Infix Server Version 4

Reference Manual

Copyright Notice

Notice Of Change

This document represents the state of version 3 of Infix Server. Iceni Technology reserves the right to make such changes at any time in the future and will make reasonable efforts to inform interested parties of the nature of the changes.

Contact Details

For the most up to date addresses and telephone numbers, please visit the contacts section of our web site at www.iceni.com

For specific support questions, use **support@iceni.com.** Otherwise, please use sales@iceni.com for general enquiries.

We welcome any constructive comments you may have regarding the content, style or layout of this document. We are also grateful for feedback on and feature requests for our products.

Please use the above contact information to get in touch.

Thursday, November 17, 2022

Introduction

InfixServer is the most versatile PDF manipulation tool available today. It is built upon Iceni's Infix PDF editing engine and our cross-platform PDF interpreter.

InfixServer can: insert text, images & PDFs; and replace text and images into existing or new PDF documents. An XML Command file that details the operations to be performed drives InfixServer. The program has a built-in spooler for totally automated operation or can be used as part of shell script and invoked on a per-document basis.

InfixServer is a highly sophisticated and flexible tool. In fact, the degree of complexity of configuration can be daunting for the new user. Therefore, it is recommended that you familiarise yourself with the principle aspects of the program before attempting to customise it for your own requirements.

Using This Manual

This reference manual describes the function of each configuration option in the configuration file and the commands in the XML Command File. It does not necessarily describe how they may be employed. For a more rounded understanding please consult the XML command files included with the InfixServer distribution in conjunction with this manual.

Example XML Command Files

Included with the Infix Server distribution are a number of sample XML Command Files. These can be found in the directory ExampleXMLCommandFiles. Each of the XML files contained within this directory is commented to describe its function. Using these in conjunction with this manual is an excellent way of learning how to use and set up Infix Server. The XML Command files supplied use relative paths to files where required and should be run from the directory containing the infixserver executable in the following manner. To generate a default config file run.

```
infixserver
```

Now run

```
Infixserver –c default.cfg –d ExampleXMLCommandFiles/<example.xml>
-l results.xml
```

Where the <example.xml> is the name of an xml file in the ExampleXMLCommandFiles directory. PDFs produced will be generated in the ExampleXMLCommandFiles directory.

Installation

Included on the physical mendia which comes with this package are all current versions of InfixServer: **Linux** (Intel), **MacOSX** (Intel) and **Windows.** If downloaded then the package downloaded will contain just Infix Server for the platform selected.

Installing the Software

Copy the relevant InfixServer distribution folder (Linux, MacOSX or Windows) to the host computer OR extract InfixServer from the package downloaded to the host computer.

The program will run only in demonstration mode until a valid license key is provided. In demonstration mode all PDFs produced will contain visible watermarks, which render them unusable for all but testing purposes.

Obtaining a License Key

All versions of InfixServer use a software key license based upon the identity of the host computer. A code unique to the host machine is required to generate a license key file which will be sent to you by Iceni.

To generate a default config file execute the command

```
Infixserver
```

This will generate a config file called default.cfg

To create the unique host-id for the machine running InfixServer execute the command

```
infixserver -c <config file>
```

Where <config file> can be the default.cfg file created in the previous step

On executing the command, a file called “hostid.txt” will be created. If you have purchased an InfixServer license please e-mail this file to sales@iceni.com and we will create a new license key based upon it.

When InfixServer runs it searches for a file called “infixserver.key”, which contains nothing but the special key supplied by Iceni. If it does not find the key file or the key does not match the host machine, InfixServer will operate in demo mode.

An alternative location for the key can be specified with the “-h” flag on the command line.

Operation

```
infixserver -c <config file> -h <host key filename> -d <xml data
file> -t <pdf template> -r <render file> -s <render scale> -o
<output path> -l <xml result path>
-spool -pdfspool
```

-c Specifies the path of a configuration file to use. If no command line parameters are supplied then the program will generate a default configuration file called “default.cfg”in the current working directory. This is often a good place to start when preparing a new configuration.

-h Specifies an alternate file path for the host id key file. The default is infixserver.key”.

-d Specifies the name of the XML Command file. The XML Command file specifies the operations to be performed (see **The XML Command File** Section). Should not be used with -spool parameter.

-f List all compatible fonts found in the path specified by the System Font Path entry in the given configuration file. The names generated in the list can be used for the "Place Text" command to add new text to documents.

-t Specifies a PDF Input file. If the XML command file does not specify a Input PDF File then this will be used instead. Any operations specified in the XML Command file will be performed on this file. Should not be used with the -spool parameter.

-r Specifies the path to a JPEG file to rendered. The PDF produced as a result of running Infix Server will be rendered to a JPEG file. The PDF will be rendered using the scale specified in the XML Command file or with the -s parameter. The PDF will be rendered at original size if neither are specified. This parameter will be ignored if the XML Command File also specifies a JPEG file to render to. Should not be used with the -spool parameter.

-o The output PDF file path. The path to which the PDF that is produced as a result of running Infix Server should be written. This parameter will be ignored if the XML Command file also specifies an output PDF path. Should not be used with the -spool parameter.

-s Specifies the scale as a percentage of its size to which the PDF produced as a result of running Infix Server will be scaled when rendered using the –r parameter. This parameter will be ignored if the XML Command file also specifies an render scale. Will have no effect if –r is not specified. Should not be used with the -spool parameter.

-l Specifies the Results XML path. An XML file will be written to this path that contains the results of running the commands specified in the XML Command file. If this paramater is not supplied then the XML results will be written to stdout. Should not be used with the -spool parameter. See **The Results XML** Section for more information.

-spool Puts Infix Server into spool mode. Spooler configuration is specified in the config file provided using the -c param. Please refer to the **Configuration File Section** for details of how the spooling operates.

-pdfspool Puts Infix Server into pdf spool mode. Spooler configuration is specified in the config file provided using the -c param. Please refer to the **Configuration File Section** for details of how the spooling operates. In this mode infixserver spools pdf files and applies the XML command file specified by the –d command line option to each one. For this reason the command xml should not specify an Input PDF file. If it does not specify an output file and render file then these will be <Output

Directory>/<input filename> & <Output Directory>/<input filename>.jpg. The results XML file will be named <Output Directory>/result_<input filename>.xml.

The Configuration File

The configuration file supplies information about the computer environment on which Infix Server will be running. It also defines the directories used when spooling if InfixServer is run in spool mode.

To generate a default config file execute the command “infixserver”. This will generate a config file called default.cfg.

The file is an ASCII file similar in form to the common Microsoft Windows preferences file format.

At the start of each main configuration file there are some version and owner data which is essential for InfixServer. For InfixServer 3.0, this header looks as follows:

```
ICENI PREFS
OWNER:InfixServer
VERSION:300
```

Supported Escape Sequences

The following escape sequences are supported by InfixServer when reading a value from a configuration file:

\NNN	3-digit octal character
\xNNNN	4-digit hexadecimal character
\\	Backslash character
\n	Carriage return
\r	Line feed
\t	Tab

Sections

The configuration file is divided into discrete sections using the format:

[***section name***]

[-- END --]

Each section contains simple key - value pairs of the form:

key = ***value***;

The <*value*> may span more than one line but must always be terminated with a final semi-colon “;”. Any newline characters included, will be taken literally and included in the definition of the value. This may be useful for producing ‘neat’ output such as HTML with line-breaks.

The terminating semi-colon ‘;’ is vital. Without it, the configuration file may still load without error but InfixServer may associate the wrong values with the wrong keys.

To include a newline character in the file but not in the definition (that is a line break in the configuration file to render it more readable) then place a backslash before the newline.

For example:

myKey =this is a value\

which will not eventually contain a new line\

even though the config file appears to contain two;

To include a semi-colon within the value itself, escape it using “\”. Similarly to include the back-slash character it needs to be escaped as in ‘\\’.

Everything after the semi-colon is ignored up until the next line starts. Also anything outside of a section is ignored. Either of these places can be used for comments.

[SPOOLER]

The spooler built into InfixServer has two modes of operation - *batch* and *queue*.

In *batch mode* a list of all files to be processed is gathered then InfixServer works its way through the list processing each file in turn. Once the list is empty it attempts to compile another list.

When in *queue mode*, the spooler operates a strict first-in, first-out queue where it continuously polls the input folder for new files. The oldest file found is processed first.

The advantage of *queue mode* is that InfixServer detects the files added to the input folder almost immediately. This means that files can be deliberately aged using a date stamp to ensure they go to the head of the processing queue.

The disadvantage comes when the input folder contains many matching files. The overhead of scanning and sorting all available files between each job can be considerable.

