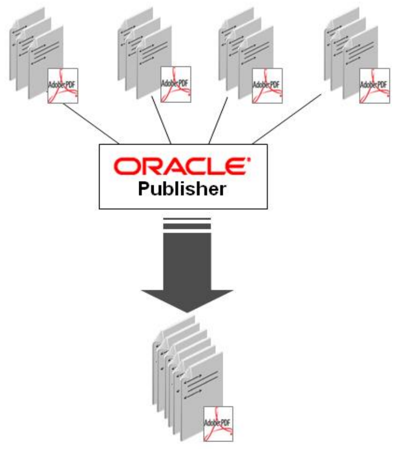
Bookbinding with Oracle BI Publisher

An Oracle White Paper
May 2010

INTRODUCTION

Many organizations need to be able to create consolidated documents; these are made up of a set of sub documents that may have come from disparate systems. The documents must to be brought together into a single combined document and made to look like they are a single contiguous document. Examples of such documents might be, a yearly budget book presented by government agencies or a shipping document that is made up of the main document plus supporting material documents.



This process is typically a manual one requiring significant investments of time and effort to bring the sub documents together and then to create a consolidated master document.

Oracle Business Intelligence Publisher (BI Publisher) is an enterprise reporting solution to author, manage, and deliver all types of highly formatted documents eliminating the need for costly point solutions. End users can easily design report layouts using familiar desktop tools, dramatically reducing the time and cost needed to develop and maintain reports. Built on open standards, IT staff and developers can create data models against practically any data source and use BI Publisher APIs to build custom applications leveraging existing data sources and infrastructure. Extremely efficient and highly scalable, BI Publisher can generate tens of thousands of documents per hour with minimal impact to transactional systems.

Oracle Business Intelligence Publisher offers the ability to simplify and automate the consolidation of documents. It provides a set of application program interfaces to allow customers to provide various levels of sophistication in their combined documents. Whether for simple document concatenation or combining documents containing table of contents, cover pages, section covers, section page numbering and master page numbering BI Publisher offers the full gamut to satisfy almost any consolidation requirement.

BINDING OPTIONS

Oracle BI Publisher provides the ability to take multiple disparate documents and merge them into a single PDF output.

BI Publisher offers multiple ways to bind documents together, from the simple concatenation of files to the more sophisticated joining of files with overlays on top of the final document including multiple document features. The various binding options are achieved by using BI Publisher's java APIs. There is currently no out-of-the-box user interface to achieve the bind. It must be handled using either a java class or a servlet approach.

This document will cover the simpler API calls to combined documents through to the more sophisticated use of the bookbinding API.

Simple Binding

Using the published java doc available from BI Publisher's home page: http://download.oracle.com/docs/cd/E12844_01/doc/bip.1013/e12693/toc.htm. One can see the two available binding classes, the PDFDocMerger class and the more sophisticated, PDFBookbinder class.
The simpler API offers the ability to bind multiple PDF documents together and to overlay page numbering and provide background images to the document. User requirements might be more complicated, requiring more features in the final document. Such as cover pages, chapter page numbering, table of contents, bookmarks, etc. To achieve this, the PDFBookbinder API must be used.

For simpler concatenations the PDFDocMerger class can be used.
Its usage is as follows:

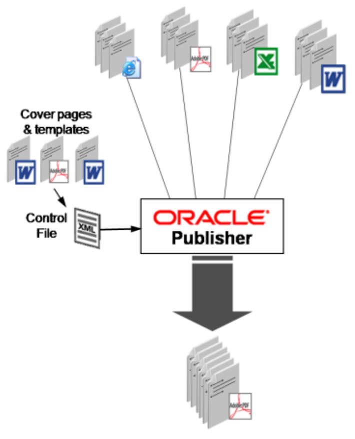
```
// PDF documents to be merged.
 InputStream[] inStreams;

 // Output destination of the merged PDF document.
 OutputStream outStream;
  :
  :
  :
 // Create PDFDocMerger instance.
 PDFDocMerger pdfMerger = new PDFDocMerger(inStreams, outStream);
pdfMerger.setConfi(props);

 // Run the merging process.
 pdfMerger.mergePDFDocs();
```

