d. Any character which is not [A-Z] [a-z] [0-9] or ‘_’ will be replaced with a ‘_’ character before attempting to match.
      i. If the *NoPunctuationCharSubstitute? Attribute is TRUE, we will not replace ‘.’ or ‘-‘ with an ‘_’ character.

PPD Features are mapped in the following order:

1. A PrintSchemaKeywordMap value is specified and matches the PrintTicket Feature name.
2. The PrintSchemaPrivateNamespaceURI attribute is specified, and the PPD Feature name matches the PrintTicket Feature name. Matching Feature names is not straightforward, and follows a number of rules:

   a. If the OrderDependency section is ExitServer, Prolog, or JCLSetup, and the PPD Feature does not begin with “Job”, “Job” will be prepended to the PPD Feature before attempting to match.

   b. If the OrderDependency section is DocumentSetup, and the PPD Feature does not begin with “Document”, “Document” will be prepended to the PPD Feature before attempting to match.

   c. If the OrderDependency section is AnySetup, we do two match checks:

      i. If the PPD Feature does not begin with “Document”, “Document” will be prepended to the PPD Feature before attempting to match.

      ii. If no match found, if the PPD Feature does not begin with “Job”, “Job” will be prepended to the PPD Feature before attempting to match.

   d. If the OrderDependency section is PageSetup, and the PPD Feature does not begin with “Page”, “Page” will be prepended to the PPD Feature before attempting to match.

   e. Any character which is not [A-Z] [a-z] [0-9] or ‘_’ will be replaced with a ‘_’ character before attempting to match.

      i. If the *MSNoPunctuationCharSubstitute? String is TRUE, we will not replace ‘.’ or ‘-’ with an ‘_’ character.

GPD and PPD Options are mapped in the following order:

1. A PrintSchemaKeywordMap value is specified and matches the PrintTicket Option name.

2. The PrintSchemaPrivateNamespaceURI attribute is specified, and the GPD/PPD Option name matches the PrintTicket Option name. Matching Option names is not straightforward, and follows a number of rules:

4. If the GPD/PPD Option name starts with [0-9] or ‘_’, then a ‘_’ character will be prepended to the GPD/PPD Option before attempting to match.

    i. If GPD, and the *NoPunctuationCharSubstitute? Attribute is TRUE, we will not prepend ‘_’ with a ‘_’ character.

    ii. If PPD, and the *MSNoPunctuationCharSubstitute? String is True, we will not prepend ‘_’ with a ‘_’ character.
b. Any character which is **not** [A-Z] [a-z] [0-9] or `'` will be replaced with a `'` character before attempting to match.
   i. If GPD, and the *NoPunctuationCharSubstitute? Attribute is TRUE, we will **not** replace `'` or `.` with an `'` character.
   ii. If PPD, and the *MSNoPunctuationCharSubstitute? String is True, we will **not** replace `'` or `.` with an `'` character.

4.1.6 XPS Format
The XPS Print API and/or the print filter pipeline will convert seamlessly between [Microsoft XML Paper Specification 1.0](https://msdn.microsoft.com/en-us/library/ee719797(v=VS.85).aspx) (MS XPS), and **OpenXPS** (ECMA-388). Unless otherwise specified, v4 print drivers default to consuming MS XPS. Using the manifest directive XpsFormat, drivers may choose to support one or both of the available XPS formats.


4.1.7 XPS Rasterization Service Improvements
The XPS Rasterization Service has been improved in Windows 8 to make use of the Graphics Processing Unit (GPU) to provide faster XPS rasterization. These performance improvements are available on Windows 8 systems with GPUs that use the Windows Display Driver Model (WDDM) 1.2. XPS rendering filters do not require any modification to take advantage of this improvement, and it will be available for both v3 and v4 print drivers.

The XPS Rasterization Service can also provide rasterization in several pixel formats, including the following new, high precision formats. As a result, print drivers that use the XPS Rasterization Service can now target color precision at 8-bits, 16-bits and 32-bits per channel. For more information on pixel formats, see MSDN: [http://msdn.microsoft.com/en-us/library/ee719797(v=VS.85).aspx](http://msdn.microsoft.com/en-us/library/ee719797(v=VS.85).aspx). These new pixel formats are supported by the **XPSRasterizationFactory1::CreateRasterizer1()** method.

<table>
<thead>
<tr>
<th>Table 4. XPS Rasterization Service Pixel Formats</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
</tr>
<tr>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>GUID_WICPixelFormat32bppPBGRA</td>
</tr>
<tr>
<td>GUID_WICPixelFormat64bppPRGBAHalf</td>
</tr>
<tr>
<td>GUID_WICPixelFormat128bppPRGBAFloat</td>
</tr>
</tbody>
</table>

4.1.8 IPrintCoreHelperUni2
The **IPrintCoreHelperUni2** interface is provided to rendering filters in order to support retrieval of command strings from GPD files. The interface is identical to the **IPrintCoreHelperUni** interface, except for the additional method GetNamedCommand.
This interface is only supported on Windows 8. For more information on the API, see MSDN.

4.1.9 Print Class Driver Rendering

V4 print drivers may also take a dependency on a print class driver in order to render to device PDL. This is accomplished by using the v4 manifest directive RequiredClass. More information is available in the Version 4 Manifest section of this paper.

4.2 Color Management

Color management is supported in v4 print drivers. Drivers should include Windows Color System (WCS) compliant color profiles or International Color Consortium (ICC) color profiles. V4 print drivers may also use the driver property bag for device-specific color tables.

5 Configuration

The v4 driver model supports a greatly simplified configuration layer. Unlike v3 print drivers, where UI was strongly coupled with the configuration, v4 print drivers focus on providing PrintTicket, PrintCapabilities, and constraint functionality. A common configuration module, PrintConfig.dll, encapsulates the functionality that was previously available in the UnidrvUI and P55UI core drivers.

The v4 driver model does not employ configuration plug-ins, so most of the device configuration should be expressed in GPD or PPD files. In addition, v4 print drivers may provide a JavaScript file that supports advanced constraint handling as well as PrintTicket and PrintCapabilities support.

5.1 Configuration File Formats

The Generic Printer Description (GPD) and PostScript Printer Description (PPD) file formats are unchanged with v4 print drivers. Existing GPD and PPD files are compatible, however, all v4 print drivers must additionally specify the following directives in their GPD or PPD file. These directives prevent the expression of features that are not natively supported by XPSDrv, such as N-up.

### Table 5. Required Configuration File Directives

<table>
<thead>
<tr>
<th>GPD</th>
<th>PPD</th>
</tr>
</thead>
<tbody>
<tr>
<td>*Include: msxpsinc.gpd</td>
<td>*MSIsXPSDriver: True</td>
</tr>
</tbody>
</table>

PPD based drivers must not specify the *Include: msxpsinc.ppd directive as this is known to cause compatibility issues with some applications.

5.1.1 Mapping to the PrintSchema

Mapping features and options into the PrintSchema’s namespace is necessary in many cases. This causes the PrintCapabilities documents generated by your driver to be more compatible with standard print UI and applications.

Some features and options are considered standard and are automatically mapped into the PrintSchema’s namespace. These features and options are specified on MSDN and should not be remapped using *PrintSchemaKeywordMap. If not otherwise listed, drivers must use the *PrintSchemaKeywordMap directive on GPD based drivers, or the *MSPrintSchemaKeywordMap directive on PPD based drivers.
5.1.2 Copy Count
PPD based drivers v4 print drivers must specify the *MSXPSMaxCopies directive.

GPD based drivers that use the XPS->PCL6 rendering filter and do not support hardware copies must specify the *HardwareCopies directive. If set to ON, or not specified, this instructs the filter to send PDL to the device that will allow the device to handle multiple copies. Otherwise, if it is set to OFF, the filter will generate software copies.

5.1.3 No Punctuation Substitutions
Due to historical implementations using v3 print drivers, some devices may require punctuation characters such as period (.) or hyphen (-) are used in PrintCapabilities and PrintTicket implementations. The default behavior is that character substitutions will continue to occur. To configure punctuation character substitution, specify the following, root level attributes:

<table>
<thead>
<tr>
<th>GPD</th>
<th>PPD</th>
</tr>
</thead>
<tbody>
<tr>
<td>*NoPunctuationCharSubstitute?: TRUE</td>
<td>*MSPunctuationCharSubstitute: TRUE</td>
</tr>
</tbody>
</table>

5.2 JavaScript Constraints
The v4 driver model supports a new model for extended constraint and PrintTicket handling derived from the v3 IPrintOemPrintTicketProvider interface. Instead of using a compiled configuration plug-in, however, v4 print drivers will use a JavaScript that implements one or more of the following APIs.

The Windows Driver Kit includes a comprehensive sample JavaScript constraint file on the Windows Sample Gallery.

JavaScript constraints can be used to augment PrintCapabilities, validate PrintTickets and handle conversion of PrintTicket to DEVMODE and vice versa. However, JavaScript constraints have a few limitations. In particular:

- Features/options added in using CompletePrintCapabilities, as well as constraints specified in validatePrintTicket are not shown in the desktop printer preferences dialog.
- Features/options added in using CompletePrintCapabilities are not persisted into the public DEVMODE.
- JavaScript constraints cannot access language resources from resource dlls to localize added features/options or parameters.

As such, we recommend that JavaScript constraints are used only where appropriate. Features and options should be specified in the GPD or PPD where possible, and only complicated constraints should be represented in JavaScript.

5.2.1 API Reference
This section specifies the API entry points for use in the JavaScript constraint file.

5.2.1.20 validatePrintTicket
This API is called in order to validate that a PrintTicket object is valid for a particular printer. This is analogous in function to the IPrintOemPrintTicketProvider::ValidatePrintTicket API.

function validatePrintTicket(printTicket, scriptContext)
5.2.1.20.1 Parameters

`printTicket` – [in/out] The IPrintSchemaTicket object to be validated.
`scriptContext` – [in] The IPrinterScriptContext object that provides access to the driver property bag and the queue property bag.

5.2.1.20.2 Return Values

1 – Indicates that the printTicket parameter is a valid PrintTicket for this printer. Equivalent to `S_PT_NO_CONFLICT`.
2 – Indicates that the printTicket parameter was modified to make it valid. Equivalent to `S_PT_CONFLICT_RESOLVED`.
0 – Indicates that the printTicket parameter was invalid and could not be corrected. Equivalent to `E_PRINTTICKET_FORMAT`.

5.2.1.21 `completePrintCapabilities`

This API is called in order to allow the PrintCapabilities object to be modified. This should be used for conditional features (for example, borderless is only supported on photo paper) or to represent features that could not otherwise be generated by a GPD or PPD file (for example, nested feature definitions). This is analogous in function to the `IPrintOemPrintTicketProvider::CompletePrintCapabilities` API.

```cpp
function completePrintCapabilities(printTicket, scriptContext, printCapabilities)
```

5.2.1.21.1 Parameters

`printTicket` – [in] The IPrintSchemaTicket object input to constrain the generated PrintCapabilities document to.
`scriptContext` – [in] The IPrinterScriptContext object that provides access to the driver property bag and the queue property bag.
`printCapabilities` – [in/out] The IPrintSchemaCapabilities object representing the base PrintCapabilities object that was generated by the configuration module.

5.2.1.21.2 Return Values

None.

5.2.1.22 `convertDevModeToPrintTicket`

This API is called to convert values from the DEVMODE property bag into a PrintTicket. This is analogous in function to the `IPrintOemPrintTicketProvider::ConvertDevModeToPrintTicket` API, except that this implementation encapsulates the private section of the DEVMODE into an IPrinterScriptablePropertyBag object and allows no access to the public section of the DEVMODE.

```cpp
function convertDevModeToPrintTicket(devModeProperties, scriptContext, printTicket)
```

5.2.1.22.1 Parameters

**scriptContext** – [in] The IPrinterScriptContext object that provides access to the driver property bag and the queue property bag.

**printTicket** – [in/out] The IPrintSchemaTicket object representing the PrintTicket.

### 5.2.1.22.2 Return Values

None.

### 5.2.1.23 convertPrintTicketToDevMode

This API is called to convert values from a PrintTicket into the DEVMODE property bag. This is analogous in function to the `IPrintOemPrintTicketProvider::ConvertPrintTicketToDevMode` API, except that this implementation encapsulates the private section of the DEVMODE into an IPrinterScriptablePropertyBag object and allows no access to the public section of the DEVMODE.

```javascript
function convertPrintTicketToDevMode(printTicket, scriptContext, devModeProperties)
```

#### 5.2.1.23.1 Parameters

**printTicket** – [in] The IPrintSchemaTicket object representing the PrintTicket to be converted.

**scriptContext** – [in] The IPrinterScriptContext object that provides access to the driver property bag and the queue property bag.

**devModeProperties** – [in/out] The IPrinterScriptablePropertyBag object representing the DEVMODE Property Bag.

#### 5.2.1.23.2 Return Values

None.

### 5.2.2 Debugging

Basic syntactical validation of JavaScript files is supported by opening the JavaScript file in the Microsoft Windows Based Script Host. To do this, simply select the JavaScript file, Open With, and choose the Microsoft Windows Based Script Host entry in the list. If no errors are thrown, then the JavaScript is syntactically valid. Otherwise, it will point out the line number of the issue as shown in this screenshot.

![Figure 4. Windows Script Host error window](image)

Publicly available JavaScript validation tools may also be valuable as an aide in evaluating the style of JavaScript files.

Interactive debugging can be enabled by creating the registry key specified below; however, since PrintConfig.dll is loaded and unloaded frequently, attaching to an app that prints is not a recommended
testing/debugging strategy. Instead, Microsoft recommends that manufacturers build a test app that calls each of the relevant entry points for JavaScript constraints using the public APIs

**PTGetPrintCapabilities, PTCconvertDevModeToPrintTicket, PTCconvertPrintTicketToDevMode, and PTMergeAndValidatePrintTicket.**

The app itself is sufficient to enable debugging, but this is also a very good place to add unit tests to ensure that the whole driver is handling PrintTicket, PrintCapabilities and constraints as expected. For more information on how to build Unit tests in Visual Studio, please see:


**Key Name:** HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Print

**Value Name:** EnableJavaScriptDebugging

**Type:** DWORD

**Value:** 1

After the key above is created, and the hosting process has been restarted, the script can be debugged as follows:

1. Install Visual Studio 2011 on the test machine
2. Create a print queue using the driver that has the constraints JavaScript.
3. Set this print queue as the default.
4. Start your test app or an app that prints and begin a scenario that will cause JavaScript constraints to be invoked. The app must call into the PrintTicket/PrintCapabilities APIs in order to break into the JavaScript constraints; older apps like Notepad do not call into these APIs, but the XPS Viewer app does. Microsoft recommends using a test app here, since the scenarios can be more easily isolated and reproduced.
5. At this time, the “Visual Studio Just-In-Time Debugger” will pop up saying “An unhandled exception occurred in <your app>”
7. Choose Debug, then Attach To Process
8. In the Attach to Process dialog, ensure that Attach To: is set to Script code
9. Now choose the test app or app printing and finally choose Attach
10. Click on “Break All”
11. Now go back to the “Visual Studio Just-In-Time Debugger” dialog and click “No”
12. Visual Studio will break into the debugger at the location called by the current test. You may now debug the code normally.

### 5.3 Property Bags

In order to facilitate a data flow from customized UI applications to rendering, the v4 driver model provides a number of property bags. These allow custom properties and feature definitions to be created in customized UI and consumed by rendering.
All property bags are exposed by using the IPrinterscriptablePropertyBag interface in JavaScript or as the IPrinterPropertyBag interface in other environments. This table gives an overview of how to obtain this object from different parts of a v4 print driver.