In *batch mode*, there is only one initial scan for files. From then on, files are processed one after the other in an unspecified order until none are left at which point another scan is done to see if there are any new files. Due to the infrequency of scans, it is not possible to add files to the input directory and have them noticed rapidly by InfixServer. However, this method involves very little overhead between jobs.

In either mode, as each job is processed an entry is written to a text based log file together with a time-stamp.

When spooling InfixServer will not change the current working directory. It will remain the directory in which Infixserver was started. Therefore files referenced in the XML Command Files being spooled need to be full paths or paths relative to the directory in which Infix Server was started.

Batch Mode

true | false

When true, InfixServer will operate in batch mode.

In Batch Mode, InfixServer scans the input folder once then processes each file found. Once all files have been processed it will re-scan the input folder for new files.

This mode of operating is faster than queue-based spooling and is suitable for those occasions when a large number of files have been accumulated ready for processing.

Input Dir

pathname

The folder to be polled for suitable input documents. All sub-folders of this folder will also be polled. Infix Server will expect the input documents to be XML command files. Please see the XML Command File section for more information.

Output Dir

pathname

The Results XML produced as a result of processing an XML Command File will be written to this directory. The Results XML file will be named

results_<XML Command File Name>.xml

Where <XML Command File Name> is the name of the XML Command File processed minus any extension.

Please see the **Results XML section** for more information.

Error Dir

pathname

If an error is encountered during processing, the XML Command File will be moved to this folder.

When in batch mode, if the value of *Error Dir* is the same as that of *Input Dir*, files are not moved but remain in the input directory.

Completed Dir

pathname

The folder in which to move XML Command Files that have been processed successfully. If this entry is omitted or it blank, then completed jobs are deleted on success.

When in batch mode, if the value of *Completed Dir* is the same as that of *Input Dir* then files are left in place. Once a batch is completed, processing stops and InfixServer exits.

Filter

file pattern

The pattern describing which files should be processed in the input folder. The pattern can include '*' which matched anything and '?' which matched single characters only.

For example:

```
Filter=*.xml;
Filter="Legal? Text.xml";
```

Only one filter may be supplied in a configuration file.

Pattern matching is case independent so for example, "*.xml:" would match files ending in ".XML" and ".xml".

Helios dt

pathname

Specifies the pathname of the 'dt' command used by Helios systems to perform various file operations whilst maintaining the Helios file database.

If InfixServer is to be used in spooling mode on a Helios file system, this option should be used. Example specifications are:

```
Helios dt=dt;        /* this works only if 'dt' is in the path of the shell executing InfixServer */

Helios dt=/usr/local/Helios/bin/dt;
```

Error Halt

true | *false*

Informs InfixServer whether it should halt all processing when a PDF causes an error. If true, the spooler stops and InfixServer quits. Otherwise, the job is moved to the error directory and spooling continues.

Keep Log

true | *false*

When true, InfixServer adds spooler activity to any log file specified in the Settings Section. Otherwise it does not.

Poll rate

rate in seconds

The delay between successive polls of the input directory. Setting it to a higher number causes a longer delay and less load on the host machine during idle periods.

The effect of this setting is most noticeable when in *queue mode*.

Stable time

time in seconds

Before InfixServer processes a matching input file, it will wait for a specified period of time to see if the file size if still changing. This may happen if for example, the file is being written across a busy network.

Waiting in this manner helps to ensure that only files which have been completely written are processed.

[SETTINGS]

Log File

pathname

The location of the log file. InfixServer activity is written to the log file together with a time stamp showing when the activity occurred.

In order to stop the log file from growing unchecked, InfixServer will truncate it periodically to any length required. See Max. Log Size below.

Max. Log Size (kB)
Log Trunc. Size (kB)

size in kilobytes

These two setting control how the log file is truncated. The Max size setting is the maximum size that InfixServer will allow the log file to grow. Beyond this, the size is truncated by removing a chunk from the start equal to Log Trunc Size.

Default Font Name

fontname

The name of a font to use if Infix Server cannot complete the operation in the XML Command File using the current font in the PDF or the font specified in the XML Command File. The font named will need to be in the directory specified by **System Font Path**. If not supplied or left blank then InfixServer will error when it cannot use the current font or the font specified.

This default font will be used whenever a bullet or hyphen is required and cannot be supplied by the fonts in the current PDF.

To obtain a list of font names in your font folder, use:

infixserver -c myconfig.cfg -f

which will dump all of the fonts found in the 'System Font Path' folder in myconfig.cfg

Built in Fonts Dir

pathname

The InfixServer is provided with a fonts directory and the pathname of this directory is specified in this key. The built in fonts directory is supplied with InfixServer and is called fonts.

System Font Path

*pathname[&pathname]**

This key specifies the pathnames of the system fonts directories. On Windows this will typically be something like C:/Windows/Fonts.

Multiple directories may be specified. For example:

System Font Path = infixserver/fonts/global**&**/infixserver/fonts/ms-corefonts;

CMaps Path

pathname

This key specifies the pathname of the Character Maps Directory. By default (if left blank), InfixServer will load the “cmaps” from the current working directory. A cmaps directory is supplied with InfixServer.

Max Pages Per Block

integer

This key specifies the number of pages of a PDF to be concurrently processed by InfixServer. The higher this value the more memory intensive Infix Server processing will be but processing will be faster. This key will have not effect if PDFs to be processed are less than the value specified.

New Document Path

pathname

This key specifies the pathname of the New Document file. This is used to create new PDF files. By default (if left blank), InfixServer will load “New Document.pdf” from the current working directory. The “New Document.pdf” is supplied with InfixServer.

Key Functions

access rights

This key should always be set to All unless instructed otherwise by Iceni.

CMYK Profile

filename

InfixServer includes a public-domain colour conversion engine (from www.littlecms.com), which helps to preserve colours when converting between colour spaces. The most problematic conversion - CMYK to RGB - is substantially improved by the use of this engine and suitable profile data.

This key specifies the colour profile to use when converting from CMYK colour spaces. By default (if left blank), InfixServer will load “cmyk-profiles/ISOwebcoated.icc” from the current working directory on Unix platforms and “profiles/USWebCoatedSWOP.icc” from the current working directory on MS Windows operating systems.

There are four public-domain profiles included with the InfixServer distribution in the “cmyk-profiles” folder. You may instead use any standard ICC profile defining a CMYK output space.

If you see an error such as

```
lcms: Error #12288; File' myProfile.icc' not found
```

it indicates that the profile specified cannot be found on disc. InfixServer will exit if this happens.

Hyphenation Dict

filename

InfixServer uses a hyphenation dictionary to determine where words can be hyphenated when flowing text into galleys. Each hyphenation dictionary available is for a particular language. This option should be set the hyphenation dictionary that matches the language of the PDFs to be processed. The hyphenation dictionaries included with InfixServer are:

dehyphn.tex	German
eshyph.tex	Spanish
frhyph.tex	French
ushyphen.tex	US English
ukhyphen.tex	UK English

Do Underlining

true | *false*

Turns the automatic recognition of underlines in the PDF on and off. If words are replaced that have a recognised underline in the original document then the replaced text will also be underlined correctly. Since this is a heuristic method it can be prone to error and it is recommended that this is set to false in a majority of cases.

Do Tabs

true | *false*

Enables the detection of tabstop in the PDF. There is no concept of a tab or tabstop in PDF. When this option is enabled, InfixServer will infer the existence of a tabstop. These tabs will be saved with the PDF in the structural information added by InfixServer. They will not effect the look of the PDF but may effect the way in which text reflows when edits are made.

Do Bullets

true | *false*

Enables the detection of unordered bullet lists in the PDF. There is no concept of a list or bullet in PDF. When enabled, InfixServer will infer the existence of bullet lists. This will not effect the appearance of a PDF but will effect reflow during text editing and translation export.

Do Runaround

true | *false*

When true, text will flow around images and other text boxes that overlap. This is the equivalent of setting a text-wrap margin around objects that collide with normal text flow.

Used during automated translation of PDF by transpdf.com

Do Columns

true | *false*

When true, the software looks for columns of aligned text on a page. Text is classified by column and centred accordingly. This produces better results when performing text replacements of pages that contain formatted data.

Used during automated translation of PDF by transpdf.com

Minimise Reflow

true | *false*

When true, detection and construction of tabstops and bullets it postponed and only performed on text blocks that are about to be edited. If a text box is not edited, no reflow or modification to it will be made at all when this flag is enabled.

Should be *true* during template processing and folio management to ensure fidelity of original pages.

Pay As You Go Key

string

Infixserver can be used to save documents on a pay-per-saved page basis. This optional key specifies the pay as you go key to be used for pay as you go operations.

Translation Config

string

This should always be set to be either the relative or absolute path to the TransXML.cfg file supplied as part of the InfixServer installation. The contents of the file specify the way in which XML is exported from PDFs by InfixServer.