As can be seen from the example code, the API requires an array of input stream PDF documents. These can be brought in from the disc or from URLs but they must be converted into java *InputStreams* for the API to work upon them. The result is a concatenated PDF output file. The API provides the ability to add page numbering on top of the output document; this is achieved by calling *setPageNumberXXX* methods, on the same API. A background image can be added to every page of the resulting document by using the *setBackground* and *setWaterMark* methods. The *setConfig* method allows the user to set further properties on the final document, such as document security settings.

Sophisticated Binding

The PDFBookbinder class provides many more features for the final document. The available features and their description are as follows:

- Page numbering – this can be master page numbering for the complete document and/or 'chapter' page numbering.
- Table of Contents – a table of contents based on the child documents can be created to ease navigation around the document.
- Bookmarks – an extension of the table of contents, 'bookmarks' can also be created to ease document navigation when viewing within Adobe Reader (or similar PDF viewer.)
- Start/ End pages – the document can have cover and ending pages providing the flexibility to provide dynamic cover pages depending on the document to be generated.
- Section or Chapter Start/End pages – each child document or set of child documents can be specified as a chapter or section within the master document. These can each be wrapped by a section start and end page.
- Graphic/Text Overlays – to help the document look more cohesive one can add a common graphic or overlay on every page of the document e.g. logo or header text.
- Cross references – if the master document contains both summarized and detail data. Cross reference links can be added to enable the reader to drill down from a summary to a detail page when viewing the document in a PDF reader.
- Document splitting – consolidated documents that become very large can become unwieldy for users to open. Publisher provides the ability to split the document into smaller documents. Upon opening the user is still able to see and navigate the table of contents and bookmarks. If they click on a link that is stored in another file, that file is loaded silently in the background.



The binding process needs the sub documents, any overlay or template files for the common header, footer, table of contents, etc. These templates are all created using either MSWord and the RTF format or PDF files. To control the merging of the documents a control file needs to be created, this will instruct the binding engine on all aspects of the final document creation such as, the order of the documents, the overlays to be used, page numbering position, master/section page numbering, etc.

Once the binding process is complete the output document can be written to an accessible directory on a server or local machine. Alternatively, it could be printed or delivered by some other delivery channel.

The final document consumer does not require any client side install on their machine to read the document other than Adobe Reader. It is an Adobe PDF format and therefore the user needs only Adobe Reader or a similar software application to read the document.

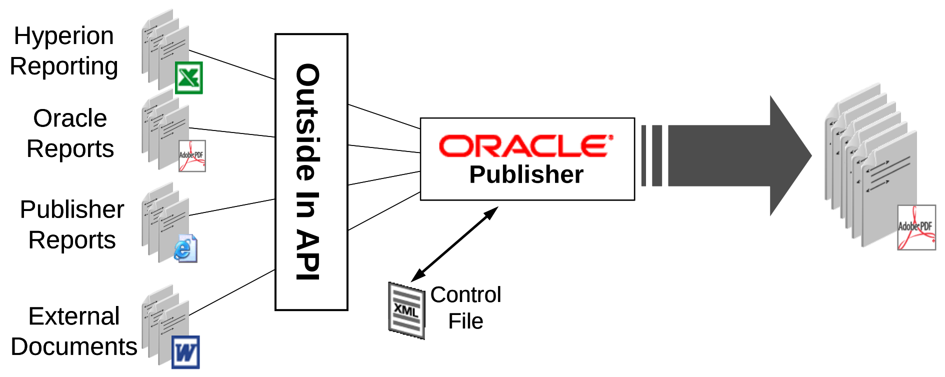
During the binding process, the binding API reads the control file. The control file defines the sub document locations and the order in which they will appear. It also defines the required document features that should be added to the final document.