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
</table>
| JavaScript constraint script | The driver and queue property bags are passed to JavaScript constraint scripts using the scriptContext parameter. This parameter is of type IPrinterScriptContext and contains children:  
DriverProperties – The driver property bag  
QueueProperties – The queue property bag  
The user property bag is not available in JavaScript constraints.  
The DEVMODE property bag is passed into the DEVMODE <-> PrintTicket conversion methods as the devModeProperties parameter as an IPrinterScriptablePropertyBag. It is unavailable on the other methods.  |
| USB Bidi JavaScript | The driver and queue property bags are passed to USB Bidi JavaScript scripts using the scriptContext parameter. This parameter is of type IPrinterScriptContext and contains children:  
DriverProperties – The driver property bag  
QueueProperties – The queue property bag  |
| Printer extension app | All property bags are passed in as part of the IPrinterExtensionEventArgs parameter to the OnDriverEvent handler. They are all of type IPrinterPropertyBag.  
They are specified as:  
DriverProperties – The driver property bag  
UserProperties – The user property bag  
PrinterQueue.GetProperties()– The queue property bag  |
<table>
<thead>
<tr>
<th><strong>Component</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Metro style device app</td>
<td>All property bags are passed in during activation using the IPrinterExtensionContext object. They are specified as:</td>
</tr>
<tr>
<td></td>
<td>DriverProperties – The driver property bag</td>
</tr>
<tr>
<td></td>
<td>UserProperties – The user property bag</td>
</tr>
<tr>
<td></td>
<td>PrinterQueue.GetProperties() – The queue property bag</td>
</tr>
<tr>
<td>XPS rendering filter</td>
<td>XPS filters can access the driver property bag from within the Print Filter Pipeline Property Bag using the property name &quot;DriverPropertyBag&quot;, or the defined value XPS_FPPROPERTY_BAG from filterpipeline.h.</td>
</tr>
<tr>
<td></td>
<td><strong>Property type:</strong> VT_UNKNOWN</td>
</tr>
<tr>
<td></td>
<td><strong>Description:</strong> A pointer to an IUnknown interface. Call QueryInterface to obtain a pointer to the IPrinterPropertyBag interface to the driver property bag.</td>
</tr>
<tr>
<td></td>
<td>XPS filters can access the queue property bag from within the Print Filter Pipeline Property Bag using the property name &quot;QueuePropertyBag&quot;, or the defined value XPS_FP_QUEUEPROPERTY_BAG from filterpipeline.h.</td>
</tr>
<tr>
<td></td>
<td><strong>Property type:</strong> VT_UNKNOWN</td>
</tr>
<tr>
<td></td>
<td><strong>Description:</strong> A pointer to an IUnknown interface. Call QueryInterface to obtain a pointer to the IPrinterPropertyBag interface to the queue property bag.</td>
</tr>
</tbody>
</table>

In JavaScript implementations, property bags are passed in as parameters. In printer extension applications, property bags are passed in as members of the event argument used to start the application.

The property bag accessors provided by the COM IPrinterQueue, IPrinterExtensionContext and IPrinterExtensionEventArgs interfaces as well as the property bag accessors in Javascript implementations will throw an exception if the property bag is not specified or not found. Additionally, querying for individual properties on an IPrinterPropertyBag interface will throw exceptions if the property is not found. Please use a try catch statement in order to avoid crashing if a property was not available.

### 5.3.1 Driver Property Bag

The driver property bag is a data store for drivers to predefined properties or data blobs for read-only use by the driver. It can be specified by using the "PropertyBag" directive in the v4 manifest file and may not be modified at runtime.
The Windows Driver Kit includes a template project for a driver property bag. The driver property bag is a compiled, binary blob. Visual Studio includes a template to generate a compiled driver property bag. The XML file generated for this template is not the property bag, instead the compiled output of this template is the property bag file that should be specified in the v4 manifest file.

5.3.2 User Property Bag

The user property bag allows partners to store settings in a per-user, per-application, machine local context. This property bag is well suited as a storage mechanism for user preferences like "Don't show this again". This property bag is not manageable by administrators and is not synchronized between client and server during printer sharing. The user property bag is only set at runtime and is only available to the printer extensions and Metro style device apps.

5.3.3 DEVMODE Property Bag

The DEVMODE property bag is used to organize content in the private section of the DEVMODE structure. During ConvertPrintTicketToDevMode calls, JavaScript is invoked to populate the contents of the DEVMODE property bag. During ConvertDevModeToPrintTicket calls, JavaScript is invoked to read the persisted settings from the DEVMODE property bag and store them back in the PrintTicket.

This property bag is limited in size to less than 60 KB (the exact amount will vary based on the size of the allocated sections of the DEVMODE), because it must be serialized into a DEVMODE structure in order to avoid data loss in some scenarios. The exact size available will vary per driver because it is determined by the size of the public section of the DEVMODE plus the private section managed by the configuration module.

The DEVMODE property bag uses an XML file to specify the members of the property bag and uses the convertPrintTicketToDevMode and convertDevModeToPrintTicket APIs to handle the conversions. The XML DEVMODE mapping file must be specified in the v4 manifest by using the DevModeMap directive.

5.3.3.24 DEVMODE Property Bag Mapping XML Sample

```xml
<xml version="1.0" encoding="utf-8"?>
  <Property Name="FabrikamAccountCode">
    <String Length="32"></String>
  </Property>
</Properties>
</Properties>
```

5.3.3.25 DEVMODE Property Bag Mapping XML Schema

The XML schema for the DEVMODE property bag in the Windows Driver Kit at \Include\um\printerdriverdevmodemap.xsd.
Figure 5. DEVMODE Property Bag XML Schema

5.3.3.26 DEVMODE Property Bag Mapping XML Validation
The XML files for the DEVMODE property bag mapping will be validated by the INFGate tool.

5.3.4 Queue Property Bag
The queue property bag stores per-queue configuration settings, including form to tray mappings and the configuration of printer properties like installable options. Driver defined properties are configurable in PowerShell, whereas form to tray mappings and printer properties are configurable in the printer properties UI. Printer extensions cannot edit any of the property values.

The queue property bag is an XML file and should not be compiled using the driver property bag tool.

Queue property bags are available for printers supported by v4 print drivers that:

1. Specify multiple trays OR
2. Specify installable options in the GPD or PPD OR
3. Specify a queue property bag in the driver manifest using the QueueProperties directive.

Administrators configure queue property bags using PowerShell. The following cmdlets are children of a printer object, which can be obtained using the Get-Printer cmdlet.

Table 7. Description of PowerShell Cmdlets Used To Edit Queue Property Bag

<table>
<thead>
<tr>
<th>Cmdlet Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get-PrinterProperty -printerName</td>
<td>Retrieves one or more properties (-name supports globbing)</td>
</tr>
<tr>
<td>&lt;printerName&gt; -name &lt;propertyName*&gt;</td>
<td></td>
</tr>
<tr>
<td>Cmdlet Name</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Set-PrinterProperty</td>
<td>Changes a print queue property using a persisted printerPropertyObject.</td>
</tr>
<tr>
<td>-inputObject</td>
<td></td>
</tr>
<tr>
<td>&lt;printerPropertyObject&gt;</td>
<td></td>
</tr>
<tr>
<td>set-printerproperty</td>
<td>Changes the property specified to the value specified.</td>
</tr>
<tr>
<td>-printerName</td>
<td></td>
</tr>
<tr>
<td>&lt;printerName&gt;</td>
<td></td>
</tr>
<tr>
<td>-PropertyName</td>
<td></td>
</tr>
<tr>
<td>&lt;propertyName&gt;</td>
<td></td>
</tr>
<tr>
<td>-Value</td>
<td></td>
</tr>
<tr>
<td>&lt;value&gt;</td>
<td></td>
</tr>
</tbody>
</table>

### 5.3.4.27 Installable Options

Installable options, such as the state of a duplexer, will be exposed into the queue property bag as individual properties. Each property will be named as follows, where the feature name is based on the name of the feature from the driver’s GPD or PPD file:

- Config:<feature name>
- Eg: Config:DuplexUnit

The value of the property is keyword name for the option that has been selected by the administrator.

- Eg: Installed

Installable options are editable using the same Set-PrinterProperty cmdlet that is used for queue properties.

### 5.3.4.28 Form to Tray Mappings

For printers with more than one tray, supported with a v4 print driver, form to tray mappings are exposed via the queue property bag in the property named “FormTrayTable“.

This property is formatted as a null terminated string containing pairs of the format “<tray name>,<form name>,”, where the form name is one of the following:

1. If the paper size is mapped into the Print Schema in the GPD or PPD file (either by using standard *PaperSize/*PageSize keywords, or *(MS)PrintSchemaKeywordMap), then the form name will follow the following format:
   a. PrintSchema:<Paper Size name>
   b. Eg: PrintSchema:NorthAmericaLetter

2. If the form is a user-defined form, as determined by the FORM_USER flag, then the form name will be as follows. The form index is the same value used in the spooler’s form database. This is consistent with index used when the paper size is specified in the PrintTicket as UserForm<Form index>.
   a. UserForm<form index>
b. Eg: UserForm123

3. Otherwise, the form name will follow the following format, where the form name is the name that was specified in the GPD's *PaperSize or the PPD's *PageSize
   a. Config:<name>
   b. Eg: Config:_8_5x16

A full example string would be:
"Config:Tray1,PrintSchema:NorthAmericaLetter,Config:Tray2,Config:_8_5X16,Config:Manual,UserForm123,\0"

Rendering filters should read the incoming PrintTicket’s PageMediaSize setting, and search for that value in the form name values from the FormTrayTable.

5.3.4.29 Queue Property Bag XML Sample

```xml
<?xml version="1.0" encoding="utf-8"?>
    <Property Name="Name1">
        <String>String1</String>
    </Property>
    <Property Name="Name2">
        <Int32>3244</Int32>
    </Property>
    <Property Name="Name3">
        <Bool>true</Bool>
    </Property>
</Properties>
```

5.3.4.30 Queue Property Bag XML Schema

The queue property bag XML schema is available in the Windows Driver Kit at \Include\um\printqueueproperties.xsd.

![Diagram of XML schema](image-url)
5.4 Localization

V4 print drivers support the use of a single, string-only resource DLL per driver. These may not include any other functionality, resources, or executable code. Resource DLLs must be specified in the v4 manifest file as described in this paper.

5.4.1 Standard Strings

In order to support development of printer extensions and Metro style device apps, Windows is providing standard, localized display strings through the new IPrintSchemaCapabilities objects to support the following features and the associated standard options:

- Input bins
  - Job/Document/PageInputBin
- Media types
  - PageMediaType
- Duplexing
  - JobDuplexAllDocumentsContiguously
- Collation
  - DocumentCollate
- Output color
  - PageOutputColor
- Orientation
  - PageOrientation
- N-Up
  - JobNUpAllDocumentsContiguously
- Hole punching
  - JobHolePunch
  - DocumentHolePunch
- Stapling
  - JobStapleAllDocuments
  - DocumentStaple
- Binding
  - JobBindAllDocuments
  - DocumentBinding
- Output quality
  - PageOutputQuality
- Media Size
  - PageMediaSize

In addition, these strings are available in the XML forms of PrintCapabilities, provided that the driver does not specify a display name using a resource DLL for the feature or option. If a driver does specify a display name using a resource DLL, it will be provided in the XML, as well as to the legacy COMPSTUI-based print preferences UI used on previous versions of Windows.
Across the different user interfaces and APIs, the display names vary. Below is an outline of which strings will be used in which circumstances:

**Figure 7. Flowchart of Expected Localization Behavior**

To use the Microsoft localized display names, follow the instructions in this table.
<table>
<thead>
<tr>
<th>GPD</th>
<th>PPD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use *PrintSchemaKeywordMap to map GPD features or options to the corresponding Print Schema-defined features or options, unless they are specified as standard. Specify the following entry for the GPD feature or option: *Name</td>
<td>Use *MSPrintSchemaKeywordMap to map PPD features or options to the corresponding Print Schema-defined features or options. For N-Up, do not use *MSPrintSchemaKeywordMap on the option values. Do not specify:</td>
</tr>
</tbody>
</table>

More information on the standard strings is available in section 11.

6 Customized User Interfaces

V4 print drivers support customized UI in both the Windows desktop and in the Metro style UI. Due to the very different nature of these experiences, these UIs must be implemented as two different applications, however both are built upon a common COM API provided by the configuration module.

Printer extensions support v4 print drivers in the desktop and work with all existing applications. They also work in printer sharing scenarios with the Enhanced Point and Print driver. Support is planned for all operating systems from Windows Vista through Windows 8.

Metro style device apps support v4 print drivers in the Metro style UI. More information on developing Metro style device apps is available in the Developing Metro style Device Apps for Printers whitepaper.
Figure 8. High Level Overview of Communication Architecture Between Customized User Interfaces
and Print System

6.1 Architecture

One of the high level goals of the v4 driver architecture was a design for the new Windows Metro style
user interface. The application-based UI paradigm that is employed is a clear example of this. Metro
style device apps provide users with a full screen experience that is supported in the Metro UI. Specific
to printers supported with v4 print drivers, Metro style device apps for printing provide extensibility for
print preferences and printer notifications and also give the device a presence on the new Start
experience.

Printer extension apps support print preferences and printer notifications while users are running the
Windows desktop and using existing applications. While the UI for these applications is very different,
with one tailored for touch and the other optimized for mouse and keyboard users, the business logic
and connection to v4 print drivers is very likely common regardless of the UI.

The architecture below shows the high level architecture of the Metro style device apps for printing and
printer extension samples that are provided on the Windows Sample Gallery. The model/view/controller
architecture employed enables the apps to share code at the model layer, written in C#.
### V4 printer driver UI customization architecture

<table>
<thead>
<tr>
<th>Metro style device app for printing</th>
<th>Common Code</th>
<th>Printer extension</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="DeviceAppForPrinters" /></td>
<td><img src="image2" alt="PrinterExtensionLibrary" /></td>
<td><img src="image3" alt="PrinterExtensionSample" /></td>
</tr>
<tr>
<td><img src="image4" alt="DeviceAppForPrintersLibrary" /></td>
<td><img src="image5" alt="PrintDialogs.dll" /></td>
<td><img src="image6" alt="PrintDialogs.dll" /></td>
</tr>
</tbody>
</table>

**Figure 9. V4 Print Driver UI Customization Architecture**

#### 6.1.1 Extending PrinterExtensionLibrary

The PrinterExtensionLibrary project that ships in the various samples can be extended using new classes, or by extending the provided set of classes. Since Microsoft periodically makes updates to the sample code, we recommend that partners minimize the number of code changes to the provided source files. For partners that are extending the provided set of classes, we recommend that you mark the existing classes as “partial” and add new functions or overrides in a separate source file.

#### 6.1.2 Sharing Compiled Binaries Between Metro style and Desktop Apps

The PrinterExtensionLibrary project that is shipped in the Metro style device app and printer extension samples utilizes the same source code, but it may be valuable to build the code so it is portable between the projects without being built separately.

To convert the PrinterExtensionLibrary project to a Portable Class Library:
1. In Visual Studio, Go to File, New, Project, then search for Portable. Choose Portable Class Library Visual C#. Click through and name the project.

2. Copy the source code from your existing PrinterExtensionLibrary project into the new project.

3. Right click on your Portable Class Library project and choose Unload. Open the .csproj file and add the following section to your file, just prior to the last tag in the document.