If you need to modify the way XML is exported, you may make very limited edits to this specification. You must be cautious since InfixServer will not tolerate major changes to it.

If you need to include additional information in the exported XML, please get in touch with us since there are other, undocumented macros that may be included to achieve this.

Stamp Pdf

string

This should always be set to be either the relative or absolute path to the Stamp.pdf file supplied as part of the InfixServer installation.

Timeout

Integer

InfixServer will end execution of an XML command file if the CPU time taken to process it exceeds this value. The value is in seconds. The default value is 0 which means no timeout. If a timeout occurs then an error will be reported in the XML results within the XML tag for the command that was running at the point the timeout occurred.

Single Line Overset

true | *false*

If true, single line text boxes will grow eastwards when they cannot accommodate the text within them. This will stop them becoming overset but may cause their contents to overrun into other items on the page.

The default is *false*.

[LIMITS]

This section allows control over various internal limits governing specific aspects of behavior such as kerning and paragraph formation

BigGapRatio
FontSizeDiff
ParaFontDiff
MinFontSize
FontSizeGapRatio

All values integers.

MinFontSize

Text falling below the minimum font size will be ignored by Argus during text output. Default value 4pts.

FontSizeGapRatio

A ratio used when joining word fragments. Altering this value may be useful if your output contains many broken words due to unusually wide inter-character spacing.

Decrease the value to join more words together. Increase it to split more words apart. The default value is 89.

XML Command File

The XML Command File specifies the operations that are to be performed by Infix Server. It is either supplied using the command line -d argrument or is spooled from the “Input Dir” when Infix Server is run as a spooler.

The XML Command file is XML format. The top-level tag is InfixServer. The tags under this level are:

Version
Input
Command
Output
PostFile

The following tags are at the same level but are included for backwards compatibility with older versions of InfixServer. They should not be used for new configurations:

Document
Manifest
Data
DocumentInfo
Render

The commands available in each of these tags are defined in the rest of this chapter. This list also specifies the order in which Infix Server processes the commands specified in each tag, i.e. the commands in the Manifest section will be performed before the commands in the Data section etc. The ordering within the XML will not have any effect.

For examples of using particular features of InfixServer, please also see the XML Command Files included with the distribution.

Coordinates & Measurements - All coordinates and measurements used within XML Command Files have their origin at bottom left and are in points.

Page Numbers - Page numbers are indexed from 1.

Colours - Colours are specified using the col1, col2, col3 and col4 attributes of the tag that is specifying a colour. The value of these attributes should be in the range 0 to 255. If just col1 is supplied then it will be a greyscale value where 0 is black and 255 is white. If col1, col2, col3 are supplied then they will be an RGB colour where r is col1, g is col2 and b is col3. If all four are supplied then they will specify a CMYK colour where C is col1, M is col2, Y is col3 and K is col4.

For Example

```
<FillColour col1=”0”, col2=”0”, col3=”255”></FillColour>
```

Specifies an RGB colour that is bright blue.

XML Content and CDATA - Since the XML Command File conforms to the XML specification the content of any XML tag used within the XML command file can be enclosed within CDATA tags where it is desirable to do so.

Version

The InfixServer version information will be returned in the results XML. i.e.

```
<Version>InfixServer (Version: 3.00 Build Date: 13:10:38 Aug 13
2014)</Version>
```

Input

Specifies the PDF to be processed by InfixServer. Can contain the following tags:

File
NewDocument
ResourcePage

File

Specifies the path to an existing PDF to process. If not supplied here can also be specified on the Infix Server command line using the -t command line param.

For Example:

```
<File>InfixServer.pdf</File>
```

NewDocument

Instructs Infix Server to generate a new PDF document and perform any operations specified in the XML Command File on it. It has the following attributes

Width - Width in points of the new PDF

Height - Height in points of the new PDF

NumPages - Number of pages in the new PDF

For Example

```
<NewDocument Width="828" Height="958" NumPages="1"></NewDocument>
```

ResourcePage

Specifies the path to a PDF to be used as a resource PDF. Infix Server will refer to this PDF when performing any operations that involve fonts. The PDF supplied should therefore contain complete definitions of any fonts used within the Infix Server commands.

For Example:

```
<ResourcePage>resource.pdf</ResourcePage>
```

Output

Use this section to specify the PDF to written as a result of Infix Server processing. It has the following sub tags.

- BBox
- File
- Final
- FontCheck
- LineCount
- OutlineFonts
- Scale
- SizeToGalley
- SubsetFonts
- UsePayAsYouGo

File

This is the path to the PDF file to write. If not specified here this can be supplied on the command line using the -o command line parameter.

Scale

Specifies scaling to be applied to all the PDF pages when written. This scale is a percentage of the original PDF page size.

SubsetFonts

Either True or False. If this is True then any fonts used by Infix Server in the PDF will be subsetted when the PDF is written. If this is False then the complete font will be included in the PDF.

SizeToGalley

Size the PDF so that the bottom of its crop box is at the bottom of the text in the galley specified by the attributes **PageNum** and **GalleyNum.** Where GalleyNum specifies the number of the galley on page PageNum. Galleys are numbered from 1 in an undefined order. The following attributes effect the actual height of the resulting PDF.

Multiple – If specified the resulting PDF must be a height that is a multiple of this value.

MinHeight – If specified the resulting PDF must be a height of at least this value. MinHeight will override the Multiple value. If MinHeight is applied then the resulting PDF may not be a multiple of the Multiple attribute.

Gap – If specified there must be a gap of this size between the bottom of the text and the bottom of the PDF. This gap is applied before any Multiple or MinHeight adjustments are made.

The **VerticalAlign** attribute specifies how the text should be aligned vertically if as a result of Multiple or MinHeight adjustment the text no longer fills the galley. This can either be:

Top – Default. Text will be vertically aligned to the top of the galley.

Middle - Text will be vertically aligned to the middle of the galley.

Bottom – Text will be vertically aligned to the bottom of the galley.

Full – Text will be stretched to fit using the parameters specified in the Text->Align Vertical->Full dialog in the Infix family of products.

FullPara - Text will be stretched to fit using the parameters specified in the Text->Align Vertical->Full dialog in the Infix family of products but no adjustment to line leading will be made.

LineCount

Return in the Results XML the number of lines of text in the Galley specified by the attributes **PageNum** and **GalleyNum**. Where GalleyNum specifies the number of the galley on page PageNum. Galleys are numbered from one in an undefined order.

BBox

Return in the Results XML the bounding box of the Page specified by the **PageNum** attribute. The bounding box returned is specified by the **Name** attribute and can be:

CropBox – Default. The crop box defined in the PDF. If not specified the corresponding results tag will be empty.

MediaBox – The Media box defined in the PDF.

BleedBox – The Bleed box defined in the PDF. If not specified the corresponding results tag will be empty.

TrimBox – The Media box defined in the PDF. If not specified the corresponding results tag will be empty.

VisibleBox – The bounding box of all the visible objects on the page. Will attempt to exclude white boxes when determining.

LargestBox – The largest stroked but not filled rectangular path on the page.

LargestClipBox – The largest rectangular clip box on the page that is not bigger than crop box.

AutoTrimBox – Heuristically determined trim box. The heuristics use the crop marks on the page.

AutoBleedBox – Heuristically determined bleed box. The heuristics use the bleed/crop marks on the page.

If not specified the default is “CropBox”

OutlineFonts

Turn all the text in the output PDF into filled vectors. There will be no fonts or editable text in the resulting PDF.

FontCheck

Check fonts in the PDF produced. This tag can have multiple subtags called **Fonts**, which have the following attributes.

StartToken & EndToken – If these attributes are supplied then fonts used in text that starts and ends with these characters will be checked.

Fields – If set to “true” then fonts used in text fields will be checked.

CharsetPDF – Path to a PDF that contains all the characters required to be available in the fonts.

BoldItalic – If set to “true” then bold and italic version of fonts to be checked will also be checked.

If neither StartToken/EndToken nor Fields is supplied then all fonts in the PDF will be

checked.