If the document needs to be secured; Oracle BI Publisher can apply password protection to the document (at both Reader and Acrobat Professional levels.) Users can be prevented from copying text from or printing the document; it can be encrypted and a digital signature added. This is all achieved by setting the appropriate properties for the document prior to generation.

Binding other Formats

In the current release, 10.1.3.4.1, BI Publisher is capable of binding sub PDF documents. If the user wishes to bind other document formats, then they must first be converted to the PDF format. Oracle Universal Content Manager currently provides the Outside In API's to achieve this conversion, this requires a further license. More information on the Outside In technology is available here, http://www.oracle.com/us/products/middleware/content-management/outside-in-tech/index.html
The Outside In java API can be used to convert the base documents prior to calling the PDF bookbinder API. This document conversion technology will be incorporated into the BI Publisher binding API in a future release.

Once the Outside In APIs are installed the java API can be called to convert many of the major document formats to PDF.



Bookbinder under the Covers

The bookbinding process can source documents from multiple systems and in multiple formats (currently with the user of the InsideOut API) such as MSOffice, PDF, and HTML. It can also pull image formats such as jpg, gif, png, etc and merge those into the final document.

As with the PDFDocmerger, the API is accessed via a java call. The use of the API is as follows:

```
String xmlInputPath = "c:\\tmp\\ctrl.xml";
String pdfOutputPath = "c:\\tmp\\final_book.pdf";
PDFBookBinder bookBinder = new PDFBookBinder(xmlInputPath, pdfOutputPath);
 bookBinder.setConfig(new Properties());
 bookBinder.process();
```

As mentioned earlier, the API employs the use of a control file to control the binding process. This is an XML file that defines the sub documents and the features that need to be added to the final document. Once invoked, the API will parse the control file, bringing in the sub documents and applying the requested features to the output document. The setConfig method allows the application of further features to the document such as security.

Typically, the API is called from an embedded Java class, or from a servlet listening for binding requests. The API could equally be wrapped up inside a web service or could be wrapped inside a PL/SQL call.