   ```xml
   <ItemGroup>
   <COMReference Include="PrinterExtensionLib">
   <Guid>{91CE54EE-C67C-4B46-A4FF-99416F27A8BF}</Guid>
   <VersionMajor>1</VersionMajor>
   <VersionMinor>0</VersionMinor>
   <Lcid>0</Lcid>
   <WrapperTool>tlbimp</WrapperTool>
   <Isolated>False</Isolated>
   <EmbedInteropTypes>True</EmbedInteropTypes>
   </COMReference>
   </ItemGroup>
   
   4. If you see warnings as a result of COM references, add the following to the <PropertyGroup> tag:

   ```xml
   <ResolveComReferenceSilent>true</ResolveComReferenceSilent>
   ```

6.2 API

Supporting printer extensions and Metro style device apps for printing is a new API for interacting with the print driver. At a high level, the print preferences scenario uses PrintTicket, PrintCapabilities and the new property bags to get and store all of its information. Printer notifications are driven by a new eventing system that is based on the Bidirectional Communication Schema and uses the AsyncUI protocol between client and server. The data-centric nature of this API means that one application could easily support many devices.

Printer extensions need to be built to gracefully degrade if required data is unavailable. For example, if a particular PrintCapabilities feature is unavailable, or if a property in one of the property bags is unavailable, this should not prevent the rest of the app from functioning. When accessing property bags, or specific properties in a property bag, the app should use try catch syntax in order to ensure that any exceptions that are thrown do not cause the app to crash.

The full details of the API are described on MSDN.

6.3 Driver Support for Customized UI

In order to build printer extensions or Metro style device apps, the print queue being targeted must be installed with a v4 print driver. Additional design considerations are described in the following sections.
6.3.1 Print Preferences

All v4 print drivers work with print preferences, however, it is important to maintain the boundaries between the configuration and UI layers in order to ensure maximum consistency across all scenarios. Since printer extensions and Metro style device apps may not be installed (or installed automatically), v4 print drivers need to ensure that they are functional without a customized printer preferences experience. In particular, this means that the PrintTicket and PrintCapabilities support should be complete and comprehensive in the GPD/PPD + JavaScript constraints implementation in the driver.

Some constraint validation in the printer extension or Metro style device app may be good in terms of providing a highly informative, interactive experience, but it should not replace the driver's validation, which is considered authoritative.

Printer extensions and Metro style device apps should use the SendBidiQuery API instead of making any direct network calls to a network resource. If a network resource must be contacted, it should be done on another thread, or asynchronously in order to prevent UI hangs. Data should be cached after it is retrieved to make future calls faster.

6.3.2 Printer Notifications

Printer notifications are driven by Bidi and the DriverEvent XML file. In order to better manage battery life and to minimize interruptions, however, notifications will only be shown when a user is printing.

While printer preferences is contextual to the app that was printing, printer notifications are not. The flow chart below explains the decision tree that Windows uses. If available, Metro style device apps take precedence over printer extensions.
Figure 10. Description of Notification Behavior

6.3.2.31 Creating Driver Events

Using a Driver Event XML file, drivers describe Bidi queries and triggers that should cause an event to occur. Driver events support only standard strings. In the current implementation, this will cause an AsyncUIBalloon message to be created and published using the MS-PAN protocol. This implementation may change in the future to enable better performance, so it is crucial that drivers not take dependencies on the underlying protocol.

The diagram below shows the protocol utilization. The box “invoke appropriate app” corresponds with the diagram in 6.5.2.
6.3.2.32 Driver Event XML Sample

The sample in this section specifies one driver event. The event checks for yellow ink to be less than 21% of the total capacity as reported by Bidi. If this occurs, an AsyncUIBalloon message is created with the string referenced by resourceID 132 or "%1 is low on toner/ink." Resource 2002, "Yellow", would be substituted for %1.

```xml
<?xml version="1.0" encoding="utf-8"?>
  <DriverEvent eventid="{AO4CFOFC-1CEB-4C62-B967-6FOAE5C5F81E}">
    <Transport>USB</Transport>
    <Transport>WSD</Transport>
    <Query>
      <Printer.Consumables></Query>
      <Trigger result="\Printer.Consumables.Yellow:Level" comparison="LessThan" value="21">
        <StandardMessage resourceid="132">
          <StringParameter index="1" resourceid="2002"/>
        </StandardMessage>
      </Trigger>
    </Query>
  </DriverEvent>
</de:DriverEvents>
```
6.3.2.33 Driver Event Schema

The DriverEvent schema is available in the Windows Driver Kit as \Include\um\PrinterDriverEvents.xsd.
Figure 12. DriverEvent XML Schema

6.3.2.34 Driver Event XML Validation
Driver Event XML is validated automatically by the INFGate tool when described properly in the driver manifest.
6.4 Printer Extensions

Printer extensions may be built in any COM capable language, but are optimized to be built on Microsoft .NET Framework 4. They may be distributed with a print driver package if they are XCopy capable and have no dependencies on external runtimes other than those included with the operating system, like .NET. If the application doesn’t meet these criteria, it could be distributed in a setup.exe or MSI (with or without the print driver) and advertised in the printer’s Device Stage™ experience and/or using the PrinterExtensionUrl directive specified in the v4 manifest. The PrinterExtensionUrl is shown on the printer preferences experience.

IT administrators also have a few options for managing the distribution of printer extensions. If the application is packaged in a setup.exe or MSI, IT administrators can use standard software distribution tools such as System Center Configuration Manager (SCCM) or include the applications in their standard OS image. Finally, the PrinterExtensionUrl specified in the v4 manifest can be overridden by IT admins if they edit HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Print\Printers\<print queue name>\PrintDriverData\PrinterExtensionUrl.

Some enterprises may also choose to block printer extensions altogether, though. This is supported by means of a group policy called "Computer Configuration\Administrative Templates\Printers\Do not allow v4 printer drivers to show printer extension applications".

6.4.1 Building a Printer Extension

The Windows Sample Gallery includes a sample called PrinterExtensionSample that demonstrates how to build a printer extension using C#. In order to allow code sharing between Metro style device apps and printer extensions, this sample uses two projects: PrinterExtensionLibrary is a C# interop library and PrinterExtensionSample is a printer extension that is dependent on the PrinterExtensionLibrary.

The code snippets below are all taken from the PrinterExtensionSample solution. If you are building a printer extension in C, C++ or some other COM based language the concepts are similar, but the APIs must instead match those specified in PrinterExtension.IDL, which is included in the Windows Driver Kit. The code comments in the PrinterExtensionLibrary from the sample document also include code comments which indicate which underlying COM interface a particular object corresponds to.

6.4.1.35 Registration

Printer extensions are registered with the print system by specifying a set of registry keys or by specifying the application information in the PrinterExtensions section of the v4 manifest file.

6.4.1.35.1 Printer Extension Reasons

There are specified GUIDs that support each of the different entry points for printer extensions. Using these in the v4 manifest file is not necessary, but installation using the registry format requires knowledge of the GUID values.

<table>
<thead>
<tr>
<th>Table 9. Printer Extension Reasons</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Entry Point</strong></td>
</tr>
<tr>
<td>Print Preferences</td>
</tr>
<tr>
<td>Printer Notifications</td>
</tr>
</tbody>
</table>

May 14, 2012
© 2012 Microsoft. All rights reserved.
6.4.1.35.2 Registry Format

Printer extensions that are installed outside of the printer driver need to be registered using the following registry key format. This ensures that printer extensions can be installed regardless of the status of the spooler, or the v4 configuration module on the client machine.

Once the PrintNotify service starts, it will check for registry keys under the [OfflineRoot] path and process any pending registrations or unregistrations. Once any pending registrations or unregistrations are completed, the registry keys are deleted in real time. Note that if you are using a script or iterative process to place registry keys, you may need to recreate the \[PrinterExtensionID] key every time a \[PrinterDriverId] key is to be specified. Incomplete or malformed keys are not deleted.

This registration is only necessary on first install.

[OfflineRoot]: HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Print\OfflinePrinterExtensions

[OfflineRoot]
\[PrinterExtensionId] {GUID}
   AppPath=[PrinterExtensionAppPath] {String}
   \[PrinterDriverId] {GUID}
      \[PrinterExtensionReasonGuid]
         {default} = ["0"|"1"] {REG_SZ 0:Unregister, 1:Register}
      ...
      \[PrinterExtensionReasonGuidN]
      \[PrinterDriverId2]
         \[PrinterExtensionReasonGuid2.1]
         ...
         \[PrinterExtensionReasonGuid2.Z]
      ...
      \[PrinterDriverIdM]
      \[PrinterExtensionId2]
      ...
      \[PrinterExtensionIdT]

For example, the following set of keys would register a printer extension with the PrinterExtensionID {PrinterExtensionIDGuid} and a fully-qualified path to the executable “C:\Program Files\Fabrikam\pe.exe” for the PrinterDriverIDs {PrinterDriverId1Guid} and {PrinterDriverId2Guid} with the printer preferences and printer notifications reasons.

[OfflineRoot]
\{PrinterExtensionIDGuid}
   AppPath=“C:\Program Files\Fabrikam\pe.exe”
   \{PrinterDriverId1Guid}
      \{EC8F261F-267C-469F-B5D6-3933023C29CC} (default) = “1”
      \{23BB1328-63DE-4293-915B-A6A23D929ACB} (default) = “1”
   \{PrinterDriverId2Guid}
      \{EC8F261F-267C-469F-B5D6-3933023C29CC} (default) = “1”
      \{23BB1328-63DE-4293-915B-A6A23D929ACB} (default) = “1”

To uninstall the same printer extension, the following set of keys should be specified.
[OfflineRoot]
Since printer extensions can run in both a user launched context and an event launched context, it is useful to know which context you are in deterministically. This can allow an app to, for example, not enumerate the status on all queues if it has been launched for a notification or print preferences. Microsoft recommends that printer extensions which are installed separately from the driver (e.g. with an MSI or setup.exe) use command line switches either on the start menu shortcuts created, or the in the AppPath entry populated in the registry during registration. Since printer extensions which are installed with the driver are installed to the DriverStore, these will not be launched outside the print preferences or printer notifications events, and therefore specifying command line switches is unsupported in this case.

### 6.4.1.36 Enabling Events

At runtime, printer extensions must enable eventing for each PrinterDriverID they support. This allows the print system to provide an appropriate event context to handle reasons like print preferences or printer notifications. In order to maximize the flexibility of each printer extension, each instance of the PrinterExtensionManager class handles exactly one PrinterDriverID and one OnDriverEvent handler. While Microsoft recommends that the same OnDriverEvent handler is used so that code scattering is minimized, it may be beneficial to use a different OnDriverEvent handler with some PrinterDriverIDs to reduce complexity.

So for each PrinterDriverID supported, the application should create a new PrinterExtensionManager, register a delegate to handle the OnDriverEvent event, and call the EnableEvents method with a PrinterDriverID.

```csharp
PrinterExtensionManager mgr = new PrinterExtensionManager();
mgr.OnDriverEvent += OnDriverEvent;
mgr.EnableEvents(new Guid(PrinterDriverID1));
```

If an app does not call EnableEvents within 30s, we will timeout and launch standard UI.

### 6.4.1.37 OnDriverEvent Handler

After an OnDriverEvent handler is registered and events are enabled, if the printer extension was launched to handle print preferences or printer notifications, then the handler will be invoked. In the code in the previous section, the handler registered was a method called OnDriverEvent. The PrinterExtensionEventArgs (wrapper for IPrinterExtensionEventArgs) parameter is the object that enables the print preferences and printer notifications scenarios to be constructed.

```csharp
static void OnDriverEvent(object sender, PrinterExtensionEventArgs eventArgs)
```
In order to prevent a bad user experience associated with crashing or slow printer extensions, Windows implements a timeout if EnableEvents is not called within a short amount of time after the app is launched. To enable debugging, this timeout is disabled if there is a debugger attached to the PrintNotify service.

In most cases, however, all of the interesting code in the app occurs during or after the OnDriverEvent callback. During development, it may also be useful to show a MessageBox prior to starting either a print preferences or printer notifications experience from the OnDriverEvent callback. When you see this, go back to Visual Studio, choose Debug > Attach to Process and choose the name of your process. Finally, go back to your MessageBox and click OK to resume. This will ensure that you see exceptions and hit any breakpoints from that point onward.
New ReasonIds may be supported in the future. As a result, printer extensions must explicitly check the
ReasonID and must not use an “else” statement to detect the last known ReasonID. If a ReasonID is
received and unknown, the app should exit gracefully.

6.4.1.38 Print Preferences
Print preferences is driven by the PrintSchemaEventArgs.Ticket object. This object encapsulates both the
PrintTicket and PrintCapabilities documents that describe the features and options for a device. While the
underlying XML is also available, the object model makes working with these formats easier.

Inside each IPrintSchemaTicket or IPrintSchemaCapabilities object are features (IPrintSchemaFeature)
and options (IPrintSchemaOption). While the interfaces used for features and options are the same
regardless of the origin, the behavior varies slightly as a result of the underlying XML. For example,
PrintCapabilities documents specify many options per feature, while PrintTicket documents specify only
the selected (or default) option. Similarly, PrintCapabilities documents specify localized display strings,
whereas PrintTicket documents do not.

The PrinterExtensionSample uses data binding to create ComboBox controls for printer preferences.
Microsoft recommends that partners use data binding as it makes the code much easier to maintain by
reducing scattering. For more information on data binding in WPF please see MSDN.

In order to maximize performance, Microsoft recommends that calls to GetPrintCapabilities are only
done when updated PrintCapabilities document is necessary.

6.4.1.38.1 Committing PrintTicket Changes
As a user makes choices using the data bound ComboBox controls, the PrintTicket object is
automatically updated. When the user finally clicks “OK”, a chain of asynchronous validation and
completion begins. This asynchronous pattern is used extensively in order to prevent long running tasks
from occurring on UI threads and hanging either the print preferences UI or the app that is printing.

1. The PrintSchemaTicket is validated asynchronously using the
   IPrintSchemaTicket::ValidateAsync() method.
2. When the asynchronous validation completes it invokes the method called
   PrintTicketValidateCompleted
   a. If validation was successful, it calls the CommitPrintTicketAsync method.
      i. This method calls the IPrintSchemaTicket::CommitAsync() method.
         1. When complete, this invokes the PrintTicketCommitCompleted
         method, which calls a convenience method that calls the
         PrinterExtensionEventArgs.Request.Complete() method to indicate
         that print preferences are complete, and then it closes the app.
   b. Otherwise, it presents UI to the user to handle the constraint situation.

If the user clicked cancel or closed the print preferences window directly, the printer extension will call
IPrinterExtensionEventArgs.Request.Cancel() with an appropriate HRESULT and message for the log.
If the process for the printer extension has closed and not called the Complete() or Cancel() as above, the print system will automatically fall back to using Microsoft-provided UI.

### 6.4.1.38.2 Getting Status from Bidi

In order to show ink status or other kinds of status about the device, printer extensions can use the IPrinterExtensionEventArg.PrinterQueue.SendBidiQuery() method to issue Bidi queries to the device. Getting the latest Bidi status is a two-step process involving setting up an event handler for the OnBidiResponseReceived event, and calling the SendBidiQuery() method with a valid Bidi query.

```csharp
PrinterQueue.OnBidiResponseReceived += new
    EventHandler<PrinterQueueEventArgs>(OnBidiResponseReceived);
PrinterQueue.SendBidiQuery("\Printer.consumables");
```

When the Bidi response is received, the event handler below will be invoked. Note that this event handler also has a mocked ink status implementation, which may be useful for development when a device is unavailable. The PrinterQueueEventArgs object includes both an HRESULT and a Bidi XML response. For more information on Bidi XML responses see: http://msdn.microsoft.com/en-us/library/dd183368(VS.85).aspx.