For each Fonts tag supplied a corresponding tag will appear in the Results XML with the same attributes. The “Fonts” tags will be in the FontCheck tag, which will be in Output tag, i.e.

```
<Output>
  <FontCheck>
     <Fonts StartToken="[" EndToken="]">
        <Font Name="GillSans">
           <Check Name="Embedded" Status="true"></Check>
           <Check Name="ToUnicodeMapping"
Status="partial">CFOLPE+GillSans</Check>
           <Check Name="MissingChars" Status="false"></Check>
           <Check Name="FontOnDisk" Status="false"></Check>
           <Check Name="InvalidWidths" Status="false"></Check>
        </Font>
        <Font Name="GillSans-Light">
           <Check Name="Embedded" Status="true"></Check>
           <Check Name="ToUnicodeMapping"
Status="false">CFOMBE+GillSans-Light, NELNCI+GillSans-
Light</Check>
           <Check Name="MissingChars"
Status="true">EIvRPgTqyXkjzHQKOWJVZYDG!”£$%^&amp;*()_+{}~@?
&gt;&lt;|`-=;’#\/</Check>
           <Check Name="FontOnDisk" Status="false"></Check>
           <Check Name="InvalidWidths" Status="false"></Check>
        </Font>
        <Fields>[fname], [lname], [salutation], [title], [phone],
[address1], [address2], [suburb], [state], [pc], [fax],
[address3], [address4], [suburb2], [state2], [pc2]</Fields>
     </Fonts>
  </FontCheck>
  <Status>OK</Status>
</Output>
```

Each font checked will have a font section with a Name attribute.

Under each **Font** tag will be a number of Check tags, which have the following Names and meanings:

Embedded – Status attribute can be

true - all instances of the font are embedded

false - no instances of this font are embedded. The Check tag will contain the names of all the instances

partial - Some instances embedded some not. The Check tag will contain the names of all the instances not embedded.

ToUnicodeMapping – Status attribute can be

true - all instances of the font have a "to unicode" mappings

false - no instances of this font have "to unicode" mappings. The Check tag will contain the names of all the instances

partial - Some do some don't. The Check tag will contain the names of all the instances that do not contain "to unicode" mappings.

MissingChar – Status attribute can be

true: some characters required not embedded. The Check tag will contain the characters not supplied.

false: all characters required embedded.

FontOnDisk – Status attribute can be

true - Font available locally on disk

false - Font not available locally on disk

InvalidWidths – Status attribute can be

true - Font would appear to have an invalid widths array.

false – The font widths array is OK.

If there were StartToken/EndTokens attributes and/or the Fields attribute was set to true then under the Fonts tag will be a **Fields** tag. This will contain comma-separated list of all the tokenised text and fields found in the PDF. This list should be checked to ensure that all the expected fields are present.

Final

When true, all Iceni-specific information such as change history, field-ids and layout information is removed. This will often have the knock-on effect of reducing the file size.

The default is False for this flag.

```
<Output>
   <Final>True</Final>
</Output>
```

DoOptimise

When False no optimisation is performed when saving the document. This will often result in the size of the document being larger.

The default is True for this flag.

```
<Output>
   <Optimise>False</Optimise>
</Output>
```

UsePayAsYouGo

This allows the pdf to be saved in an unlicensed version of Infixserver without a watermark if a valid pay as you go key is supplied. The attribute **Key** specifies the pay as you go key to be used when saving the document. If this attribute is absent the pay as you go key specified in the configuration file is used.

Infixserver pay as you go keys can be purchased from the Iceni website www.iceni.com.

PostFile

Specifies operations to be performed after all the Infix Server processing including writing any PDF file has been completed. Useful when running in spooler mode to move or delete PDFs that are no longer required. It has the following sub tags.

Move
Delete

Move

Has From and To subtags. The file with path specified in From will be moved to the path specified in To. For Example.

```
<PostFile>
   <Move>
      <From>c:/input/test.pdf</From>
      <To>c:/completed/test.pdf</To>
   </Move>
</PostFile>
```

Delete

Has File subtag. File specifies the path to the PDF to delete. For Example.

```
<PostFile>
   <Delete>
      <File>c:/input/test.pdf</File>
   </Delete>
</PostFile>
```

Commands

This is where you place the main processing actions for each PDF input file.

An example of a complete XML file including a Commands section:

```
<?xml version="1.0" encoding="UTF-8"?>
<InfixServer>
   <Input>
      <File>input.pdf</File>
   </Input>
   <Commands>
      <Command Name="Replace Text">
         <StartPage>1</StartPage>
         <EndPage>LastPage</EndPage>
         <WholeWords>true</WholeWords>
         <MatchCase>true</MatchCase>
         <Replace>
           <From><Text>Baily</Text></From>
           <To><Text>Daily</Text></To>
         </Replace>
      </Command>
   </Commands>
   <Output>
      <File>output.pdf</File>
   </Output>
</InfixServer>
```

Multiple **<Command>** sections may be listed within the **<Commands>** tag. They will be executed in the order in which they are listed.

Add Fields

Add fields/tags to the PDF text. Fields/tags are used to mark text as changeable by Infix Server. The text can be altered by name with the Infix Server "Replace Fielded Text" command. The process of adding fields/tags to PDFs is normally done using the Infix Professional desktop application. This interface is designed to allow other applications access to this mechanism when used in conjunction with the "Export Text" and "Doc Fields" commands.

```
<Command Name="Add Fields" RemoveExisting="[true | false]">
   <Field id="[number]">
      <Col>[hex RGB colour in form rrggbb]</Col>
      <EditType>[Single | Multiple]</EditType>
      <CalloutName>[text]</CalloutName>
      <Control>[None | DeletePara]</Control>
      <Align>[Auto | Left | Centre | Right | Full]</Align>*
      <ReflowMode>[Text | Line | Para]</ReflowMode>
      <Description>[text]</Description>*
      <Span>
         <Start>
           <OID>[number]</OID>
           <Line>[number]</Line>
           <Char>[number]</Char>
         </Start>
         <End>
           <OID>[number]</OID>
           <Line>[number]</Line>
           <Char>[number]</Char>
         </End>
      </Span>
   </Field>
</Command>
```

Multiple **<Field>** definitions can be specified within each "Add Fields" Command. **<Span>** is used to define a text selection upon which the field will be added. The **<OID>** tag is the unique id of a galley/text box which is returned by the "Export Text" command. The **<Line>** tag is a zero based index of the line and the **<Char>** tag is a zero based index of the characters within that line. **<Align>** is only applied if the **<ReflowMode>** is Text or Line. If it is Line then only Left, Centre & Right should be used.

Add Page Annots

adds new annotations to a specific page in the pdf or updates existing annottations on a specific page in the pdf.

```
<Command Name="Add Page Annots">
      <PageNum>[page number]</PageNum>
      <Annot id="[annotation id]>
         [annotation XML definition]
      </Annot>
      <Annot>
         [annotation XML Definition]
      </Annot>
</Command>
```

Each annotation specified by an **<Annot>** tag will be added to **<PageNum>** in the pdf. An annotations definition should match the equivalent annotation definition outputted via the **"Dump annots"** command.

If an **id** attribute is defined then the annotation matching that id will be updated on the page. If this attribute is not present then a new annotation will be added to the page

Page numbers in an annotation definition should be preceded by the string **"ICNPN 22"** this is so page numbers can be matched to page references when creating/updating the annotation.

An example annotation definition for a Link annotation

```
<Annot PageNum="3" id="A">
  <Border>0</Border>
  <Border>0</Border>
  <Border>0</Border>
  <Dest>ICNPN 22</Dest>
  <Dest>/XYZ</Dest>
  <Dest>95.8</Dest>
  <Dest>759.3</Dest>
  <Dest>0</Dest>
  <Rect>105.1</Rect>
  <Rect>80.6</Rect>
  <Rect>486.9</Rect>
  <Rect>93.3</Rect>
  <Subtype>/Link</Subtype>
  <Type>/Annot</Type>
</Annot>
```

The **<Subtype>** tag restricts the annotations output to be of a specific type. The Subtype can be any valid Subtype of annotation as specified in the pdf reference manual.

Annot

Add a new annotation and associated pop-up window to a page.

```
<Command Name="Annot">
   <StartPage>[1 | number | LastPage]</StartPage>*
   <EndPage>[number | LastPage]</EndPage>*
   <RestrictTo>[Odd | Even | All]</RestrictTo>*
   <Type>[Comment | Help | Insert | Key | NewParagraph
          | Note | Paragraph]</Type>*
   <Left>[number]</Left>
   <Top>[number]</Top>
   <Colour col1=[colour] ... col4=[colour] />*
   <Popup>
      <Left>[number]</Left>
      <Top>[number]</Top>
      <Right>[number]</Right> | <Width>[number]</Width>
      <Bottom>[number]</Bottom> | <Height>[number]</Height>
      <Contents>[text]</Contents>*
      <Author>[text]</Author>*
      <Title>[text]</Title>*
   </Popup>*
</Command>
```

The annotation described will be added to every page in the page range specified.

If not supplied, the default colour is yellow.

Multiple Annot commands can be listed within a single **<Commands>** tag.

Argus

Extract the content of a PDF as formatted text.

```
<Command Name="Argus">
      <Config>[Configuration File]</Config>
      <TextPath>[Text Output File]</TextPath>
      <LicenseKey>[Host Key]</LicenseKey>
</Command>
```

This command calls the Argus sub-system within Infix Server to convert the contents of the given PDF into a text and or image format.