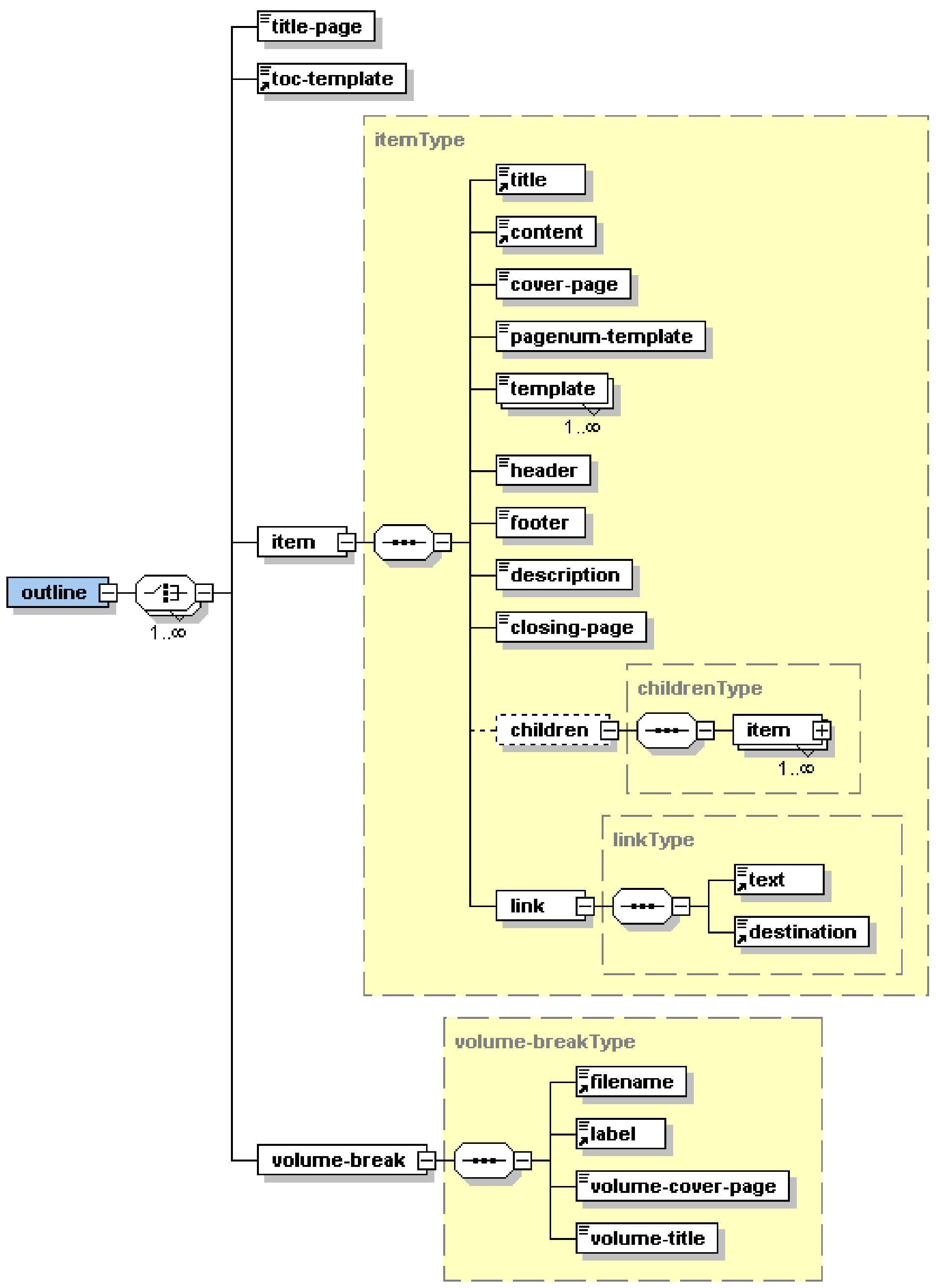
The API has a couple of requirements; when called, the sub documents need to be accessible either via the file system or a URI. Other overlay or template files must also be accessible via the same means.
The other requirement of the API, is the control file; this is an XML file that defines how the bind should proceed, the order of the documents, start/end pages, etc.

Here is a snippet of a control file:

```
<outline xmlns="http://xmlns.oracle.com/oxp/book/">
<!--Here is the title page reference →
 <title-page type="rtf">templates/title_page.rtf</title-page>
 <toc-template>templates/toc-template.rtf</toc-template>
 <item>
  <title>Oracle Database Overview</title>
  <content>../oracle_books/database_overview.pdf</content>
  <cover-page type="rtf">templates/chapter-cover.rtf</cover-page>
  <pagenum-template type="rtf">templates/pagenum.rtf</pagenum-template>
  <template type="pdf">templates/overlay.pdf</template>
  <template type="rtf">templates/overlay.rtf</template>
  <header xmlns="">Oracle Database</header>
  <footer xmlns="">Oracle Database</footer>
  <description xmlns="">Oracle Database 11g ...</description>
  <closing-page type="rtf">templates/chapter-closing.rtf</closing-page>
  <children>
    <item>
    <title>Oracle Database Enterprise Edition</title>
    <content>../oracle_books/Database10gR2_EE.pdf</content>
<cover-page type="rtf">templates/chapter-cover.rtf</cover-page>
```

The control file is simply built up sequentially adding the sub documents and the features that are required by the binding specification. This can be done by hand if the format and structure of the final document is known. It can also be built up based on a set of user inputs (custom UI.) Code can be written to construct the control file on an ad-hoc basis prior to calling the API. Some examples follow later on in this document.