```csharp
private void OnBidiResponseReceived(object sender, PrinterQueueEventArgs e)
{
    if (e.StatusHRESULT != int HRESULT.S_OK)
    {
        MockInkStatus();
        return;
    }

    // Display the ink levels from the data.
    //
    BidiHelperSource = new BidiHelper(e.Response);
    if (PropertyChanged != null)
    {
        PropertyChanged(this, new PropertyChangedEventArgs("BidiHelperSource"));
    }
    InkStatusTitle = "Ink status (Live data)";
}
```

### 6.4.1.39 Printer Notifications

Printer notifications are invoked in precisely the same way as print preferences. In the OnDriverEvent handler, if the IPrinterExtensionEventArgs indicates that a ReasonID matches the DriverEvents GUID, then we can build an experience to handle this event.

The PrinterExtensionSample project does not demonstrate a functional printer notifications experience, but the following variables are most helpful in handling this.
- PrinterExtensionEventArg.BidiNotification – This carries the Bidi XML that caused the event to be triggered.
- PrinterExtensionEventArg.DetailedReasonId – This contains the eventID GUID from the driver event xml file.
The most important attribute from the IPrinterExtensionEventArgs object for notifications is the BidiNotification element. This carries the Bidi XML that caused the event to be triggered. For more information on Bidi XML responses, see MSDN: http://msdn.microsoft.com/en-us/library/dd183368(VS.85).aspx.

### 6.4.1.40 Managing Printers

In order to support the role of a printer extension as an app that can be used as a hub for managing/maintaining printers, it is possible to enumerate the print queues for which the current printer extension is registered and get their status. This is not demonstrated in the PrinterExtensionSample project, but the code below could be added into the Main() method of App.xaml.cs to register an event handler.

```csharp
mgr.OnPrinterQueuesEnumerated += new EventHandler<PrinterQueuesEnumeratedEventArgs>(mgr_OnPrinterQueuesEnumerated);
```

Once the queues are enumerated, the event handler will be called and status operations can take place. This event will fire periodically during the lifetime of the app in order to ensure that the list of print queues enumerated is current even if the user has installed more queues since it was opened. As a result, it is important that the event handler below does not create a new window each time it is executed.

```csharp
static void mgr_OnPrinterQueuesEnumerated(object sender, PrinterQueuesEnumeratedEventArgs e)
{
    foreach (IPrinterExtensionContext pContext in e)
    {
        // show status
    }
}
```

In order to perform maintenance tasks using a printer extension, Microsoft recommends that the legacy WritePrinter API is used as outlined below. For more information on how to marshal these legacy APIs into .NET, please see the resources How to send raw data to a printer by using Visual C# .NET or How to send raw data to a printer by using Visual Basic .NET.

1. OpenPrinter
   a. StartDocPrinter
      i. StartPagePrinter
      1. WritePrinter
      ii. EndPagePrinter
   b. EndDocPrinter
2. ClosePrinter
6.5 Metro style Device Apps for Printers

Metro style device apps are distributed by the Windows Store and automatically installed for users who have opted in to download applications. Metro style device apps for printers may be provided for any printer on Windows.

Printers that are installed using v4 print drivers can provide a customized printer preferences experience by supporting the printTaskSettings contract or support printer notifications by supporting the backgroundTasks contract. More information on Metro style device apps for printers is available in the Developing Metro style Device Apps for Printers paper.

7 Setup

The version 4 driver model will use a new setup model in order to improve user experiences and reduce support costs. V4 print drivers are run directly from the Driver Store, which eliminates the possibility of file collisions and improves install performance. The v4 setup model continues to use INF files, but also employs a new manifest file to capture the printer specific setup directives.

7.1 Version 4 INF

This section shows sample v4 printer INFs. With the exception of the ClassVer=4.0 directive included in the Version section, there are no printer-specific directives included in this INF. The printer-specific instructions are contained in the v4 manifest file, which is always named ending with "--manifest.ini". Each driver within a driver package may specify its own v4 manifest file.

7.1.1 INF Sample

```plaintext
[Version]
Signature="$Windows NT$"
Provider="Fabrikam"
ClassGUID={4D36E979-E325-11CE-BFC1-08002BE10318}
Class=Printer
CatalogFile=prnfa999.CAT
DriverVer=09/12/2010.6.2.8060.4
ClassVer=4.0 ;This causes v4 setup to take place

[Manufacturer]
"Fabrikam"=Fabrikam,NTamd64

[Fabrikam,NTamd64] ;Add your models here
"Fabrikam Laser 9000" = Laser9000,Fabrik9000_sdfjkals ;HWID example
"Fabrikam Laser 9100" = Laser9000,Fabrik9100_sjkasj ;HWID example
"Fabrikam Laser 9000 series" = Laser9000,[EO691E8C-F7CC-456E-A785-D1FC19BA2279] ;PrinterDriverID

[Laser9000]
```
CopyFiles=Laser9000_FILES

[Laser9000_FILES]
faPDL_gpd
faPDL-pipelineconfig.xml
faPDL-manifest.ini
faPDL.dll

[SourceDisksNames.amd64]
1 = %Location%%, 
2 = %Location%%,amd64

[SourceDisksNames.x86]
1 = %Location%%, 
2 = %Location%%,x86

[DestinationDirs]
DefaultDestDir=66000

[SourceDisksFiles]
faPDL_gpd=1
faPDL-pipelineconfig.xml=1
faPDL-manifest.ini = 1
faPDL.dll =2

[Strings]
Location="Fabrikam DVD"

7.1.2 Print Class Driver GenericDriverInstalled Properties

Print class drivers need to specify that they are generic drivers in order to enable automatic upgrade to a newer print driver on Windows Update. Note: This property is allowed for inbox drivers only.

For inbox drivers, this property should be specified in the install section for the driver. This is a two-part declaration.

1. Add the "AddProperty=GENERIC.AddProp" line to all install sections that are class drivers.
2. Add the [GENERIC.AddProp] section.
   a. Add the "GenericDriverInstalled,,,,1" line under this section.
7.1.3 Directives

This table shows the list of printer-specific directives that are permitted in v4 print drivers and print class drivers.

**Table 10. Printer Specific v4 INF Directives**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Restrictions</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClassVer</td>
<td>Used to indicate that a printer class driver is v4.</td>
<td>V4 print drivers must specify ClassVer=4.0. V3 print drivers may specify ClassVer=3.0, but it is optional. No other values are supported at this time.</td>
<td>ClassVer=4.0</td>
</tr>
</tbody>
</table>

7.1.4 DestinationDirs

The v4 driver INF requires that DestinationDir is specified for all files in the package. The supported DestinationDirs are listed in this table.

**Table 11. Printer Driver DestinationDirs**

<table>
<thead>
<tr>
<th>DirID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>66000</td>
<td>[Overloaded] V4: This must be set as the DefaultDestDir for a v4 print driver. Specifies that the files should be run from the Driver Store. V3: This specifies that files should be installed to the \3 directory.</td>
</tr>
<tr>
<td>23</td>
<td>V4: This must be set as the DestinationDir for any color profiles. V3: Color profiles should be installed using the printer-specific DirID 66003.</td>
</tr>
</tbody>
</table>

7.1.5 V4 INF Restrictions

V4 print drivers must not define other printer-specific directives or items called out in this list.

- AddInterface – Directive
- AddReg – Directive
- AddService – Directive
- BitReg – Directive
- ClassInstall32 – Section Type
- ClassInstall32.Services – Section Type
- ConfigFile – v3 Print Directive
- CoreDriverDependencies – v3 Print Directive
- CoreDriverSections – v3 Print Directive
- DataFile – v3 Print Directive
- DDInstall.Coinstallers – Section Type
- DDInstall.FactDef – Section Type
- DDInstall.HW – Section Type
- DDInstall.Interfaces – Section Type
- DDInstall.LogConfigOverride – Section Type
- DDInstall.Services – Section Type
- DDInstall.WMI – Section Type
- DefaultInstall – Section Type
- DefaultInstall.Services – Section Type
- DelFiles – Directive
- DelReg – Directive
- DelService – Directive
- DontReflectOffline
- DriverFile – v3 Print Directive
- DriverIsolation – v3 Print Directive
- FeatureScore – Directive
- HelpFile – v3 Print Directive
- Include – Directive
- Ini2Reg – Directive
- InterfaceInstall32 – Section Type
- LayoutFile - Directive
- LogConfig – Directive
- Needs – Directive
- PackageAware – v3 Print Directive
- RenFiles – Directive
- UpdateIniFields – Directive
- UpdateInis – Directive

### 7.1.5.41 NTPrint References

NTPrint references are made in the manifest file. The INF does not require any information about the NTPrint references in the DDInstall, CopyFiles, or SourceDisksFiles sections of the INF.

### 7.1.5.42 Configuration Module References

All drivers use the same configuration module binary (PrintConfig.dll); there is no mechanism for a driver to select the configuration module.
7.1.6 Model Lines
Plug and Play maintains an implicit ranking of all HardwareIDs and CompatibleIDs on a model line. As a result, Microsoft recommends that partners follow these best practices to avoid unpredictable behaviors with regard to ranking.

7.1.6.43 V4 Print Driver

1. V4 print driver INFs must define two different kinds of model lines:
   a. **HardwareID lines**: "Driver name" = INSTALL_SECTION, busenumerator\HardwareID
   b.  **PrinterDriverID lines**: "Driver name" = INSTALL_SECTION,{GUID}

2. V4 print driver INFs must define bus-specific HardwareIDs on individual lines:
   a. "Driver name" = INSTALL_SECTION,WSDPRINT\HardwareID
   b. "Driver name" = INSTALL_SECTION,USBPRINT\HardwareID
   c. "Driver name" = INSTALL_SECTION,LPTENUM\HardwareID

7.1.6.44 Print Class Driver

1. Print class driver INFs must define three different kinds of model lines:
   a. **HardwareID lines**: "Driver name" = INSTALL_SECTION,HardwareID
   b. **PrinterDriverID lines**: "Driver name" = INSTALL_SECTION,{GUID}
   c. **CompatibleID lines**: "Print Class Driver name" = INSTALL_SECTION,,1284_CID_CompatID

2. Print class driver INFs must not define any bus enumerators (for example, WSDPRINT\)

7.2 Version 4 Manifest
The v4 manifest is where all of the printer-specific setup directives are specified.

7.2.1 DriverConfig Section

**Table 12. Description of Directives Under DriverConfig Section of v4 Manifest**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Restrictions</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>RequiredFiles</td>
<td>Includes files from ntprint.inf or ntprint4.inf.</td>
<td>Unidrv.dll, pscript5.dll, and mxwdrv.dll should be omitted from this list. They will be resolved automatically.</td>
<td>RequiredFiles=UNIRES.DLL,STDNAMES.GPD,V3HOSTINGFILTER.DLL</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Restrictions</td>
<td>Usage</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>RequiredClass</td>
<td>Causes this driver to include all files from a defined class driver using the driver/friendly name of the device as a key. This is the mechanism for linking a printclass driver to a model specific driver.</td>
<td>Cannot be used by a class driver. When using RequiredClass, we recommend you avoid file name collisions between the two packages. While the files won't overwrite each other, it may be difficult to determine which is the file from the class driver package and which is from the v4 driver.</td>
<td>RequiredClass=&quot;Fabrikam PCL5e Class Driver&quot;</td>
</tr>
<tr>
<td>DriverFile</td>
<td>This points to the rendering binary. Mxdwdrv is the default, but class drivers may alternatively specify unidrv.dll or pscript5.dll. This is functionally identical to the same directive in a v3 INF.</td>
<td>Can only be set in a class driver. Valid choices are unidrv.dll or pscript5.dll. V4 print drivers either inherit from a RequiredClass or default to mxdwdrv.dll.</td>
<td>DriverFile=unidrv.dll</td>
</tr>
<tr>
<td>DataFile</td>
<td>This defines the primary GPD or PPD for this driver. This is functionally identical to the same directive in a v3 INF.</td>
<td>Required.</td>
<td>DataFile=FA-PDL.gpd</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Restrictions</td>
<td>Usage</td>
</tr>
<tr>
<td>--------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>--------------</td>
<td>--------------------------------------------</td>
</tr>
</tbody>
</table>
| Flags              | This will be used to specify additional, optional attributes associated with the driver.  
|                    | **NotShareable:** This flag specifies that the driver is not shareable. This is appropriate for virtual drivers such as the Microsoft XPS Document Writer.  
|                    | **SoftResetOnJobCancellation:** This flag specifies that the device requires a USB soft reset (IOCTL_USBPRINT_SOFT_RESET) on print job cancellation. | None.        | Flags=NotShareable, SoftResetOnJobCancellation |
| PrinterDriverID    | This is a unique ID that describes the print driver. If two drivers specify the same PrinterDriverID, then they must be compatible for sharing and support the same printer extensions. | Required.    | PrinterDriverID={guid}                     |
| PropertyBag        | Specifies a driver property bag for this driver. Note, this is a compiled file generated by DriverPropertyBagTool.exe or Visual Studio. | None.        | PropertyBag=FAProperty.dpb                  |
| ResourceFile       | Defines the name of the driver's string resource DLL.                         | None.        | ResourceFile=FARC.dll                       |
| ConstraintScript   | Defines the name of the driver's JavaScript constraint file.                  | None.        | ConstraintScript=FAConst.js                 |
### Table 13. Description of Directives Under BidiFiles Section of v4 Manifest

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Restrictions</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>BidiSPMFile</td>
<td>This defines the Bidi extension file for a TCP/IP-based printer.</td>
<td>None.</td>
<td>BidiSPMFile=FaBidiSPM.xml</td>
</tr>
</tbody>
</table>

#### 7.2.2 BidiFiles Section

This is the section name used to define Bidi extension files. It is identical to the Windows 7 format for TCP and WSD. The USB keywords are new.
7.2.3 DriverRender Section
This section specifies driver rendering options.

Table 14. Description of Directives Under DriverRender Section of v4 Manifest

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Restrictions</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>PageOutputQuality.[OptionName]</td>
<td>Changes the image compression based on the value in the job PrintTicket for PageOutputQuality</td>
<td>OptionName must be a name specified in the standard PrintSchema namespace.</td>
<td>PageOutputQuality.Draft=MxdclmageType.JPEGHigh PageOutputQuality.Normal=MxdclmageType.JPEGMedium PageOutputQuality.High=MxdclmageType.PNG</td>
</tr>
<tr>
<td>XpsFormat</td>
<td>Changes the XPS format generated by the print system for this driver. Multiple values may be specified, and the order represents the driver’s preference.</td>
<td>Not available for use in class drivers that use Unidrv/PScript rendering.</td>
<td>XpsFormat=XPS XpsFormat=OpenXPS XpsFormat=OpenXPS, XPS XpsFormat=XPS,OpenXPS</td>
</tr>
</tbody>
</table>

7.2.3.45 MxdclmageType Values

- MxdclmageType.JPEGHigh – High compression JPEG (smaller files)
- MxdclmageType.JPEGMedium – Medium compression JPEG
- MxdclmageType.JPEGLow – Low compression JPEG
- MxdclmageType.PNG – PNG file type (largest files)

7.2.4 FileSave Section
This section supports the file-save scenario. When a v4 print driver is installed against the new PORTPROMPT port type, this specifies the file extensions to be shown in the Common File Dialog, and the localizable resource strings that support the extensions and the dialog box itself.
Table 15. Description of Directives Under FileSave Section of v4 Manifest

<table>
<thead>
<tr>
<th>Directive</th>
<th>Restrictions</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;FileExtensionName&gt;</td>
<td>None</td>
<td>This directive describes the FileExtension to be used when saving a file from this driver using the PORTPROMPT port. The value is a resourceID from the driver's ResourceFile. For XPS and OXPS only, a resourceID of 0 may be specified and the print spooler will use its internal resources for these.</td>
<td>&lt;FileExtensionName&gt;=&lt;resourceID&gt; Xps=1234</td>
</tr>
<tr>
<td>SaveAsTitle</td>
<td>None</td>
<td>This directive describes the title to be used on the Save File dialog. The value is a resourceID from the driver's ResourceFile.</td>
<td>SaveAsTitle=&lt;resourceID&gt; SaveAsTitle=4321</td>
</tr>
</tbody>
</table>

7.2.5 PrinterExtensions Section

This section specifies a printer UI app and the invocation modes it supports.

Table 16. Description of Directives Under PrinterExtensions Section of v4 Manifest

<table>
<thead>
<tr>
<th>Directive</th>
<th>Restrictions</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>DriverEvent</td>
<td>None</td>
<td>App servicing the DriverEvent activation contract</td>
<td>DriverEvent=app.exe,{extensionID GUID}</td>
</tr>
<tr>
<td>PrintPreferences</td>
<td>None</td>
<td>App servicing the PrintPreferences activation contract</td>
<td>PrintPreferences=app.exe, {extensionID GUID}</td>
</tr>
</tbody>
</table>

7.2.6 Version 4 Manifest Sample – XPS