All the configuration of the conversion process is done via the contents of the supplied configuration file.

The text part of the conversion is written to a file specified by the **TextPath** tag.

The **LicenseKey** is required to stop the text output being deliberately corrupted with 'Xxxx' characters. This must be an Argus, host-based license key and is distinct from the InfixServer key code.

Crop

Apply a crop box to the PDF pages.

```
<Command Name="Crop">
   <To>[TrimBox | MediaBox | BleedBox | TrimBox | VisibleBox |
LargestBox | LargestClipBox | AutoTrimBox | AutoBleedBox ]
   </To>
   <StartPage>[1 | number | LastPage]</StartPage>*
   <EndPage>[number | LastPage]</EndPage>*
   <RestrictTo>[Odd | Even | All]</RestrictTo>*
</Command>
```

OR

```
<Command Name="Crop">
   <Left>[number]</Left>
   <Right>[number]</Right>
   <Top>[number]</Top>
   <Bottom>[number]</Bottom>
   <StartPage>[1 | number | LastPage]</StartPage>*
   <EndPage>[number | LastPage]</EndPage>*
   <RestrictTo>[Odd | Even | All]</RestrictTo>*
</Command>
```

<To> can be used to specify a bounding box already set in the PDF: "MediaBox", "BleedBox", "TrimBox", "VisibleBox", "LargestBox", "LargestClipBox" or these heuristic methods "AutoTrimBox", "AutoBleedBox" that look for marks on the page.

Delete Page

Delete a range of pages from the document.

```
<Command Name="Delete Page">
   <StartPage>[1 | number | LastPage]</StartPage>*
   <EndPage>[number | LastPage]</EndPage>*
   <RestrictTo>[Odd | Even | All]</RestrictTo>*
</Command>
```

Downsample Images

Reduce the resolution of images within the PDF.

```
<Command Name="Downsample Images">
   <Image type="[Colour | Grey | Mono]">
      <DPI above="[number]">
         <Format>[JPEG | ZIP | CCITT3 | CCITT4]</Format>
         <NewDPI>[number]</NewDPI>
         <JPEGQuality>[number]</JPEGQuality>*
      </DPI>
      <ConvertToGreyScale />*
   </Image>
   <RemoveAllHiddenImageData />*
   <StartPage>[1 | number | LastPage]</StartPage>*
   <EndPage>[number | LastPage]</EndPage>*
   <RestrictTo>[Odd | Even | All]</RestrictTo>*
</Command>
```

Multiple **<Image>** tags can be defined within each “Downsample Images” Command so that Image downsampling can be specified for Colour, Grey and Mono images.

<Format> can be JPEG or ZIP for Colour and Grey Images. It can be ZIP, CCITT3 or CCITT4 for Mono images.

Draw

```
<Command Name="Draw">
   <Type>[Ellipse | Line | Poly | Rect]</Type>*
```

for draw type "Ellipse" & "Rect":

```
   <Left>[number]</Left>
   <Top>[number]</Top>
   <Right>[number]</Right> | <Width>[number]</Width>
   <Bottom>[number]</Bottom> | <Height>[number]</Height>
```

for draw type "Line":

```
   <Point X="[start x]" Y="[start y]" />
   <Point X="[end x]" Y="[end y]" />
```

for draw type "Poly":

```
   <Point X="[start x]" Y="[start y]" />
      ...
   <Point X="[end x]" Y="[end y]" />
   <Close>[true | false]</Close>*
   <StrokeColour col1=[colour] ... col4=[colour] />*
   <FillColour col1=[colour] ... col4=[colour] />*
   <StrokeWidth>[number]</StrokeWidth>

   <StartPage>[1 | number | LastPage]</StartPage>*
   <EndPage>[number | LastPage]</EndPage>*
   <RestrictTo>[Odd | Even | All]</RestrictTo>*
</Command>
```

For draw type "Poly" the **<Close>** tag specifies whether the polygon should be left open or closed.

Omit the **<FillColour>** tag to specify 'no fill'.

Omit the **<StrokeColour>** tag to specify 'no stroke'.

Example: add a small yellow circle with grey outline to every even numbered page:

```
<Command Name="Draw">
   <Type>Ellipse</Type>
   <Left>325</Left>
   <Top>400</Top>
   <Right>425</Right>
   <Bottom>300</Bottom>
   <StrokeColour col1="100" col2="100" col3="100" />
   <FillColour col1="250" col2="250" col3="100" />
   <StrokeWidth>2</StrokeWidth>
   <RestrictTo>Even</RestrictTo>
</Command>
```

Disc Fonts

Generates a catalogue of all fonts held in the given folder on disc.

```
<Command Name="Disc Fonts" ListChars="[true | false]">
   <path>[font folder]</path>
   <RenderFontPreview>
      <path>[preview img path]</path>
      <Width>[number]</Width>
      <Height>[number]</Height>
   </RenderFontPreview>
</Command>
```

Example output:

```
<SystemFonts>
   <path>/Library/Fonts</path>
   <Font>
      <Name>ACaslonPro-Bold</Name>
      <CleanName>ACaslonProBold</CleanName>
      <Family>ACaslanPro</Family>
      <SubFamily>Bold</SubFamily>
      <GlyphCount>3615</GlyphCount>
      <Path>/Library/Fonts/ACaslonPro-Bold.otf</Path>
   </Font>
   <Font>
      <Name>ACaslonPro-BoldItalic</Name>
      <CleanName>ACaslonProBoldItalic</CleanName>
      <Family>ACaslanPro</Family>
      <SubFamily>Bold Italic</SubFamily>
      <GlyphCount>3615</GlyphCount>
      <Path>/Library/Fonts/ACaslonPro-BoldItalic.otf</Path>
   </Font>
</SystemFonts>
```

If ListChars is true:

```
<SystemFonts>
   <path>/Library/Fonts</path>
   <Font>
      <Name>ACaslonPro-Bold</Name>
      <CleanName>ACaslonProBold</CleanName>
      <Family>ACaslanPro</Family>
      <SubFamily>Bold</SubFamily>
      <GlyphCount>3615</GlyphCount>
      <Path>/Library/Fonts/ACaslonPro-Bold.otf</Path>
      <MappedChars>20 180 186 186 189 189 192 192 195 195
197 197 199 19A 19E 19E 1A6 1A6 1B5 1B7 1BF 1C3 1D1 1D6 1DD 1DD
1E2 1E7 1EA 1ED 1F0 1F0 1F4 1F7 1FA 21D 222 22B 22E 22F 232 233
237 237 248 249 250 377 37A 37F 384 38A 38C 38C 38E 3A1 3A3 3CE
3D0 3FF 485 486 591 5C7 5D0 5EA 5F0 5F ....</MappedChars>
   </Font>
   <Font>
      <Name>ACaslonPro-BoldItalic</Name>
      <CleanName>ACaslonProBoldItalic</CleanName>
      <Family>ACaslanPro</Family>
      <SubFamily>Bold Italic</SubFamily>
      <GlyphCount>3615</GlyphCount>
      <Path>/Library/Fonts/ACaslonPro-BoldItalic.otf</Path>
      <MappedChars>20 180 186 186 189 189 192 192 195 195
197 197 199 19A 19E 19E 1A6 1A6 1B5 1B7 1BF 1C3 1D1 1D6 1DD 1DD
1E2 1E7 1EA 1ED 1F0 1F0 1F4 1F7 1FA 21D 222 22B 22E 22F 232 233
237 237 248 249 250 377 37A 37F 384 38A 38C 38C 38E 3A1 3A3 3CE
3D0 3FF 485 486 591 5C7 5D0 5EA 5F0 5F4 ....</MappedChars>
   </Font>
</SystemFonts>
```

Doc Fields

Export details of all the fields/tags within the PDF.

```
<Command Name="Doc Fields">
</Command>
```

The fields/tags defined within the PDF will be added to the Results XML in the following format.

```
<Commands>
...
   <Command Name="Doc Fields">
      <Field Align="[Auto | Left | Centre | Right | Full]"
CalloutName="[text]" Control="[None | DeletePara]"
EditType="[Single | Multiple]" ID="[number]" ReflowMode="[Text |
Line | Para]" col="[hex RGB colour in form
rrggbb]">[text]</Field>
      <Status>OK</Status>
   </Command>
...
</Commands>
```

The **<Field>** tag will be repeated within the “Doc Fields” Command results for each field in the PDF. The contents of the **<Field>** tag will be the field description.