Once called, the binding engine will read the control file and then start constructing the final document. It will analyze the incoming document formats and carry out necessary conversions of documents prior to the bind. As the final document is constructed the required features are added based on the control file.



Once completed it is then written to the disk. If the file needs to be delivered to users via email or perhaps put onto a web server, Publisher provides a rich set of APIs to deliver the document to its final destination.

The basic structure of the control file is as follows:

- title-page – is the element for a consolidated document title page.

```
<title-page type="rtf|pdf">template file location</title-page>
```

The type attribute can be either pdf or rtf. The file location is then referenced in the element. Optional. Single entry.

- toc-template – is the element for the location of the table of contents template.
  `<toc-template>templates/toc-template.rtf</toc-template>`
  This can only be an RTF template. Optional. Single entry.
- item – this is the container group for the documents that will make up the sub documents of the consolidated document. Each item represents a sub document and the attributes that need to be added to that document, such as an opening/closing page, page numbering, etc. These are defined as sub elements.
  - title – this is a string that will be used by the cover page entry

    `<title>Oracle Database Overview</title>`
  - content – a reference to the sub document. Needs to be a pdf.

    `<content>../oracle_books/database_overview.pdf</content>`
  - cover-page – this is the cover page template. It can use the title, header, footer and description elements. It is just like a regular rtf or pdf template in its capabilities to reference data items.

    `<cover-page type="rtf">chapter-cover.rtf</cover-page>`
  - pagenum-template – the template used to position and apply the page numbers for the sub document. Optional. RTF or PDF

    `<pagenum-template type="rtf">pagenum.rtf</pagenum-template>`
  - template – multiple layout or overlay templates can be added at this point. This might include common graphics or text to be applied to every page in the sub document. Optional RTF or PDF

    `<template type="pdf">templates/overlay.pdf</template>`

    `<template type="rtf">templates/volume-overlay.rtf</template>`
  - header – text for header
    `<header xmlns="">Oracle Database</header>`
  - footer – text for footer

    `<footer xmlns="">Oracle Database</footer>`
  - description – text to describe the sub document
    `<description xmlns="">Oracle Database 10g </description>`
  - closing-page – reference to closing page template. Optional. RTF or PDF
    `<closing-page type="rtf">chapter-closing.rtf</closing-page>`

- children – the structure of the document may require parent-child relationships between the sub documents. To achieve this the <children> element can be inserted. Then start adding more <item> entries for the child documents. The levels of nesting are unlimited.

Using this approach a hierarchy of sub documents can be built up to make up the structure of the consolidated document.

```
<outline>
 <title-page>
 <toc-template>
 <item>
  <title>
  <content>
  <cover-page>
  <pagenum-template>
  <template>
  <template>
  <header>
  <footer>
  <description>
  <closing-page>
  <children>
   <item>
    <title>
    <content>
    …
   </item>
   <item>
    <title>
    <content>
    …
 </item>
   <item>
    <title>
    <content>
    …
   </item>
  </children>
 </item>
 <item>
  <title>
  <content>
  …
 </item>
</outline>
```

Volume Splitting

If the generated PDF document is likely to be large; greater than 10 Mb, it can then it can be split into volumes. Users can load any one of the volumes into their PDF reader software. The complete set of navigational bookmarks will be present in any of the volumes. If a particular bookmark points to another volume document, it will be loaded automatically into the reader.

To achieve the volume splitting place the following entry into the control file at the position the document should be split.

```
  <volume-break>

   <filename>Volume3.pdf</filename>

   <label>III</label>
   <volume-cover-page type="rtf">volume-cover.rtf</volume-cover-page>
   <volume-title xmlns="">Volume 3</volume-title>
  </volume-break>
```

- filename – this is the name given to the volume. Required.
- label – this value can be referenced in a cover page template. Optional.
- volume-cover-page – a reference to the cover page template RTF or PDF. Optional.
- volume-title – a title for the volume that can be referenced by the cover page. Optional.