```
[DriverConfig]
DataFile=FAPDL.gpd
RequiredFiles=UNIRES.DLL,STDNAMES.GPD,STDDTYPE.GDL,STDSCHM.GDL,STDSCHMX.GDL,XPSSVCS.DLL,MSXPSINC.GPD
ResourceFile=FARC.dll
PropertyBag=FAPROPERTY.dpb
PrinterDriverID={GUID}
DriverCategory=PrintFax.Printer
ConstraintScript=faconst.js
EventFile=faevents.xml
PrinterExtensionUrl="http://www.fabrikam.com/download.asp?uiapp=120"
```
7.3 Packaging

V4 print drivers do not employ the needs and includes or core driver technologies in order to handle shared files. As a result, v4 print drivers must be self-contained, with only a few exceptions.

V4 print drivers may continue to take dependencies on common files that Windows provides. These include the files in NTPrint.INF or NTPrint4.INF. Drivers may include these files by specifying the RequiredFiles directive in the v4 manifest file.

If there are existing print class drivers that provide basic rendering functionality for your devices or your PDL, there also exists a mechanism to take a dependency on the class driver by using the RequiredClass directive. This directive causes Windows to build a driver using the files from both the v4 print driver and the required print class driver. GPD and PPD files are merged, with the most specific files taking precedence over less specific files. The following diagram illustrates the order used to merge the GPD/PPD files and also includes enhanced driver configuration files obtained from Bidi as specified in 8.3.1. Other driver files, such as JavaScript constraints, are not merged in the driver package.

![Diagram showing the logic for merging configuration data with enhanced driver configuration GPD/PPD]

**Figure 13. Logic for Merging Configuration Data with Enhanced Driver Configuration GPD/PPD**

7.4 CompatibleIDs

CompatibleIDs enable devices to advertise support for a less-specific print driver than a HardwareID does. CompatibleIDs are a key concept for print class drivers, because they enable the drivers to support devices that release after Windows without any updates to the driver.
If a particular CompatibleID is already supported by a print class driver, v4 print drivers should not specify it again. If the date of this driver is newer than the print class driver, it will be downloaded automatically from Windows Update.

Devices should include CompatibleIDs in their 1284ID string. If an existing print class driver supports the device, that CompatibleID should be used, otherwise we recommend that you use the following format. If CompatibleIDs are already implemented in existing devices, those should continue to be used.

- \[1284\text{\_CID\_.<manufacturer identifier>\_<PDL identifier>\_device family identifier} \]
- For example: 1284_CID_FA_PCL5e_Laser

CompatibleIDs are not used in the installation of TCP/IP based devices. As a result, users will need to identify an appropriate driver using only the name of the driver. Where print class drivers are concerned, we recommend that manufacturers provide compatibility lists on their websites for any devices that are supported by a print class driver.

More information about how to implement CompatibleIDs in your hardware, including a full list of rules and restrictions, is available at the Microsoft Download Center: http://download.microsoft.com/download/9/C/5/9C5B2167-8017-4BAE-9FDE-D5998AC8184A/CIDPrintDev.docx

### 7.4.1 Standard CompatibleIDs

Microsoft supports a few standard CompatibleIDs in order to support several manufacturer neutral print class drivers.

<table>
<thead>
<tr>
<th>Table 17. Standard CompatibleIDs and PDLs</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDL</td>
</tr>
<tr>
<td>XPS</td>
</tr>
<tr>
<td>OpenXPS (ECMA-388)</td>
</tr>
<tr>
<td>PCL6</td>
</tr>
<tr>
<td>PS</td>
</tr>
</tbody>
</table>

These print class drivers support only a very small set of features, so manufacturers that choose to utilize these should implement enhanced driver configuration using Bidi to add more specific paper sizes and configuration. Features supported by standard print class drivers:

<table>
<thead>
<tr>
<th>Table 18. Features Supported by Standard Print Class Drivers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature</td>
</tr>
<tr>
<td>Paper Size</td>
</tr>
<tr>
<td>Resolution</td>
</tr>
<tr>
<td>Feature</td>
</tr>
<tr>
<td>---------</td>
</tr>
<tr>
<td>Media Type</td>
</tr>
<tr>
<td>N-up</td>
</tr>
</tbody>
</table>

### 7.5 PrinterDriverID

The PrinterDriverID is a new identifier that is used to determine compatibility between drivers for printer sharing as well as between drivers and printer extensions. For example, if the driver on the server specifies a PrinterDriverID in its manifest file and the driver is then shared, clients that connect to this printer will search the local Driver Store and Windows Update for a driver that specifies the same PrinterDriverID in their driver INF. If this match occurs, a connection will be made using that driver. Client machines do not filter matching results using the driver name.

PrinterDriverID must be specified for all compatible drivers in the following ways:

1. Using the PrinterDriverID directive in the v4 manifest.
2. As a HardwareID in the v4 driver INF.

For two different drivers to share the same PrinterDriverID, they must be compatible for sharing. For the connection to always be successful, the two drivers must:

- Support the same PDL
- Use the same kind of configuration files (GPD or PPD)
- Be capable of rendering any features or options specified in the server driver’s GPD, PPD and/or constraint JS files
- Support the same printer extensions

The spooler does not validate these limitations and relies solely on the PrinterDriverID to indicate whether two drivers are compatible for sharing. Manufactures must be sure to change the PrinterDriverID for a driver if changes are made to any of the above items.

Printer extensions are also associated using PrinterDriverIDs. As a result, two drivers that share a PrinterDriverID must both work with the same printer extensions. The last printer extension installed will overwrite any prior printer extensions for all devices using the targeted PrinterDriverIDs, so they must work properly using the same app.

#### 7.5.1 Best Practices on GUIDs

GUIDs are used broadly through the v4 print driver model, most notably in the PrinterDriverID, but also in the PrinterExtensionID, the EventID, and the ModelID. These are used to uniquely identify different items in the system, or to identify them as the same for the sake of servicing, sharing, etc.

When creating new GUIDs, always use a GUID generator, such as the one included in Visual Studio or the one included in the SDK. Manually crafted GUIDs and GUIDs that have been erroneously copied and pasted are prone to collisions.
7.6 Print Queue Name
For v3 drivers, the print queue name was dictated first by the driver name, then by the user. With the introduction of print class drivers, the driver name is much less useful for user recognition of the device. Windows will rename the queue automatically for any Plug and Play devices installed against a v4 print driver as follows:
1. Initially, the print queue name is set to the driver name.
2. If the driver is a v4 print driver, Windows will query the device using Bidi.
   a. If \Printer.DeviceInfo:FriendlyName is specified, it will be used as the new queue name.
   b. Otherwise, Windows will query \Printer.DeviceInfo:Manufacturer, \Printer.DeviceInfo:ModelName
      i. If both are specified, Windows will concatenate them into "Manufacturer ModelName".
      ii. If only one of these Bidi queries fails, Windows will use the successful return from the other query as the queue name.
3. If all of the Bidi queries fail, then Windows will use the IEEE 1284ID to determine the manufacturer and model names.
   a. If DESCRIPTION or DES is specified, it will be used as the new queue name.
   b. Otherwise, Windows will search for MANUFACTURER or MFG and MODEL or MDL
      i. If both are specified, Windows will concatenate them into "MANUFACTURER MODEL".
      ii. If only one of these fails, Windows will use the value from the other key as the queue name.

7.6.1 Add Printer Wizard
The driver name will continue to be the only identifier available to users choosing a driver in the Add Printer Wizard. TCP/IP devices should implement the Port Monitor MIB (PWG 5107.1-2005) to support TCP/IP auto-detection. Existing devices that are added to a print class driver using HWID mapping may additionally use a device-specific model name.

7.7 Changing Ports and Dealing with Printer Devnodes
To provide a consistent UI experience, all print queues are given a software device node (devnode). This is how printers are discovered in the UI, and it allows virtual printers, connections to shared printers, and network printers to be enumerated and accessed in the same fashion as Plug and Play (PnP) printers. The software devnodes for physical PnP printers will inherit properties from the PnP devnode that triggered the creation of the queue.

The UI will group devnodes into Device Containers when two different objects are related. This is what allows MFPs to appear as one icon in the Devices and Printers folder. The Container ID for all of the functions in an MFP must be the same in order for the functions to all appear under the same icon. This is done automatically for PnP devices.

Changing the port associated with a queue will change the Container ID associated with the queue's devnode. This will cause the queue to no longer be grouped under the same device container as the rest of the PnP objects for the physical device. There is not enough information in the operating system to
properly clean up situations where the queue and the PnP object become separated. In some cases, that is the actual intent of the user. Only the user or application that changes the port name knows what the intended outcome is, and it is up to the user/application to clean up any confusing state left behind after a queue's port is changed. Here are two example situations, and how to clean up appropriately.

1. IT admin setting up printers – An IT admin uses WS Discovery to find a printer on the network and changes the port to TCP/IP because they like their TCP/IP management process.
   a. Expectation – There is only one "device" in the devices and printers folder.
   b. Solution – The IT admin removes the WSD PnP devnode from the devices and printers folder.

2. IHV setup software – The IHV installs a driver along with a custom port monitor (custom port monitors are not allowed in v4, but the same devnode handling applies to v3 drivers). The IHV changes the USB port of the print queue to a port the device manufacturer creates.
   a. Expectation – There is only one "device" in the devices and printers folder.
   b. Solution #1 – The PnP devnode is still needed: The setup program changes the container ID of the queue devnode to match the PnP object.
   c. Solution #2 – The PnP devnode is extraneous: The setup program removes the original PnP device.

7.8 Driver Ranking
The introduction of v4 print drivers does not modify the Plug and Play ranking behavior. When a device is plugged in, the available driver with the highest score will be selected. If the driver selected was a print class driver, and if there exists a better ranked, matching driver on Windows Update, it will be automatically replaced the next time the user downloads updates for Windows.


8 Connectivity

8.1 Port Monitors and Language Monitors
Third-party port monitors and language monitors are not supported in the v4 driver model or with print class drivers.

8.2 Bidirectional Schema
The Bidirectional Schema or Bidi continues to be supported with v4 print drivers. As with v3 print drivers, the TCPMon and WSDMon port monitors support definition of an extended Bidi schema using Bidi extension files. The format of these XML files is unchanged. Specific to v4 print drivers, the USBMon infrastructure has been extended to support Bidi communications and enable applications to receive rich status and configuration from a print device.

8.2.1 USB Bidi Extender
Windows enables manufacturers to support Bidi for USB devices by using a combination of both a Bidi XML file and a Javascript file known as a USB Bidi extender. This enables apps to use Bidi from the with USB as the transport.

The Javascript implementation does not support any device flow control or multiplexing of control information with print jobs during printing.

As shown in the following graphic, the USB Bidi extender supports three APIs for status.

![USB Bidi Extender Diagram]

**Figure 14. Architecture of USB Bidi Extender**

8.2.2 USB Bidi Extender API Reference

8.2.2.46 `getSchemas`
This function handles Bidi GET queries such as `\Printer.Consumables.YellowInk:Level`. The JavaScript code is able to make queries to the printer using the USB bus and read responses as they come back.