Doc Fonts

Generates a catalogue of all fonts used in the current document.

```
<Command Name="Doc Fonts" subsets="[true | false]"
ListChars="[true | false]"></Command>
```

Example output:

```
<Command Name="Doc Fonts">
   <Font>
      <Name>HelveticaLTStd-Bold</Name>
      <CleanName>HelveticaLTStdBold</CleanName>
      <Family>HelveticaLTStd</Family>
      <Type>Type 1 compact</Type>
   </Font>
   <Font>
      <Name>HelveticaLTStd-Roman</Name>
      <CleanName>HelveticaLTStdRoman</CleanName>
      <Family>HelveticaLTStd</Family>
      <Type>Type 1 compact</Type>
   </Font>
</Command>
```

If **subsets** is true:

```
<Command Name="Doc Fonts">
   <Font>
      <Name>XCFIWB+HelveticaLTStd-Bold</Name>
      <CleanName>HelveticaLTStdBold</CleanName>
      <Family>HelveticaLTStd</Family>
      <GlyphCount>36</GlyphCount>
      <Embedded>True</Embedded>
      <Type>Type 1 compact</Type>
   </Font>
   <Font>
      <Name>XCFIWB+HelveticaLTStd-Roman</Name>
      <CleanName>HelveticaLTStdRoman</CleanName>
      <Family>HelveticaLTStd</Family>
      <GlyphCount>100</GlyphCount>
      <Embedded>True</Embedded>
      <Type>Type 1 compact</Type>
   </Font>
</Command>
```

If **ListChars** is true:

```
<Command Name="Doc Fonts">
   <Font>
      <Name>HelveticaLTStd-Bold</Name>
      <CleanName>HelveticaLTStdBold</CleanName>
      <Family>HelveticaLTStd</Family>
      <Type>Type 1 compact</Type>
      <MappedChars>20 20 39 39 42 47 49 49 4C 4D 50 50 53 53
61 69 6B 77 79 79</MappedChars>
   </Font>
   <Font>
      <Name>HelveticaLTStd-Roman</Name>
      <CleanName>HelveticaLTStdRoman</CleanName>
      <Family>HelveticaLTStd</Family>
      <Type>Type 1 compact</Type>
      <MappedChars>1E 21 25 25 28 29 2C 2E 30 33 35 35 37 3B
40 47 49 50 52 54 56 58 61 79 80 A0</MappedChars>
   </Font>
</Command>
```

Dump Annots

Export details of annotations within the PDF.

```
<Command Name="Dump Annots">
      <Subtype>[subtype]</Subtype>
      <Links></Links>
      <OutputId>[True | False]</OutputId>
      <StartPage>[1 | number | LastPage]</StartPage>*
      <EndPage>[number | LastPage]</EndPage>*
      <RestrictTo>[Odd | Even | All]</RestrictTo>*
</Command>
```

The annotations defined within the PDF will be added to the Results XML a format that is dependant on how their dictionaries are defined in the pdf. Also a page number and a unique id for the annot is output. e.g:

```
      <Commands>
    <Command Name="Dump Annots">
      <Annot PageNum="2" id="A">
        <A>
          <S>/URI</S>
          <Type>/Action</Type>
          <URI>http://www.iceni.com/</URI>
        </A>
        <Border>0</Border>
        <Border>0</Border>
        <Border>0</Border>
        <Rect>195.4</Rect>
        <Rect>587.6</Rect>
        <Rect>260.2</Rect>
        <Rect>599.2</Rect>
        <Subtype>/Link</Subtype>
        <Type>/Annot</Type>
      </Annot>
      <Annot PageNum="3" id="A">
        <Border>0</Border>
        <Border>0</Border>
        <Border>0</Border>
        <Dest>ICNPN 22</Dest>
        <Dest>/XYZ</Dest>
        <Dest>95.8</Dest>
        <Dest>759.3</Dest>
        <Dest>0</Dest>
        <Rect>105.1</Rect>
        <Rect>80.6</Rect>
        <Rect>486.9</Rect>
        <Rect>93.3</Rect>
        <Subtype>/Link</Subtype>
        <Type>/Annot</Type>
      </Annot>
      <Status>OK</Status>
    </Command>
  </Commands>
```

The **<Subtype>** tag restricts the annotations output to be of a specific type. The Subtype can be any valid Subtype of annotation as specified in the pdf reference manual.

The **<Links>** tag restricts the annotations output to be only link annots

The **<OutputId>** tag specifies whether the object id of the annotation is output. If omitted the object id of the annotation is output

Dump All Meta

Outputs all key/value pairs in the document's Info dictionary. Typically members of the Info dictionary are 'Author', 'Subject', 'Title' though any number of custom keys may also exist.

Example:

```
<Command Name="Dump All Meta" [Format=<format>]/>
```

The optional **format** is used for outputting date-based values. The format is provided to the standard POSIX strftime() function. If omitted, the default is "%C"

See **Set All Meta**

Dump Security

Lists the security features that have been applied to the document.

```
<Command Name="Dump Security"/>
```

Example output:

```
<Command Name="Dump Security">
      <Security>
         <Encryption>0:0<Encryption>
         <AllowAccessibility>true</AllowAccessibility>
         <AllowEditing>None</AllowEditing>
         <AllowPrinting>LowQuality</AllowPrinting>
         <MasterPassword>true</MasterPassword>
      </Security>
</Command>
```

This output suggests a document that doesn't allow any editing and has a password to stop the security features being modified.

The presence of an unknown user password would stop processing of the PDF before this command could be executed. In that case an “Invalid User Password” error would be generated.

For an an unencrypted PDF, the <Encryption> value will be 0:0. Any other value means the PDF is encrypted.

Export Text

Export all the text from the PDF with layout information. Handy for grabbing geometry/layout from a page.

```
<Command Name="Export Text"
   ParaByPara="[true | false]"
   LineByLine="[true | false]"
   TextAlignment="[true | false]"
   ParaAttributes="[true | false]">

   <File>[output filename]</File>
   <StartPage>[1 | number | LastPage]</StartPage>*
   <EndPage>[number | LastPage]</EndPage>*
   <RestrictTo>[Odd | Even | All]</RestrictTo>*
</Command>
```

If LineByLine then the output includes geometry for each character and line of text in each galley.

If ParaByPara then only the geometry of galleys and paragraphs is included – a much smaller amount of output.

The text within the PDF will be added to the Results XML in the following format.

If LineByLine and !TextAlignment and !ParaAttributes:

```
<Command Name="Export Text">
   <Page PageNum="1" Left="0" Top="842" Right="595" Bottom="0">
      <TextBox OID="A 8" Left="194.8" Top="113.35" Right="387.74"
Bottom="97.24">
        <Line Left="194.8" Top="113.35" Right="387.74"
Bottom="97.24" FontSize="12" BaseLine="101.35"><f><c
w="8.96">©</c><c w="3.25"> </c><c w="6.64">2</c><c
w="6.68">0</c><c w="6.6">1</c><c w="6.68">0</c><c w="3.94">-
</c><c w="6.65">2</c><c w="6.68">0</c><c w="6.6">1</c><c
w="6.68">4</c><c w="3.26"> </c><c w="2.7">I</c><c
w="7.82">c</c><c w="7.74">e</c><c w="7.31">n</c><c
w="2.39">i</c><c w="3.31"> </c><c w="5.17">T</c><c
w="7.79">e</c><c w="7.75">c</c><c w="7.26">h</c><c
w="7.39">n</c><c w="7.84">o</c><c w="2.35">l</c><c
w="7.85">o</c><c w="8.1">g</c><c w="6.46">y</c><c w="3.28">
</c><c w="5.52">L</c><c w="4.12">t</c><c w="8.16">d</c><c
w="2.94"></c></f></Line>
      </TextBox>
      <TextBox OID="A j" Left="177" Top="308.83" Right="405.51"
Bottom="268.05">
        <Line Left="177" Top="308.83" Right="405.51"
Bottom="268.05" FontSize="28" BaseLine="280.83"><f><c
w="20.22">R</c><c w="15.54">e</c><c w="7.7">f</c><c
w="15.6">e</c><c w="9.32">r</c><c w="15.54">e</c><c
w="15.57">n</c><c w="14">c</c><c w="15.57">e</c><c w="7.7">
</c><c w="23.35">M</c><c w="15.54">a</c><c w="15.57">n</c><c
w="15.57">u</c><c w="15.57">a</c><c w="6.16">l</c><c
w="6.94"></c></f></Line>
      </TextBox>
   </Page>
      </Command>
```