Cross Linking

Cross links or references can be created in the consolidated document. This allows users to navigate from one sub document to another sub document section using a clickable link.
To create a link to a sub document, add an extra attribute, "id", at the item level in the control file. The value can be any string or number value but it must be unique to the control file.
`<item id="odb">`
The document where the link will be place requires a <link> element placed inside the closing </item> tag.

```
  <link>
   <text>Oracle Database Overview</text>
   <destination>odb</destination>
  </link>
 </item>
```

- text – this is the text that will appear in the link
- destination – this is the pointer to the id that the link will take the user to.

The link itself will appear on the last page of the subdocument as a hyperlink in the PDF document.

Binding Templates

As has been seen in the control file structure, BI Publisher templates can be used to create overlays and to format the table of contents.

- **Title Page** – this can be either an RTF or PDF document. Typically, this will be a static cover page for the consolidated document.
- **Table of Contents** – this can only be an RTF template. The commands to create the table of contents are fixed. However the look and feel of the TOC text can be modified using MSWord functionality. An example of the table of contents template is available in the templates directory of the demonstration files that accompany this paper.
- **Chapter Opening/Closing and Overlays** – these templates can be RTF or PDF formats. These can include images, static text; these are similar to the regular RTF templates in terms of MSWord features that can be used. These templates can reference the values in the control file for their specific section

i.e. the <item> section in which they appear. The data points can be the title, header, footer and description text. To include these values in the templates a specific namespace must be declared within the template:

```
<?namespace:xdobb=http://xmlns.oracle.com/oxp/book/?>
```

Values can then be referenced using the xdobb: prefix to the XML element value.

```
<?xdobb:title?>
```

At runtime values will be brought into the final output and placed in the cover pages. An example of this template is available in the templates directory of the demonstration files that accompany this paper.

- **Page Number** – this template must be an RTF document. It needs only include the commands to add the page numbers. The master document page numbering and the sub document numbering can be included in the template. It uses the xdobb: prefix, which must be declared in the template. The following commands are then used to add the page numbering.

  ```
  <?for-each:xdobb:page?>
              Book page <?xdobb:page-number?> of <?xdobb:page-total?>
    Chapter page <?xdobb:section-page-number?> of <?xdobb:section-
                                                        page-total?>
  <?split-by-page-break:?>
  <?end for-each?>
  ```

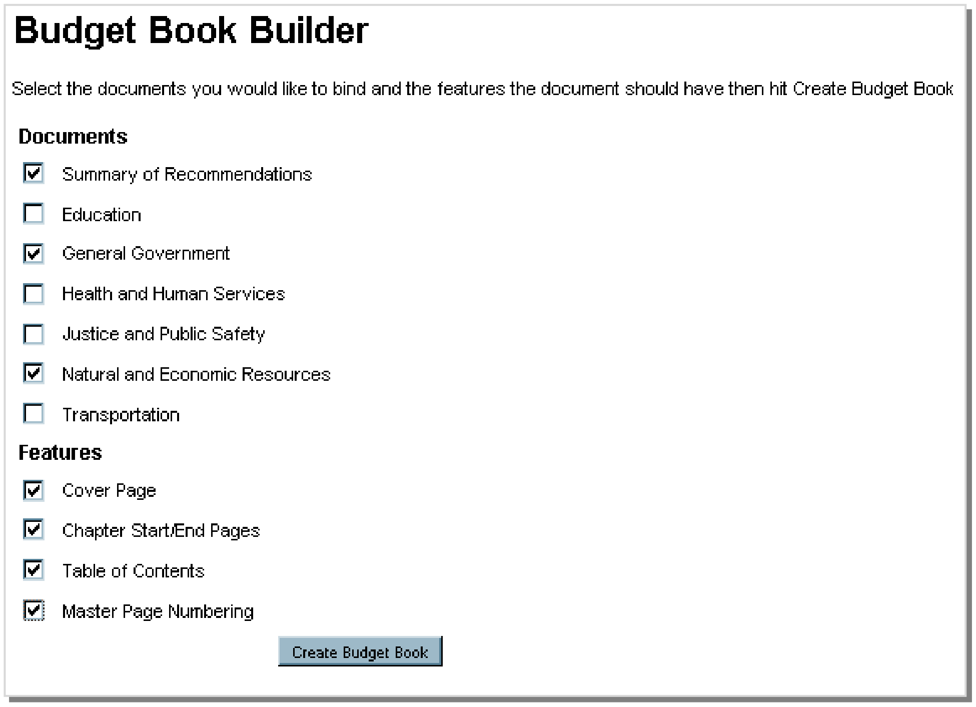
  An example of this template is available in the templates directory of the demonstration files that accompany this paper.