```javascript
function getSchemas(scriptContext, printerStream, schemaRequests, printerBidiSchemaResponses)
```

8.2.2.46.1 Parameters

- `scriptContext` – [in] The IPrinterScriptContext object that provides access to relevant property bags.
- `printerStream` – [in] The IPrinterScriptableSequentialStream object that allows read and write access to the USB bus.
- `printerBidiSchemaResponses` – [out] Object that the script uses to store all responses to query keys.
8.2.46.2 Return Values

1 – The attached device was not ready to provide some requested information. Indicates that the print system should call the function again using any Requery Keys added during processing.
0 – The script completed successfully.

8.2.47 setSchema
This function handles Bidi SET operations. The script can determine the incoming Bidi Schema value to either set data in the device, or perform some action on the device like clean ink heads.

If the device is not ready to process the specified data, the function can return a value of 1 to indicate the call should be retried after a wait period.

8.2.47.1 Parameters
scriptContext – [in] The IPrinterScriptContext object that provides access to relevant property bags.
printerStream – [in] The IPrinterScriptableSequentialStream object that allows read and write access to the USB bus.
printerBidiSchemaElement – [in] The IPrinterBidiSchemaElement that contains all the data associated with the Bidi Schema Value to set.

8.2.47.2 Return Values
1 – The attached device was not ready to provide some requested information. Indicates that the print system should call the function again using the supplied printerBidiSchemaElement.
0 – The script completed successfully.

8.2.48 getStatus
This function is used to obtain unsolicited status from a printer while the device is printing. This function is only called during printing. The device should provide data on the read channel which this script can interpret into Bidi Schema values. Since the write channel to the device is blocked by print data, only unsolicited status is supported here.

This function is called repeatedly during printing. It is expected that the device will only return data if it is available and the script can understand it. If the device does not support unsolicited status or there is no need to call this function again, the script should return a value of 2 which will tell the getStatus execution thread in USBMon to exit successfully.

8.2.48.1 Parameters
scriptContext – [in] The IPrinterScriptContext object that provides access to relevant property bags.
printerStream – [in] The IPrinterScriptableSequentialStream object that allows read access to the USB bus.
printerBidiSchemaResponses – [out] Object that the script uses to store all responses to query keys.

8.2.48.2 Return Values
2 – The attached device no longer supports unsolicited status and this function should not be called again.
0 – The script completed successfully.
8.2.49 requestStatus
This function is used to obtain status from a printer while the device is printing if the device supports a secondary USB interface. This function will be called instead of getStatus if the BidiUSBStatusInterface directive has been specified in the v4 driver’s manifest file.

This function is called repeatedly during printing. It is expected that the device will only return data if it is available and the script can understand it. If the device does not support unsolicited status or there is no need to call this function again, the script should return a value of 2 which will tell the getStatus execution thread in USBMon to exit successfully.

8.2.49.1 Parameters
scriptContext – [in] The IPrinterScriptContext object that provides access to relevant property bags.
printerStream – [in] The IPrinterScriptableSequentialStream object that allows read and write access to the USB bus.
printerBidiSchemaResponses – [out] Object that the script uses to store all responses to query keys.

8.2.49.2 Return Values
2 – The attached device no longer supports unsolicited status and this function should not be called again.
0 – The script completed successfully.

8.2.3 USBMon and USB Bidi Extension File Interaction
As each new USB port is created or opened, USBMon will determine if the attached device/driver contains a new Bidi Extension file and a Bidi Extension JavaScript file. USBMon searches the v4 driver manifest or the driver INI file and retrieves the name of the files. If present, USBMon will use these files to determine the list of extended Bidi Schema values supported by this device and then communicate with the device to query their values. At this point USBMon supports the IHV-specified Bidi Schema actions through existing Print Spooler APIs.

8.2.4 Windows Driver Kit Samples

8.2.4.50 USBMon Bidi XML File Sample
The Windows Driver Kit provides a sample of a USBMon Bidi Extension XML file on the Windows Sample Gallery. It uses the standard Bidi schema properties DeviceInfo, Configuration, and Memory and also defines a few custom extensions.

Additional information related to creating Bidi extension files is available on MSDN: http://msdn.microsoft.com/en-us/library/dd183364(VS.85).aspx

8.2.4.51 USBMon Bidi JavaScript File Sample
The Windows Driver Kit includes a USBMon Bidi Extender JavaScript file. This is available on the Windows Sample Gallery. It demonstrates how to support Bidi’s SET and GET operations, as well as how to listen for events while the printer is printing.

8.2.4.51.1 Debugging
Interactive debugging can be enabled by creating the registry key specified below. For USB Bidi JavaScript, the print spooler must be restarted before debugging will be enabled.
Key Name: HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Print
Value Name: EnableJavaScriptDebugging
Type: DWORD
Value: 1

After the key above is created, and the hosting process has been restarted, the script can be debugged as follows:
   1. Attach your debugger to the hosting process
      a. For USB Bidi JavaScript, this is spoolsv.exe
   2. Set the debugger to script debugging mode
   3. Select “Break All” (Ctrl + Alt + Break) to break into the process the next time a script runs.
   4. Run the scenario to reproduce your issue
   5. Once the debugger breaks into a JavaScript function, set any breakpoints necessary and step through the code.

8.3 Autoconfiguration
Autoconfiguration of device installable options is supported in the v4 driver model using GDL and Bidi. Devices that do not support Bidi may also continue to configure installable option state using the printer properties UI.

Note that drivers using autoconfiguration must include the following files in the driver package using the RequiredFiles directive.

Table 19. Files Required For Autoconfiguration

<table>
<thead>
<tr>
<th>GPD</th>
<th>PPD</th>
</tr>
</thead>
<tbody>
<tr>
<td>• stdschmx.gdl</td>
<td>• PS_schm.gdl</td>
</tr>
<tr>
<td>• stdscheme.gdl</td>
<td></td>
</tr>
<tr>
<td>• stdtype.gdl</td>
<td></td>
</tr>
</tbody>
</table>

8.3.1 Enhanced Driver Configuration with Bidi
The v4 driver model supports the retrieval of GPD and PPD files from the device using Bidi. This allows devices using a print class driver to support a richer feature set without requiring additional downloads from Windows Update.

This functionality is supported by default for drivers that support WS-Print v1.1, however, TCP/IP devices, and WS-Print v1.0 devices may also support it by implementing the Bidi extension files that specify the following Bidi schema elements.
8.3.1.52 Schema Path: Schema Section for Reading the GPD/PPD Files
Section Name: DriverConfigFiles
Schema Path: \Printer.Configuration.DriverConfigFiles
Description: This new section for the Bidi Schema will contain schema values to query the device for driver configuration data, including the GPD and PPD description files.

8.3.1.52.1 Extension for Reading the GPD File
Schema Name: GPDFile
Schema Path: \Printer.Configuration.DriverConfigFiles:GPDFile
Node Type: Value
Data Type: BIDI_STRING
Description: The full GPD file for the device. This contains all the specific device configuration information that is available and up to date according to the current settings of the device.

8.3.1.52.2 Extension for Reading the PPD File
Schema Name: PPDFile
Schema Path: \Printer.Configuration.DriverConfigFiles:PPDFile
Node Type: Value
Data Type: BIDI_STRING
Description: The full PPD file for the device. This contains all the specific device configuration information that is available and up to date according to the current settings of the device.

8.4 WS-Print v1.1
The WSD Printing Schema, WS-Print, has been updated in Windows 8 to support enhanced driver configuration, better color representation for ink/toner, and device model IDs. A formal revision of the specification is forthcoming.

8.4.1 Sequence Diagram
The following sequence diagram illustrates the interaction between the client and printer in order to determine whether the WS-Print v1.1 namespace is supported, and then to retrieve the extended schema elements.

![Sequence Diagram of WS-Print v1.1](image-url)
8.4.2 Namespace

**Namespace:** http://schemas.microsoft.com/windows/2010/06/wdp/printv11

**XML Namespace Definition:**


8.4.3 Specifying WS-Print 1.1 Support

Printers that wish to support WS-Print 1.1 elements must update their PrinterDescription to include wprr11:SupportsWSPrintv11. If wprr11:SupportsWSPrintv11 is not specified and set to true, then WSDMon will not request any WS-Print 1.1 elements.

8.4.3.53 Sample

This sample is derived from the WSD Print Service Specification v1.0. Devices that support WS-Print 1.1 must include the highlighted content in their PrinterDescription order for Windows to query for any other elements in that namespace.

```xml
<soap:Envelope
   xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
   xmlns:wprr="http://schemas.microsoft.com/windows/2006/08/wdp/print"
  <soap:Header>
    <wsa:To>http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous</wsa:To>
    <wsa:Action>
    </wsa:Action>
    <wsa:MessageID>uuid:UniqueMsgId</wsa:MessageID>
    <wsa:RelatesTo>uuid:MsgIdOfTheGetPrinterElementsRequest</wsa:RelatesTo>
  </soap:Header>
  <soap:Body>
    <wprr:GetPrinterElementsResponse>
      <wprr:PrinterElements>
        <wprr:ElementData Name="wprr:PrinterDescription" Valid="true">
          <wprr:PrinterDescription>
            <wprr:ColorSupported>true</wprr:ColorSupported>
            <wprr:DeviceId>MFG:Acme;MDL:PrintMaster 9020</wprr:DeviceId>
            <wprr:MultipleDocumentJobsSupported>true</wprr:MultipleDocumentJobsSupported>
            <wprr:PagesPerMinute>20</wprr:PagesPerMinute>
            <wprr:PagesPerMinuteColor>8</wprr:PagesPerMinuteColor>
            <wprr:PrinterName xml:lang="en-AU, en-CA, en-GB, en-US">
              Accounting Printer in Copy Room 2
            </wprr:PrinterName>
          </wprr:PrinterDescription>
        </wprr:ElementData>
      </wprr:PrinterElements>
    </wprr:GetPrinterElementsResponse>
  </soap:Body>
</soap:Envelope>
```
8.4.3.54 Schema

```xml
            xmlns:xs="http://www.w3.org/2001/XMLSchema"
            elementFormDefault="qualified">

  <xs:annotation>
    <xs:documentation>
      WS-Print Extensions for Driver Configuration and Consumable Definition
      Copyright 2010 Microsoft Corp. All rights reserved
    </xs:documentation>
  </xs:annotation>

  <xs:annotation>
    <xs:documentation>A Boolean element that denotes support for WS-Print V11 extensions</xs:documentation>
  </xs:annotation>