If **TextAlignment** is true:

```
<Command Name="Export Text">
   <Page PageNum="1" Left="0" Top="842" Right="595" Bottom="0">
      <TextBox OID="A-8" Left="194.8" Top="113.35" Right="387.74"
Bottom="97.24">
        <TextAlign Valign="T"></TextAlign>
        <Line Left="194.8" Top="113.35" Right="387.74"
Bottom="97.24" FontSize="12" BaseLine="101.35"><f><c
```

```
w="8.96">©</c><c w="3.25"> </c><c w="6.64">2</c><c
w="6.68">0</c><c w="6.6">1</c><c w="6.68">0</c><c w="3.94">-
</c><c w="6.65">2</c><c w="6.68">0</c><c w="6.6">1</c><c
w="6.68">4</c><c w="3.26"> </c><c w="2.7">I</c><c
w="7.82">c</c><c w="7.74">e</c><c w="7.31">n</c><c
w="2.39">i</c><c w="3.31"> </c><c w="5.17">T</c><c
w="7.79">e</c><c w="7.75">c</c><c w="7.26">h</c><c
w="7.39">n</c><c w="7.84">o</c><c w="2.35">l</c><c
w="7.85">o</c><c w="8.1">g</c><c w="6.46">y</c><c w="3.28">
</c><c w="5.52">L</c><c w="4.12">t</c><c w="8.16">d</c><c
w="2.94">
</c></f></Line>
      </TextBox>
      <TextBox OID="A-j" Left="177" Top="308.83" Right="405.51"
Bottom="268.05">
        <TextAlign Valign="T"></TextAlign>
        <Line Left="177" Top="308.83" Right="405.51"
Bottom="268.05" FontSize="28" BaseLine="280.83"><f><c
w="20.22">R</c><c w="15.54">e</c><c w="7.7">f</c><c
w="15.6">e</c><c w="9.32">r</c><c w="15.54">e</c><c
w="15.57">n</c><c w="14">c</c><c w="15.57">e</c><c w="7.7">
</c><c w="23.35">M</c><c w="15.54">a</c><c w="15.57">n</c><c
w="15.57">u</c><c w="15.57">a</c><c w="6.16">l</c><c w="6.94">
</c></f></Line>
      </TextBox>
      <TextBox OID="A-L" Left="110.2" Top="351.71" Right="478.01"
Bottom="297.22">
        <TextAlign Valign="F">
          <WordSpace min="-50" max="200"></WordSpace>
          <LetterSpace min="-35" max="85"></LetterSpace>
          <LeadingScale min="0.75" max="2"></LeadingScale>
          <SpaceAfter min="0" max="10"></SpaceAfter>
          <FontSizeAdjust min="-3" max="3"></FontSizeAdjust>
          <OnlyFitIfOverSet>1</OnlyFitIfOverSet>
          <IfSpaceAlign>B</IfSpaceAlign>
        </TextAlign>
        <Line Left="110.2" Top="333.22" Right="477.98"
Bottom="278.73" FontSize="36" BaseLine="297.22"><f><c
w="9.97">I</c><c w="21.96">n</c><c w="11.99">f</c><c
w="9.97">i</c><c w="20.02">x</c><c w="9.97"> </c><c
w="23.98">S</c><c w="20.02">e</c><c w="14">r</c><c
w="20.02">v</c><c w="20.02">e</c><c w="14">r</c><c w="9.97">
</c><c w="23.98">V</c><c w="20.02">e</c><c w="14">r</c><c
w="20.02">s</c><c w="9.97">i</c><c w="21.96">o</c><c
w="21.96">n</c><c w="9.97"> </c><c w="20.02">3</c><c w="9">
</c></f></Line>
      </TextBox>
   </Page>
   <Status>OK</Status>
</Command>
```

If **ParaAttributes** is true:

```
<Command Name="Export Text">
   <Page PageNum="1" Left="0" Top="842" Right="595" Bottom="0">
      <TextBox OID="A-8" Left="194.8" Top="113.35" Right="387.74"
Bottom="97.24">
        <Para StartLine="0" EndLine="0">
          <Parent>
            <FontName>DAAAAA+CenturyGothic</FontName>
            <FontSize>00012</FontSize>
            <Leading>1.2</Leading>
            <LeadingMode>Auto</LeadingMode>
            <Name>PStyle3</Name>
            <Type>Para</Type>
          </Parent>
```

```
            <StartIndent>00000</StartIndent>
            <TextAlign>Start</TextAlign>
            <TextIndent>00000</TextIndent>
         </Para>
         <Line Left="194.8" Top="113.35" Right="387.74"
Bottom="97.24" FontSize="12" BaseLine="101.35"><f><c
w="8.96">©</c><c w="3.25"> </c><c w="6.64">2</c><c
w="6.68">0</c><c w="6.6">1</c><c w="6.68">0</c><c w="3.94">-
</c><c w="6.65">2</c><c w="6.68">0</c><c w="6.6">1</c><c
w="6.68">4</c><c w="3.26"> </c><c w="2.7">I</c><c
w="7.82">c</c><c w="7.74">e</c><c w="7.31">n</c><c
w="2.39">i</c><c w="3.31"> </c><c w="5.17">T</c><c
w="7.79">e</c><c w="7.75">c</c><c w="7.26">h</c><c
w="7.39">n</c><c w="7.84">o</c><c w="2.35">l</c><c
w="7.85">o</c><c w="8.1">g</c><c w="6.46">y</c><c w="3.28">
</c><c w="5.52">L</c><c w="4.12">t</c><c w="8.16">d</c><c
w="2.94"></c></f></Line>
      </TextBox>
      <TextBox OID="A-j" Left="177" Top="308.83" Right="405.51"
Bottom="268.05">
         <Para StartLine="0" EndLine="0">
            <Parent>
               <FontName>CAAAAA+ArialMT</FontName>
               <FontSize>00028</FontSize>
               <Leading>1.2</Leading>
               <LeadingMode>Auto</LeadingMode>
               <Name>PStyle2</Name>
               <Type>Para</Type>
            </Parent>
            <StartIndent>00000</StartIndent>
            <TextAlign>Start</TextAlign>
            <TextIndent>00000</TextIndent>
         </Para>
         <Line Left="177" Top="308.83" Right="405.51"
Bottom="268.05" FontSize="28" BaseLine="280.83"><f><c
w="20.22">R</c><c w="15.54">e</c><c w="7.7">f</c><c
w="15.6">e</c><c w="9.32">r</c><c w="15.54">e</c><c
w="15.57">n</c><c w="14">c</c><c w="15.57">e</c><c w="7.7">
</c><c w="23.35">M</c><c w="15.54">a</c><c w="15.57">n</c><c
w="15.57">u</c><c w="15.57">a</c><c w="6.16">l</c><c
w="6.94"></c></f></Line>
      </TextBox>
      <TextBox OID="A-L" Left="110.2" Top="351.71" Right="478.01"
Bottom="297.22">
         <Para StartLine="0" EndLine="0">
            <Parent>
               <FontName>BAAAAA+Arial-BoldMT</FontName>
               <FontSize>00036</FontSize>
               <Leading>1.2</Leading>
               <LeadingMode>Auto</LeadingMode>
               <Name>PStyle1</Name>
               <Type>Para</Type>
            </Parent>
            <StartIndent>00000</StartIndent>
            <TextAlign>Start</TextAlign>
            <TextIndent>00000</TextIndent>
         </Para>
         <Line Left="110.2" Top="333.22" Right="477.98"
Bottom="278.73" FontSize="36" BaseLine="297.22"><f><c
w="9.97">I</c><c w="21.96">n</c><c w="11.99">f</c><c
w="9.97">i</c><c w="20.02">x</c><c w="9.97"> </c><c
w="23.98">S</c><c w="20.02">e</c><c w="14">r</c><c
w="20.02">v</c><c w="20.02">e</c><c w="14">r</c><c w="9.97">
</c><c w="23.98">V</c><c w="20.02">e</c><c w="14">r</c><c
w="20.02">s</c><c w="9.97">i</c><c w="21.96">o</c><c
w="21.96">n</c><c w="9.97"> </c><c w="20.02">3</c><c
```

```
w="9"></c></f></Line>
      </TextBox>
   </Page>
</Command>
```

The **<Page>** tag is repeated for each page in the PDF. The **<TextBox>** is repeated for each text box/galley on the page. The **<Para>** tag is repeated for each paragraph in the galley. The **<Line>** tag is repeated for each line within the text box. The **<f>** tag is used to group characters by field/tag. If the id attribute is missing then the characters contained within the **<f>** tag are not fielded. The **<f>** and **<c>** tags will be repeated within the **<Line>** tag as necessary to represent all of the characters in the line.

If **TextAlignment**=true then the text vertical alignment will be outputted in the same format as the "Export Text Align" command.

If **ParaAttributes**=true then the paragraph attributes will be output for each paragraph in each **TextBox**.