Implementing Binding

The binding API can be run by creating a java class or a servlet or any other java based implementation.

```
String xmlInputPath = "c:\\tmp\\ctrl.xml";
String pdfOutputPath = "c:\\tmp\\final_book.pdf";
PDFBookBinder bookBinder = new PDFBookBinder(xmlInputPath, pdfOutputPath);
 bookBinder.setConfig(new Properties());
 bookBinder.process();
```

In a servlet situation, the binding API can be set up to accept input files and a control file. It can then process the files and return the consolidated document. A user interface can be created to allow users to select the documents they wish to bind and the document features they want in the final output.


Budget Book Builder

Select the documents you would like to bind and the features the document should have then hit Create Budget Book

Documents

- [✓] Summary of Recommendations
- [ ] Education
- [✓] General Government
- [ ] Health and Human Services
- [ ] Justice and Public Safety
- [✓] Natural and Economic Resources
- [ ] Transportation

Features

- [✓] Cover Page
- [✓] Chapter Start/End Pages
- [✓] Table of Contents
- [✓] Master Page Numbering

Once the user clicks the Create Budget Book button, the document selections and features are passed to a servlet for processing. The sub-document order and parent-child relationships could be defined in a more sophisticated user interface. Processing code within the servlet then creates a control file on the fly. The bookbinding API is called to process the documents using the newly created control file; the consolidated document is returned to the browser.

Typically, budget books are very large documents. In the example, a 'run and receive' model will probably not be practical and the process may need to be run offline or scheduled for off hours.

If the input sub-documents are not in the PDF format the servlet can check the document formats and convert them using the OutsideIn APIs if available.

The binding API can also be called directly from the command line if necessary. It takes the following format:

```
java -debug true|false -tmp ..\temp_dir -xml control.xml -pdf output.pdf
```

- debug – turn debug information on or off. It can be directed to a log file if required.
- tmp – this is the temporary work directory for the binding process
- xml – the location and name of the control file
- pdf – the location and name of the output pdf file

Document Storage

The source documents can be referenced via a URL, by a directory location or from a database. Some customers may wish to store the source documents in a document repository. The documents can either be pulled from the repository using an API or the documents pushed to the API using a zipped format that standard APIs can be used to unpack them prior to being bound.

Demonstration Examples

To accompany this paper there is a set of worked examples.

- Example1 - Simple merge plus front page
- Example2 - Cover and Closing page for each document
- Example3 - Header - Footer Overlay
- Example4 - Master - document page numbering
- Example5 - Table of Contents
- Example6 - Creating Chapters
- Example7 - Cross Links
- Example8 - Volumes

Each demonstration builds on the previous example taking the user from a simple example through to a more complex bind with all of the available features.

CONCLUSION

Many organizations need to be able to periodically create consolidated documents for everything from financial briefing books to drug trial results. These are typically made up of multiple disparate documents that must be assembled in a specific order and bound together into a cohesive single document. For many organizations, accomplishing this is a time consuming and labor intensive manual process. Using Oracle BI Publisher's binding capabilities; an organization can streamline and automate this effort.

Bookbinding with Oracle BI Publisher
May 2010
Author:Tim Dexter

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com