  <xs:element name="SupportsWSPrintv11" type="xs:boolean"/>
</xs:schema>
```

8.4.4 Enhanced Driver Configuration

This schema provides device-specific GPD or PPD configuration files for this device.
8.4.5 Device Model ID
This schema describes a ModelID for the device, and will be used for device metadata retrieval. For

8.4.6 Ink/Toner Color Representation Value
This schema gets an RGB triple that represents the color for a particular ink or toner type. This value
should be specified for any ink or toner consumables to enable a better representation of the color to be
shown in app UI.
9 Printer Sharing Changes

Complementing the v4 driver model are changes to the printer sharing model. The updated printer sharing mechanism is referred to as enhanced Point and Print, and it allows print clients to print to v4 shares without downloading the manufacturer-provided device driver from the print server.

9.1 Windows 8 Client Connection Behavior

When a Windows 8 client connects to a shared print queue that is using a v4 print driver, it will try to obtain a driver that supports client side rendering. The client searches the local DriverStore for a driver with a HardwareID that matches the server driver’s PrinterDriverID. If one is found, that driver will be installed locally. Otherwise, the client will connect using the enhanced Point and Print driver.

In both cases, the client downloads configuration data from the server using GetPrinterDataEx calls. The configuration data includes data files like GPDS, PPDs, the driver property bag, JavaScript constraints and a resource DLL. The client also downloads the CAT file that was associated with the server’s driver. On the client, the print system then validates that the resource DLL contains no executable code, and that the downloaded files are validly signed by the CAT file downloaded from the server. If any files are untrusted, they will be deleted.
9.2 Windows 7 and Windows Vista Client Connection Behavior

Windows 7 and Windows Vista clients may also connect to shared print queues that use a v4 print driver. In this case, however, the client will always download the enhanced Point and Print Driver from the server. This driver uses server side rendering to ensure the proper PDL is generated for the printer.

Configuration data is downloaded from the server in the same way for Windows 7 or Windows Vista client connections. If any downloaded files fail validation against the server’s CAT file, they will be deleted.
Shared printers that are backed by a v3 print driver will continue to work using the existing Point and Print system.

10 Security

Security is an important consideration in the design of v4 print drivers. In addition to the usual threats such as elevation of privilege, spoofed devices, or man in the middle attacks, v4 print drivers need to also be compatible with low-rights applications like Internet Explorer 9.

XPS rendering filters and JavaScript files must all be hardened against all forms of untrusted data from applications, users, or across machine boundaries. Malformed PrintTickets, XPS documents, property bags, and even BidiResponses must be validated and parsed carefully and should never be used to store executable code. We recommend that partners use extensive fuzzed file testing to ensure graceful failure without compromising security integrity.
11 Appendix A: Localized Keywords

Below is a list of the keywords that will be localized using standard values. The final localized values are subject to change.

**Table 20. Localized Keywords**

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Option Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JobInputBin</td>
<td></td>
<td>Input Bin</td>
</tr>
<tr>
<td>JobInputBin</td>
<td>AutoSelect</td>
<td>Auto Select</td>
</tr>
<tr>
<td>JobInputBin</td>
<td>Manual</td>
<td>Manual</td>
</tr>
<tr>
<td>JobInputBin</td>
<td>Cassette</td>
<td>Removable Cassette</td>
</tr>
<tr>
<td>JobInputBin</td>
<td>Tractor</td>
<td>Tractor Feed</td>
</tr>
<tr>
<td>JobInputBin</td>
<td>AutoSheetFeeder</td>
<td>Automatic Sheet Feeder</td>
</tr>
<tr>
<td>DocumentInputBin</td>
<td></td>
<td>Input Bin</td>
</tr>
<tr>
<td>DocumentInputBin</td>
<td>AutoSelect</td>
<td>Auto Select</td>
</tr>
<tr>
<td>DocumentInputBin</td>
<td>Cassette</td>
<td>Removable Cassette</td>
</tr>
<tr>
<td>DocumentInputBin</td>
<td>AutoSheetFeeder</td>
<td>Tractor Feed</td>
</tr>
<tr>
<td>PageInputBin</td>
<td></td>
<td>Automatic Sheet Feeder</td>
</tr>
<tr>
<td>PageInputBin</td>
<td>AutoSelect</td>
<td>Input Bin</td>
</tr>
<tr>
<td>PageInputBin</td>
<td>Manual</td>
<td>Auto Select</td>
</tr>
<tr>
<td>PageInputBin</td>
<td>Cassette</td>
<td>Manual</td>
</tr>
<tr>
<td>PageInputBin</td>
<td>AutoSheetFeeder</td>
<td>Removable Cassette</td>
</tr>
<tr>
<td>MediaSheetCapacity</td>
<td></td>
<td>Bin Capacity</td>
</tr>
<tr>
<td>JobOutputBin</td>
<td></td>
<td>Output Bin</td>
</tr>
<tr>
<td>DocumentOutputBin</td>
<td></td>
<td>Output Bin</td>
</tr>
<tr>
<td>PageOutputBin</td>
<td></td>
<td>Output Bin</td>
</tr>
<tr>
<td>JobCopiesAllDocuments</td>
<td></td>
<td>Copies</td>
</tr>
<tr>
<td>DocumentCopiesAllPages</td>
<td></td>
<td>Copies</td>
</tr>
<tr>
<td>DocumentPageRanges</td>
<td></td>
<td>Page Range</td>
</tr>
<tr>
<td>PageCopies</td>
<td></td>
<td>Copies</td>
</tr>
<tr>
<td>PageMediaColor</td>
<td></td>
<td>Media Color</td>
</tr>
<tr>
<td>PageMediaColor</td>
<td>Black</td>
<td>Black</td>
</tr>
<tr>
<td>PageMediaColor</td>
<td>Blue</td>
<td>Blue</td>
</tr>
<tr>
<td>PageMediaColor</td>
<td>Brown</td>
<td>Brown</td>
</tr>
<tr>
<td>PageMediaColor</td>
<td>Gold</td>
<td>Gold</td>
</tr>
<tr>
<td>PageMediaColor</td>
<td>GoldenRod</td>
<td>Goldenrod</td>
</tr>
<tr>
<td>PageMediaColor</td>
<td>Gray</td>
<td>Gray</td>
</tr>
<tr>
<td>PageMediaColor</td>
<td>Green</td>
<td>Green</td>
</tr>
<tr>
<td>PageMediaColor</td>
<td>Ivory</td>
<td>Ivory</td>
</tr>
<tr>
<td>Feature Name</td>
<td>Option Name</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>------------------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>PageMediaColor</td>
<td>NoColor</td>
<td>No Color</td>
</tr>
<tr>
<td>PageMediaColor</td>
<td>Orange</td>
<td>Orange</td>
</tr>
<tr>
<td>PageMediaColor</td>
<td>Pink</td>
<td>Pink</td>
</tr>
<tr>
<td>PageMediaColor</td>
<td>Red</td>
<td>Red</td>
</tr>
<tr>
<td>PageMediaColor</td>
<td>Silver</td>
<td>Silver</td>
</tr>
<tr>
<td>PageMediaColor</td>
<td>Turquoise</td>
<td>Turquoise</td>
</tr>
<tr>
<td>PageMediaColor</td>
<td>Violet</td>
<td>Violet</td>
</tr>
<tr>
<td>PageMediaColor</td>
<td>White</td>
<td>White</td>
</tr>
<tr>
<td>PageMediaColor</td>
<td>Yellow</td>
<td>Yellow</td>
</tr>
<tr>
<td>PageMediaColor</td>
<td>Custom</td>
<td>Custom Color</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>Media Size</td>
<td>Custom Media Size</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>CustomMediaSize</td>
<td>Custom Media Size</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>PSCustomMediaSize</td>
<td>Custom Media Size</td>
</tr>
<tr>
<td>PageMediaType</td>
<td>Media Type</td>
<td>Media Type</td>
</tr>
<tr>
<td>PageMediaType</td>
<td>AutoSelect</td>
<td>Auto Select</td>
</tr>
<tr>
<td>PageMediaType</td>
<td>Archival</td>
<td>Archival</td>
</tr>
<tr>
<td>PageMediaType</td>
<td>BackPrintFilm</td>
<td>Back printed film</td>
</tr>
<tr>
<td>PageMediaType</td>
<td>Bond</td>
<td>Bond</td>
</tr>
<tr>
<td>PageMediaType</td>
<td>CardStock</td>
<td>Card stock</td>
</tr>
<tr>
<td>PageMediaType</td>
<td>ContinuousFeed</td>
<td>Continuous feed</td>
</tr>
<tr>
<td>PageMediaType</td>
<td>EnvelopePlain</td>
<td>Envelope (Plain)</td>
</tr>
<tr>
<td>PageMediaType</td>
<td>EnvelopeWindow</td>
<td>Envelope (Window)</td>
</tr>
<tr>
<td>PageMediaType</td>
<td>Fabric</td>
<td>Fabric</td>
</tr>
<tr>
<td>PageMediaType</td>
<td>HighResolution</td>
<td>High Resolution Paper</td>
</tr>
<tr>
<td>PageMediaType</td>
<td>Label</td>
<td>Label</td>
</tr>
<tr>
<td>PageMediaType</td>
<td>MultiLayerForm</td>
<td>Multi Layer Form</td>
</tr>
<tr>
<td>PageMediaType</td>
<td>Photographic</td>
<td>Photo Paper</td>
</tr>
<tr>
<td>PageMediaType</td>
<td>PhotographicFilm</td>
<td>Photo Film</td>
</tr>
<tr>
<td>PageMediaType</td>
<td>PhotographicGlossy</td>
<td>Photo Paper (Glossy)</td>
</tr>
<tr>
<td>PageMediaType</td>
<td>PhotographicHighGloss</td>
<td>Photo Paper (High Gloss)</td>
</tr>
<tr>
<td>PageMediaType</td>
<td>PhotographicMatte</td>
<td>Photo Paper (Matte)</td>
</tr>
<tr>
<td>PageMediaType</td>
<td>PhotographicSatin</td>
<td>Photo Paper (Satin)</td>
</tr>
<tr>
<td>PageMediaType</td>
<td>PhotographicSemiGloss</td>
<td>Photo Paper (Semi-gloss)</td>
</tr>
<tr>
<td>PageMediaType</td>
<td>Plain</td>
<td>Plain Paper</td>
</tr>
<tr>
<td>PageMediaType</td>
<td>Stationery</td>
<td>Stationery</td>
</tr>
<tr>
<td>PageMediaType</td>
<td>TabStockFull</td>
<td>Tab Stock</td>
</tr>
<tr>
<td>PageMediaType</td>
<td>TabStockPreCut</td>
<td>Tab Stock (Pre-cut)</td>
</tr>
<tr>
<td>PageMediaType</td>
<td>Transparency</td>
<td>Transparency</td>
</tr>
<tr>
<td>PageMediaType</td>
<td>TShirtTransfer</td>
<td>T-Shirt Transfer Media</td>
</tr>
<tr>
<td>PageMediaType</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>Feature Name</td>
<td>Option Name</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------------------------</td>
<td>---------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>JobDuplexAllDocumentsContiguously</td>
<td>OneSided</td>
<td>Duplex Printing</td>
</tr>
<tr>
<td>JobDuplexAllDocumentsContiguously</td>
<td>TwoSidedShortEdge</td>
<td>Print on Two Sides (flip on short edge)</td>
</tr>
<tr>
<td>JobDuplexAllDocumentsContiguously</td>
<td>TwoSidedLongEdge</td>
<td>Print on Two Sides (flip on long edge)</td>
</tr>
<tr>
<td>DocumentDuplex</td>
<td>OneSided</td>
<td>Duplex Printing</td>
</tr>
<tr>
<td>DocumentDuplex</td>
<td>TwoSidedShortEdge</td>
<td>Print on Two Sides (flip on short edge)</td>
</tr>
<tr>
<td>DocumentDuplex</td>
<td>TwoSidedLongEdge</td>
<td>Print on Two Sides (flip on long edge)</td>
</tr>
<tr>
<td>JobCollateAllDocuments</td>
<td>Collated</td>
<td>Collation</td>
</tr>
<tr>
<td>JobCollateAllDocuments</td>
<td>Uncollated</td>
<td>Uncollated</td>
</tr>
<tr>
<td>DocumentCollate</td>
<td>Collated</td>
<td>Collation</td>
</tr>
<tr>
<td>DocumentCollate</td>
<td>Uncollated</td>
<td>Uncollated</td>
</tr>
<tr>
<td>PageOutputColor</td>
<td>Color</td>
<td>Color Mode</td>
</tr>
<tr>
<td>PageOutputColor</td>
<td>Grayscale</td>
<td>Grayscale</td>
</tr>
<tr>
<td>PageOutputColor</td>
<td>Monochrome</td>
<td>Monochrome</td>
</tr>
<tr>
<td>PageOrientation</td>
<td>Landscape</td>
<td>Landscape</td>
</tr>
<tr>
<td>PageOrientation</td>
<td>Portrait</td>
<td>Portrait</td>
</tr>
<tr>
<td>PageOrientation</td>
<td>ReverseLandscape</td>
<td>Landscape (Reversed)</td>
</tr>
<tr>
<td>PageOrientation</td>
<td>ReversePortrait</td>
<td>Portrait (Reversed)</td>
</tr>
<tr>
<td>JobNUpAllDocumentsContiguously</td>
<td>PagesPerSheet</td>
<td>Multiple Pages Per Sheet</td>
</tr>
<tr>
<td>DocumentNUp</td>
<td>PagesPerSheet</td>
<td>Pages Per Sheet</td>
</tr>
<tr>
<td>JobHolePunch</td>
<td>Hole Punching</td>
<td>Hole Punching</td>
</tr>
<tr>
<td>JobHolePunch</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>JobHolePunch</td>
<td>LeftEdge</td>
<td>Left Edge</td>
</tr>
<tr>
<td>JobHolePunch</td>
<td>RightEdge</td>
<td>Right Edge</td>
</tr>
<tr>
<td>JobHolePunch</td>
<td>TopEdge</td>
<td>Top Edge</td>
</tr>
<tr>
<td>JobHolePunch</td>
<td>BottomEdge</td>
<td>Bottom Edge</td>
</tr>
<tr>
<td>DocumentHolePunch</td>
<td>Hole Punching</td>
<td>Hole Punching</td>
</tr>
<tr>
<td>DocumentHolePunch</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>DocumentHolePunch</td>
<td>LeftEdge</td>
<td>Left Edge</td>
</tr>
<tr>
<td>DocumentHolePunch</td>
<td>RightEdge</td>
<td>Right Edge</td>
</tr>
<tr>
<td>DocumentHolePunch</td>
<td>TopEdge</td>
<td>Top Edge</td>
</tr>
<tr>
<td>Feature Name</td>
<td>Option Name</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>----------------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>DocumentHolePunch</td>
<td>BottomEdge</td>
<td>Bottom Edge</td>
</tr>
<tr>
<td>JobStapleAllDocuments</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>JobStapleAllDocuments</td>
<td>StapleTopLeft</td>
<td>Top Left</td>
</tr>
<tr>
<td>JobStapleAllDocuments</td>
<td>StapleTopRight</td>
<td>Top Right</td>
</tr>
<tr>
<td>JobStapleAllDocuments</td>
<td>StapleBottomLeft</td>
<td>Bottom Left</td>
</tr>
<tr>
<td>JobStapleAllDocuments</td>
<td>StapleBottomRight</td>
<td>Bottom Right</td>
</tr>
<tr>
<td>JobStapleAllDocuments</td>
<td>StapleDualLeft</td>
<td>Left Edge (2 staples)</td>
</tr>
<tr>
<td>JobStapleAllDocuments</td>
<td>StapleDualRight</td>
<td>Right Edge (2 staples)</td>
</tr>
<tr>
<td>JobStapleAllDocuments</td>
<td>StapleDualTop</td>
<td>Top Edge (2 staples)</td>
</tr>
<tr>
<td>JobStapleAllDocuments</td>
<td>StapleDualBottom</td>
<td>Bottom Edge (2 staples)</td>
</tr>
<tr>
<td>JobStapleAllDocuments</td>
<td>SaddleStitch</td>
<td>Saddle Stitching</td>
</tr>
<tr>
<td>JobStapleAllDocuments</td>
<td>SheetCapacity</td>
<td>Sheet Capacity</td>
</tr>
<tr>
<td>JobStapleAllDocuments</td>
<td>Angle</td>
<td>Staple Angle</td>
</tr>
<tr>
<td>DocumentStaple</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>DocumentStaple</td>
<td>StapleTopLeft</td>
<td>Top Left</td>
</tr>
<tr>
<td>DocumentStaple</td>
<td>StapleTopRight</td>
<td>Top Right</td>
</tr>
<tr>
<td>DocumentStaple</td>
<td>StapleBottomLeft</td>
<td>Bottom Left</td>
</tr>
<tr>
<td>DocumentStaple</td>
<td>StapleBottomRight</td>
<td>Bottom Right</td>
</tr>
<tr>
<td>DocumentStaple</td>
<td>StapleDualLeft</td>
<td>Left Edge (2 staples)</td>
</tr>
<tr>
<td>DocumentStaple</td>
<td>StapleDualRight</td>
<td>Right Edge (2 staples)</td>
</tr>
<tr>
<td>DocumentStaple</td>
<td>StapleDualTop</td>
<td>Top Edge (2 staples)</td>
</tr>
<tr>
<td>DocumentStaple</td>
<td>StapleDualBottom</td>
<td>Bottom Edge (2 staples)</td>
</tr>
<tr>
<td>DocumentStaple</td>
<td>SaddleStitch</td>
<td>Saddle Stitching</td>
</tr>
<tr>
<td>DocumentStaple</td>
<td>SheetCapacity</td>
<td>Sheet Capacity</td>
</tr>
<tr>
<td>DocumentStaple</td>
<td>Angle</td>
<td>Staple Angle</td>
</tr>
<tr>
<td>JobBindAllDocuments</td>
<td>Bale</td>
<td>Bale Binding</td>
</tr>
<tr>
<td>JobBindAllDocuments</td>
<td>BindBottom</td>
<td>Bottom Edge</td>
</tr>
<tr>
<td>JobBindAllDocuments</td>
<td>BindLeft</td>
<td>Left Edge</td>
</tr>
<tr>
<td>JobBindAllDocuments</td>
<td>BindRight</td>
<td>Right Edge</td>
</tr>
<tr>
<td>JobBindAllDocuments</td>
<td>BindTop</td>
<td>Top Edge</td>
</tr>
<tr>
<td>JobBindAllDocuments</td>
<td>Booklet</td>
<td>Booklet</td>
</tr>
<tr>
<td>JobBindAllDocuments</td>
<td>EdgeStitchBottom</td>
<td>Bottom Edge (Stitching)</td>
</tr>
<tr>
<td>JobBindAllDocuments</td>
<td>EdgeStitchLeft</td>
<td>Left Edge (Stitching)</td>
</tr>
<tr>
<td>JobBindAllDocuments</td>
<td>EdgeStitchRight</td>
<td>Right Edge (Stitching)</td>
</tr>
<tr>
<td>JobBindAllDocuments</td>
<td>EdgeStitchTop</td>
<td>Top Edge (Stitching)</td>
</tr>
<tr>
<td>JobBindAllDocuments</td>
<td>Fold</td>
<td>Fold</td>
</tr>
<tr>
<td>JobBindAllDocuments</td>
<td>JogOffset</td>
<td>Offset</td>
</tr>
<tr>
<td>Feature Name</td>
<td>Option Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>JobBindAllDocuments</td>
<td>Trim</td>
<td>Trim Edge</td>
</tr>
<tr>
<td>JobBindAllDocuments</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>DocumentBinding</td>
<td></td>
<td>Document Binding</td>
</tr>
<tr>
<td>DocumentBinding</td>
<td>Bale</td>
<td>Bale Binding</td>
</tr>
<tr>
<td>DocumentBinding</td>
<td>BindBottom</td>
<td>Bottom Edge</td>
</tr>
<tr>
<td>DocumentBinding</td>
<td>BindLeft</td>
<td>Left Edge</td>
</tr>
<tr>
<td>DocumentBinding</td>
<td>BindRight</td>
<td>Right Edge</td>
</tr>
<tr>
<td>DocumentBinding</td>
<td>BindTop</td>
<td>Top Edge</td>
</tr>
<tr>
<td>DocumentBinding</td>
<td>Booklet</td>
<td>Booklet</td>
</tr>
<tr>
<td>DocumentBinding</td>
<td>EdgeStitchBottom</td>
<td>Bottom Edge (Stitching)</td>
</tr>
<tr>
<td>DocumentBinding</td>
<td>EdgeStitchLeft</td>
<td>Left Edge (Stitching)</td>
</tr>
<tr>
<td>DocumentBinding</td>
<td>EdgeStitchRight</td>
<td>Right Edge (Stitching)</td>
</tr>
<tr>
<td>DocumentBinding</td>
<td>EdgeStitchTop</td>
<td>Top Edge (Stitching)</td>
</tr>
<tr>
<td>DocumentBinding</td>
<td>Fold</td>
<td>Fold</td>
</tr>
<tr>
<td>DocumentBinding</td>
<td>JogOffset</td>
<td>Offset</td>
</tr>
<tr>
<td>DocumentBinding</td>
<td>Trim</td>
<td>Trim Edge</td>
</tr>
<tr>
<td>DocumentBinding</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>BusinessCard</td>
<td>Business Card</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>CreditCard</td>
<td></td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>ISOA0</td>
<td>A0</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>ISOA1</td>
<td>A1</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>ISOA2</td>
<td>A2</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>ISOA3</td>
<td>A3</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>ISOA4</td>
<td>A4</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>ISOA5</td>
<td>A5</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>ISOA6</td>
<td>A6</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>ISOA7</td>
<td>A7</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>ISOA8</td>
<td>A8</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>ISOA9</td>
<td>A9</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>ISOA10</td>
<td>A10</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>ISOA3Extra</td>
<td>A3 (Extra)</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>ISOA3Rotated</td>
<td>A3 (Rotated)</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>ISOA4Extra</td>
<td>A4 (Extra)</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>ISOA5Extra</td>
<td>A5 (Extra)</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>ISOA4Rotated</td>
<td>A4 (Rotated)</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>ISOA5Rotated</td>
<td>A5 (Rotated)</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>ISOA6Rotated</td>
<td>A6 (Rotated)</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>ISOB0</td>
<td>B0</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>ISOB1</td>
<td>B1</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>ISOB2</td>
<td>B2</td>
</tr>
<tr>
<td>Feature Name</td>
<td>Option Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>ISOB3</td>
<td>B3</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>ISOB4</td>
<td>B4</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>ISOB5</td>
<td>B5</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>ISOB6</td>
<td>B6</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>ISOB7</td>
<td>B7</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>ISOB8</td>
<td>B8</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>ISOB9</td>
<td>B9</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>ISOB10</td>
<td>B10</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>ISOB4Envelope</td>
<td>B4 (Envelope)</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>ISOB5Envelope</td>
<td>B5 (Envelope)</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>ISOB5Extra</td>
<td>B5 (Extra)</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>ISOC0</td>
<td>C0</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>ISOC1</td>
<td>C1</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>ISOC2</td>
<td>C2</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>ISOC3</td>
<td>C3</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>ISOC4</td>
<td>C4</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>ISOC5</td>
<td>C5</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>ISOC6</td>
<td>C6</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>ISOC7</td>
<td>C7</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>ISOC8</td>
<td>C8</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>ISOC9</td>
<td>C9</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>ISOC10</td>
<td>C10</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>ISOC3Envelope</td>
