Printing

Outline

- Printing and XPS
- XpsDocumentWriter
- PrintTicket & PrintCapabilities
- PrintDialog
- System.Printing

XPS – XML Paper Specification

- Native WPF print spool format
- Fixed layout documents

XpsDocumentWriter

- **Printing:**

```
XpsDocumentWriter dw = PrintQueue.CreateXpsDocumentWriter(...);
```

- **XPS file creation:**

```
XpsDocument xpsDoc = new XpsDocument("Out.xps", FileAccess.Write);
XpsDocumentWriter dw = XpsDocument.CreateXpsDocumentWriter(xpsDoc);
```

Simple Printing

```
// 1: Get XpsDocumentWriter

PrintDocumentImageableArea area = null;
XpsDocumentWriter dw = PrintQueue.CreateXpsDocumentWriter(ref area);


// 2: Generate content

TextBlock tb = new TextBlock(); tb.Text = "Hello, world!";
tb.Margin = new Thickness(area.OriginWidth, area.OriginHeight, 0, 0);


// 3: Perform layout

Size outputSize = new Size(area.MediaSizeWidth, area.MediaSizeHeight);
tb.Measure(outputSize); tb.Arrange(new Rect(outputSize));
tb.UpdateLayout();


// 4: Write content to XpsDocumentWriter

dw.Write(tb);
```

Printing Multiple Pages

- Can only call Write once

```
XpsDocumentWriter dw = ...;

dw.Write(firstPage);
dw.Write(secondPage); // ERROR!
```

Multi-page: CreateVisualsCollator

```
XpsDocumentWriter dw = ...;
SerializerWriterCollator c = dw.CreateVisualsCollator();
c.BeginBatchWrite();
for (int i = 1; i <= 10; ++i)
{
    TextBlock tb = new TextBlock();
    tb.Text = i.ToString();
    tb.TextAlignment = TextAlignment.Center;
    tb.VerticalAlignment = VerticalAlignment.Center;
    tb.FontSize = 500;
    tb.FontFamily = new FontFamily("Consolas");

    tb.Margin = new Thickness(area.OriginWidth, area.OriginHeight, 0, 0);
    Size outputSize = new Size(area.MediaSizeWidth, area.MediaSizeHeight);
    tb.Measure(outputSize);
    tb.Arrange(new Rect(outputSize));
    tb.UpdateLayout();

    c.Write(tb);
}
c.EndBatchWrite();
```

Multi-page: DocumentPaginator

```
XpsDocumentWriter dw = ...;
FlowDocument myFlowDocument = ...;

IDocumentPaginatorSource ps = myFlowDocument;
DocumentPaginator paginator = ps.DocumentPaginator;

dw.Write(paginator);
```

- **IDocumentPaginatorSource implementations:**
  - FixedDocument
  - FlowDocument

Multi-page: XPS

```
public class XpsDocumentWriter
{
    ...

     public override void Write(FixedDocumentSequence fixedDocumentSequence);

     public override void Write(FixedDocument fixedDocument);

     public override void Write(FixedPage fixedPage);

    ...
}
```

Asynchronous Printing

- **WriteAsync**
- **Progress and completion events:**
  - WritingCompleted
  - WritingProgressChanged
- **CancelAsync**
- **Requirements:**
  - Must run WPF dispatcher
  - Must keep objects around

PrintTicket

- **Controls printing options**
  - Collation, orientation, resolution, quality etc.
- **Ticket scopes**
  - Job, document and page
- **Default tickets**
  - UserPrintTicket: user preferences
  - DefaultPrintTicket: system defaults
- **PrintCapabilities**
  - Valid PrintTicket options for specific printer
  - Printer-specific custom options

PrintDialog

- **Explicit control of print dialog**

```
PrintDialog pd = new PrintDialog();
pd.PageRangeSelection = PageRangeSelection.AllPages;
pd.UserPageRangeEnabled = true;
pd.ShowDialog();
```

- **Often not necessary**
  - Creating XpsDocumentWriter can open automatically

Queues and Servers

- **System.Printing namespace**
  - Discovery
  - Monitoring
  - Management and configuration
- **PrintServer**
  - LocalPrintServer
- **PrintQueue**

Summary

- Printing and XPS
- XpsDocumentWriter
- PrintTicket & PrintCapabilities
- PrintDialog
- System.Printing