<td>C3 (Envelope)</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>ISOC4Envelope</td>
<td>C4 (Envelope)</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>ISOC5Envelope</td>
<td>C5 (Envelope)</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>ISOC6C5Envelope</td>
<td>C6/C5 (Envelope)</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>ISO6Envelope</td>
<td>C6 (Envelope)</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>ISODLEEnvelope</td>
<td>DL (Envelope)</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>ISODLEEnvelopeRotated</td>
<td>DL (Envelope, Rotated)</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>IOSRA3</td>
<td>SRA3</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>Japan2LPhoto</td>
<td>Japan 2L Photo</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>JapanChou3Envelope</td>
<td>Japan Chou 3 Envelope</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>JapanChou3EnvelopeRotated</td>
<td>Japan Chou 3 Envelope Rotated</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>JapanChou4Envelope</td>
<td>Japan Chou 4 Envelope</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>JapanChou4EnvelopeRotated</td>
<td>Japan Chou 4 Envelope Rotated</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>JapanDoubleHagakiPostcard</td>
<td>Japan Double Hagaki Postcard</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>JapanDoubleHagakiPostcardRotated</td>
<td>Japan Double Hagaki Postcard Rotated</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>JapanHagakiPostcard</td>
<td>Japan Hagaki Postcard</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>JapanHagakiPostcardRotated</td>
<td>Japan Hagaki Postcard Rotated</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>JapanKaku2Envelope</td>
<td>Japan Kaku 2 Envelope</td>
</tr>
<tr>
<td>Feature Name</td>
<td>Option Name</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------------------------------------------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>JapanKaku2EnvelopeRotated</td>
<td>Japan Kaku 2 Envelope Rotated</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>JapanKaku3Envelope</td>
<td>Japan Kaku 3 Envelope</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>JapanKaku3EnvelopeRotated</td>
<td>Japan Kaku 3 Envelope Rotated</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>JapanLPhoto</td>
<td>Japan L Photo</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>JapanQuadrupleHagakiPostcard</td>
<td>Japan Quadruple Hagaki Postcard</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>JapanYou1Envelope</td>
<td>Japan You 1 Envelope</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>JapanYou2Envelope</td>
<td>Japan You 2 Envelope</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>JapanYou3Envelope</td>
<td>Japan You 3 Envelope</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>JapanYou4Envelope</td>
<td>Japan You 4 Envelope</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>JapanYou4EnvelopeRotated</td>
<td>Japan You 4 Envelope Rotated</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>JapanYou6Envelope</td>
<td>Japan You 6 Envelope</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>JapanYou6EnvelopeRotated</td>
<td>Japan You 6 Envelope Rotated</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>JIS80</td>
<td>JIS B0</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>JIS81</td>
<td>JIS B1</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>JIS810</td>
<td>JIS B10</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>JIS82</td>
<td>JIS B2</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>JIS83</td>
<td>JIS B3</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>JIS84</td>
<td>JIS B4</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>JIS84Rotated</td>
<td>JIS B4 Rotated</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>JIS85</td>
<td>JIS B5</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>JIS85Rotated</td>
<td>JIS B5 Rotated</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>JIS86</td>
<td>JIS B6</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>JIS86Rotated</td>
<td>JIS B6 Rotated</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>JIS87</td>
<td>JIS B7</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>JIS88</td>
<td>JIS B8</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>JIS89</td>
<td>JIS B9</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>NorthAmerica10x11</td>
<td>10&quot;x11&quot;</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>NorthAmerica10x12</td>
<td>10&quot;x12&quot;</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>NorthAmerica10x14</td>
<td>10&quot;x14&quot;</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>NorthAmerica11x17</td>
<td>11&quot;x17&quot;</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>NorthAmerica14x17</td>
<td>14&quot;x17&quot;</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>NorthAmerica4x6</td>
<td>4&quot;x6&quot;</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>NorthAmerica4x8</td>
<td>4&quot;x8&quot;</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>NorthAmerica5x7</td>
<td>5&quot;x7&quot;</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>NorthAmerica8x10</td>
<td>8&quot;x10&quot;</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>NorthAmerica9x11</td>
<td>9&quot;x11&quot;</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>NorthAmericaArchitectureASheet</td>
<td>Arch A</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>NorthAmericaArchitectureBSheet</td>
<td>Arch B</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>NorthAmericaArchitectureCSheet</td>
<td>Arch C</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>NorthAmericaArchitectureDSheet</td>
<td>Arch D</td>
</tr>
<tr>
<td>Feature Name</td>
<td>Option Name</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>--------------------------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>NorthAmericaArchitectureESheet</td>
<td>Arch E</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>NorthAmericaCSheet</td>
<td>C Size Sheet</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>NorthAmericaDSheet</td>
<td>D Size Sheet</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>NorthAmericaESheet</td>
<td>E Size Sheet</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>NorthAmericaExecutive</td>
<td>Executive</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>NorthAmericaGermanLegalFanfold</td>
<td>German Legan Fanfold</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>NorthAmericaGermanStandardFanfold</td>
<td>German Standard Fanfold</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>NorthAmericaLegal</td>
<td>Legal</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>NorthAmericaLegalExtra</td>
<td>Legal Extra</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>NorthAmericaLetter</td>
<td>Letter</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>NorthAmericaLetterExtra</td>
<td>Letter Extra</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>NorthAmericaLetterPlus</td>
<td>Letter Plus</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>NorthAmericaLetterRotated</td>
<td>Letter Rotated</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>NorthAmericaMonarchEnvelope</td>
<td>Envelope Monarch</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>NorthAmericaNote</td>
<td>Note</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>NorthAmericaNumber10Envelope</td>
<td>Envelope #10</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>NorthAmericaNumber10EnvelopeRotated</td>
<td>Envelope #10 Rotated</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>NorthAmericaNumber11Envelope</td>
<td>Envelope #11</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>NorthAmericaNumber12Envelope</td>
<td>Envelope #12</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>NorthAmericaNumber14Envelope</td>
<td>Envelope #14</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>NorthAmericaNumber9Envelope</td>
<td>Envelope #9</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>NorthAmericaPersonalEnvelope</td>
<td>Envelope Personal</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>NorthAmericaQuarto</td>
<td>Quarto</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>NorthAmericaStatement</td>
<td>Statement</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>NorthAmericaSuperA</td>
<td>Super A</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>NorthAmericaSuperB</td>
<td>Super B</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>NorthAmericaTabloid</td>
<td>Tabloid</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>NorthAmericaTabloidExtra</td>
<td>Tabloid Extra</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>OtherMetricA3Plus</td>
<td>A3 Plus</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>OtherMetricA4Plus</td>
<td>A4 Plus</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>OtherMetricFolio</td>
<td>Folio</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>OtherMetricInviteEnvelope</td>
<td>Envelope Invite</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>OtherMetricItalianEnvelope</td>
<td>Envelope Italian</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>PRC10Envelope</td>
<td>PRC Envelope #10</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>PRC10EnvelopeRotated</td>
<td>PRC Envelope #10 Rotated</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>PRC16K</td>
<td>PRC 16K</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>PRC16KRotated</td>
<td>PRC 16K Rotated</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>PRC1Envelope</td>
<td>PRC Envelope #1</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>PRC1EnvelopeRotated</td>
<td>PRC Envelope #1 Rotated</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>PRC2Envelope</td>
<td>PRC Envelope #2</td>
</tr>
</tbody>
</table>

May 14, 2012
© 2012 Microsoft. All rights reserved.
<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Option Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PageMediaSize</td>
<td>PRC2EnvelopeRotated</td>
<td>PRC Envelope #2 Rotated</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>PRC32K</td>
<td>PRC 32K</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>PRC32KBig</td>
<td>PRC 32K (Big)</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>PRC32KRotated</td>
<td>PRC 32K Rotated</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>PRC3Envelope</td>
<td>PRC Envelope #3</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>PRC3EnvelopeRotated</td>
<td>PRC Envelope #3 Rotated</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>PRC4Envelope</td>
<td>PRC Envelope #4</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>PRC4EnvelopeRotated</td>
<td>PRC Envelope #4 Rotated</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>PRC5Envelope</td>
<td>PRC Envelope #5</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>PRC5EnvelopeRotated</td>
<td>PRC Envelope #5 Rotated</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>PRC6Envelope</td>
<td>PRC Envelope #6</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>PRC6EnvelopeRotated</td>
<td>PRC Envelope #6 Rotated</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>PRC7Envelope</td>
<td>PRC Envelope #7</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>PRC7EnvelopeRotated</td>
<td>PRC Envelope #7 Rotated</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>PRC8Envelope</td>
<td>PRC Envelope #8</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>PRC8EnvelopeRotated</td>
<td>PRC Envelope #8 Rotated</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>PRC9Envelope</td>
<td>PRC Envelope #9</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>PRC9EnvelopeRotated</td>
<td>PRC Envelope #9 Rotated</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>Roll04Inch</td>
<td>Roll 4&quot;</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>Roll06Inch</td>
<td>Roll 6&quot;</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>Roll08Inch</td>
<td>Roll 8&quot;</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>Roll12Inch</td>
<td>Roll 12&quot;</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>Roll15Inch</td>
<td>Roll 15&quot;</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>Roll18Inch</td>
<td>Roll 18&quot;</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>Roll22Inch</td>
<td>Roll 22&quot;</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>Roll24Inch</td>
<td>Roll 24&quot;</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>Roll30Inch</td>
<td>Roll 30&quot;</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>Roll36Inch</td>
<td>Roll 36&quot;</td>
</tr>
<tr>
<td>PageMediaSize</td>
<td>Roll54Inch</td>
<td>Roll 56&quot;</td>
</tr>
</tbody>
</table>
12 Appendix B: Brief Technical Requirements

12.1 v4 Print Drivers
V4 print drivers may be distributed on Windows Update or directly by the manufacturer.

12.1.1 Naming
V4 print drivers should follow the prescribed naming guidance. Manufacturer codes should be the same as in Windows 7. If a manufacturer code has not been established for your brand or company, the manufacturer name may be used to guarantee uniqueness. 8.3 notation is not required for any driver files.

Here are some examples using FA as the manufacturer code for Fabrikam, the fictional company that is used in many samples in this whitepaper:

- **File**: fafoo.dll
- **Pipeline config**: FA-*pipelineconfig.xml
- **Manifest**: FA-*manifest.ini

12.1.2 Rendering
V4 print drivers may provide either XPSDrv rendering or take a dependency on an existing print class driver by using the RequiredClass directive (section 7.2).

12.1.2.55 XPSDrv Rendering
Manufacturers may provide full-featured XPSDrv rendering in their v4 print drivers. They must provide a valid Print Filter Pipeline Config (section 4.1.2), but should include any filters deemed necessary.

12.1.2.56 RequiredClass
Manufacturers may also choose to take a dependency on any existing print class driver by using the RequiredClass directive (section 7.2). This allows partners to create processor architecture neutral, model-specific drivers. However, driver INFs must still contain architecture-specific sections to comply with PnP rules even if there are no processor-specific files being deployed in the driver package.

12.1.3 Configuration
The device configuration represented in v4 print drivers should reflect the full capabilities and constraints of the device. These drivers must not provide configuration plug-ins, but should contain any GPD, PPD, Autoconfig GDL, and Bidi extension XML files that are necessary. All features should be fully described in the GPD or PPD data files.

- **GPD**
  - Must include msxpsinc.gpd
- **PPD**
  - Must include *MSIsXPSDriver: True
- **Resource DLL (section 5.3.4)**
• Bidi Extension XML (section 8.3)
• Autoconfig GDL (section 8.3)
• Driver Property Bag (section 5.3.1)
• JavaScript Constraints (section 0)
• Printer extension (section 6.3)

12.1.4 Setup

12.1.4.57 INFs
In the v4 driver model, printer INFs have been simplified with most print-specific details being moved into the driver’s manifest. V4 print driver INFs must specify the following directives:
• ClassVer=4.0

More information about print driver INFs is available with a sample in section 7.

12.1.4.58 Manifests
V4 print driver manifests may specify any v4 manifest directives except:
• DriverFile - This directive is reserved for print class drivers. Unless otherwise specified by a print class driver, all v4 drivers use mxdwdrv.dll as the DriverFile.

12.1.5 Connectivity
V4 print drivers may provide the following connectivity files:
• Bidi extension files
• Autoconfig GDL files
• JavaScript USB Bidi Extensions

12.2 Print Class Drivers
Print class drivers are a special variant of the v4 print driver model that will be allowed to ship on the Windows OS media. Print class drivers are designed to deliver basic functionality to as many devices as possible. They may also specialize into family-specific class drivers, as described in section 7.3.

Following is an overview of what is and is not permitted in print class drivers. This section must be used in conjunction with the rest of the paper to understand the full scope of requirements for a v4 class driver.

12.2.1 File Naming
Print class drivers should follow the prescribed file naming guidance:
• Manufacturer codes should be the same as those used in Windows 7.
• 8.3 notation is not required for driver files.
• However, note that there are naming conventions for the INF file that happen to be in an 8.3 format. INF files should be named using the following format:
  prn\V\CLn.inf
VV = the manufacturer code
n = a number or letter

Here are some examples using FA as the manufacturer code for Fabrikam, the fictional company that is used in many samples in this whitepaper:

- **INF**: prnfac12.inf
- **File**: fafoo.dll
- **Pipeline config**: FA-PDL-pipelineconfig.xml
- **Manifest**: FA-PDL-manifest.ini

### 12.2.1.59 Print Class Driver Naming

Print class drivers should be named with the following format in order to improve supportability:

- `<Manufacturer Name> <Descriptive Class Name> Class Driver`
- **Examples**:
  - Fabrikam LaserPro Class Driver
  - Fabrikam InkPrinter PCL Class Driver
  - Fabrikam Color LaserPro PS Class Driver

Print class drivers that specify many HardwareIDs may also list model names to help users select the correct driver using the Add Printer Wizard.

### 12.2.2 Rendering

Print class drivers may provide either XPSDrv rendering or Unidrv/PScript GPD/PPD only rendering using the V3HostingFilter.

### 12.2.2.60 XPSDrv Rendering

Manufacturers that provide XPSDrv rendering for a class driver must submit an architecture review as documented in the Driver Check-in Guide. They may provide an IHV Rendering Filter (section 4.1.2) to convert from XPS to their PDL and may optionally provide an IHV Inbox Filter (section 4.1.4). They must provide a valid Print Filter Pipeline Config (section 4.1.2).

### 12.2.2.61 Unidrv/PScript Rendering

Manufacturers that provide Unidrv/PScript based rendering must provide GPD/PPD files as well as a Print Filter Pipeline Config that specifies only v3hostingfilter.dll as a filter.

### 12.2.2.62 Configuration

Configuration provided in print class driver packages should be very basic and common to all devices that support a particular class. These GPD and PPD files should support only the "lowest common denominator" paper sizes, features, and device constraints. This will allow new devices to be supported after RTM with no effort from partners or Microsoft.

Examples of expected functionality include:

- Paper sizes such as A4, letter, postcard, and photo
- Color
- Duplex on devices with duplexing hardware
- Photo printing on devices that support this scenario

Autoconfig and Bidi extension XML will be available in print class driver packages, but should also be common to all devices that support this class driver. This can allow devices to support advanced features like finishing tasks without device-specific configuration files.

The following files are the only files permitted in print class drivers:

- GPD
  - Must include msxpsinc.gpd
- PPD
  - Must include *MSIsXPSDriver: True
- Resource DLL (section 5.3.4)
- Bidi Extension XML (section 8.3)
- Autoconfig GDL (section 8.3)
- Driver Property Bag (section 5.3.1)

**12.2.2.63 Setup**

**12.2.2.63.1 INFs**

Print class driver INFs must be valid v4 driver INFs and specify the following directive:

- GenericDriverInstalled (section 7.1.2)

More information on v4 driver INFs and a sample INF are available in section 7.1.1.

**12.2.2.63.2 Manifests**

Print class driver manifests may specify any v4 manifest directives except:

- RequiredClass – Print class drivers cannot specify a dependency on any other print class drivers.

**12.2.2.63.3 Connectivity**

Print class drivers may provide the following connectivity files:

- Bidi extension files
- Autoconfig GDL files
- JavaScript Bidi USB Extension