Export Text Align

Export all the texts vertical alignment from the PDF

```
<Command Name="Export Text Align">
   <File>[output filename]</File>
   <StartPage>[1 | number | LastPage]</StartPage>*
   <EndPage>[number | LastPage]</EndPage>*
   <RestrictTo>[Odd | Even | All]</RestrictTo>*
</Command>
```

The text within the PDF's alignment will be added to the Results XML in the following format.

```
<Command Name="Export Text Align">
   <Page PageNum="1">
      <TextAlign OID="A-8" Valign="T"></TextAlign>
      <TextAlign OID="A-j" Valign="T"></TextAlign>
      <TextAlign OID="A-L" Valign="F">
         <WordSpace min="-50" max="200"></WordSpace>
         <LetterSpace min="-35" max="85"></LetterSpace>
         <LeadingScale min="0.75" max="2"></LeadingScale>
         <SpaceAfter min="0" max="10"></SpaceAfter>
         <FontSizeAdjust min="-3" max="3"></FontSizeAdjust>
         <OnlyFitIfOverSet>1</OnlyFitIfOverSet>
         <IfSpaceAlign>B</IfSpaceAlign>
   </TextAlign>
   </Page>
   <Status>OK</Status>
</Command>
```

The **<Page>** tag is repeated for each page in the PDF. The **<TextAlign>** is repeated for each text box/galley on the page. The **Valign** attribute specifies the vertical alignment of the galley, "**T**" means **Top aligned**, "**M**" means **middle aligned**, "**B**" means **bottom aligned** and "**F**" means **full aligned**. If the galley is **Full Aligned** then additional tags specifying the minimum and maximum **word space**, the minimum and maximum **letter spacing**, the minimum and maximum **leading scale**, the minimum and maximum **space after** Paragraph, the minimum and maximum **font size adjustment**, whether you need to **only fit if the text is overset** and if so what **alignment to apply if there is space.**

Extract Page

Extracts a range of pages from the input document and write them to one or more files.

```
<Command Name="Extract Page">
   <File>[output filename]</File>
   <StartPage>[1 | number | LastPage]</StartPage>*
   <EndPage>[number | LastPage]</EndPage>*
   <RestrictTo>[Odd | Even | All]</RestrictTo>*
   <PageNums>[number ]*</PageNums>
</Command>
```

The output filename may include macros. In fact, if macros are not used, each extracted page will be written over the same file resulting in only a single page being output. To avoid this, Infix Server will error if it detects an overwrite.

The following example will extract all pages in a document and write each to a new file with the page number as a suffix.

```
<InfixServer>
    <Input>
        <File>transfix pdfs/Avifaune12.pdf</File>
    </Input>

    <Commands>
        <Command Name="Extract Page">
            <File>split/[FILENAME]-[PAGENUM].pdf</File>
        </Command>
    </Commands>
</InfixServer>
```

If you need to extract a range of pages to the same file, use the **`Delete Page`** command to delete the unwanted pages then save the remaining pages to a new file.

Use **PageNums** array of numbers when the input page numbers should be mapped to different output page numbers.

```
<PageNums>1 3 5 11 19</PageNums>
```

would map input page 1 → 2, 2 → 3, 3 → 5, 4 → 11 and 5 → 19.

Field BBox

Export the bounding boxes of all the fielded/tagged text and images within the PDF.

```
<Command Name="Field BBox"></Command>
```

The Results XML will contain the following

```
<Commands>
   ...
   <Command Name="Field BBox">
      <Page PageNum="[number]">
         <Field Name="[text]">
            <BBox>
               <Top>[float]</Top>
               <Left>[float]</Left>
               <Bottom>[float]</Bottom>
               <Right>[float]</Right>
            </BBox>
         </Field>
         <Image Name="[text]">
            <Top>[float]</Top>
            <Left>[float]</Left>
            <Bottom>[float]</Bottom>
            <Right>[float]</Right>
            <ScaleMode>[CropToFit | FitInside]</ScaleMode>
         </Image>
      </Page>
      <Status>OK</Status>
   </Command>
   ...
</Commands>
```

The **<Page>** tag will be repeated for each page. The **<Field>** tag will be repeated for each fielded/tagged piece of text within the page. The **<BBox>** tag will be repeated for each bounding box that contains that fielded text. The **<Image>** tag will be repeated for each fielded/tagged image within the page.

Font Check

Check the fonts contained within the PDF.

```
     <Command Name="Font Check">
        <Fonts StartToken="[text]"* EndToken="[text]"*
Fields="[true | false]"* CharsetPDF="[file path]"
BoldItalic="[true | false]"* />
     </Command>
```

Check the fonts in the PDF produced. This tag can have multiple subtags called **Fonts**, which have the following attributes.

StartToken & EndToken – If these attributes are supplied then fonts used in text that starts and ends with these characters will be checked.

Fields – If set to "true" then fonts used in text fields will be checked.

CharsetPDF – Path to a PDF that contains all the characters required to be available in the fonts.

BoldItalic – If set to "true" then bold and italic version of fonts to be checked will also be checked.

If neither StartToken/EndToken nor Fields is supplied then all fonts in the PDF will be checked.

For each Fonts tag supplied a corresponding tag will appear in the Results XML with the same attributes. The "Fonts" tags will be in the Font Check tag, which will be in "Font Check" Command xml tag, i.e.

```
<Commands>
   ...
   <Command Name="Font Check">
      <Fonts StartToken="[" EndToken="]">
         <Font Name="GillSans">
            <Check Name="Embedded" Status="true"></Check>
            <Check Name="ToUnicodeMapping"
Status="partial">CFOLPE+GillSans</Check>
            <Check Name="MissingChars" Status="false"></Check>
            <Check Name="FontOnDisk" Status="false"></Check>
            <Check Name="InvalidWidths" Status="false"></Check>
         </Font>
         <Font Name="GillSans-Light">
            <Check Name="Embedded" Status="true"></Check>
            <Check Name="ToUnicodeMapping"
Status="false">CFOMBE+GillSans-Light, NELNCI+GillSans-
Light</Check>
            <Check Name="MissingChars"
Status="true">EIvRPgTqyXkjzHQKOWJVZYDG!”£$%^&amp;*()_+{}~@?
&gt;&lt;|`-=;’#\/</Check>
            <Check Name="FontOnDisk" Status="false"></Check>
            <Check Name="InvalidWidths" Status="false"></Check>
         </Font>
         <Fields>[fname], [lname], [salutation], [title], [phone],
[address1], [address2], [suburb], [state], [pc], [fax],
[address3], [address4], [suburb2], [state2], [pc2]</Fields>
      </Fonts>
      <Status>OK</Status>
   </Command>
   ...
</Commands>
```

Each font checked will have a font section with a Name attribute.

Under each **Font** tag will be a number of Check tags, which have the following Names and meanings:

Embedded – Status attribute can be

true - all instances of the font are embedded

false - no instances of this font are embedded. The Check tag will contain the names of all the instances

partial - Some instances embedded some not. The Check tag will contain the names of all the instances not embedded.

ToUnicodeMapping – Status attribute can be

true - all instances of the font have a "to unicode" mappings

false - no instances of this font have "to unicode" mappings. The Check tag will contain the names of all the instances

partial - Some do some don't. The Check tag will contain the names of all the instances that do not contain "to unicode" mappings.

MissingChar – Status attribute can be

true: some characters required not embedded. The Check tag will contain the characters not supplied.

false: all characters required embedded.

FontOnDisk – Status attribute can be

true - Font available locally on disk

false - Font not available locally on disk

InvalidWidths – Status attribute can be

true - Font would appear to have an invalid widths array.

false – The font widths array is OK.

If there were StartToken/EndTokens attributes and/or the Fields attribute was set to true then under the Fonts tag will be a **Fields** tag. This will contain comma-separated list of all the tokenised text and fields found in the PDF. This list should be checked to ensure that all the expected fields are present.

Generate Fielded XML

Generates a Command XML file that can be altered to easily change fielded/tagged PDFs.

```
<Command Name="Generate Fielded XML">
   <File>[file path]</File>
   <Token Start="[text]" End="[text]" />*
</Command>
```

It is best to describe this function by example. If the following XML Command File is input to Infix Server.

```
<InfixServer>
   <Input>
      <File>CarAd.pdf</File>
   </Input>
   <Commands>
    <Command Name="Generate Fielded XML">
        <Token Start="[" End="]" />
        <File>CarAdGen.xml</File>
    </Command>
   </Commands>
</InfixServer>
```

And the PDF CarAd.pdf has been edited by Infix Pro to contains the text **[title]**, **[price], [telephone]** & **[description]** along with an Image that has been tagged with a name attribute set to **Picture** using one of the Infix family of products. The file CarAd.xml produced will contain

```
<InfixServer>
   <Input>
      <File>CarAd.pdf</File>
   </Input>
   <Commands>
      <Command Name="Replace Fielded Text">
         <Token Start="[" End="]"></Token>
         <Text Token="title" galleyId="5" type="single"></Text>
         <Text Token="description" galleyId="6"
type="multi"></Text>
         <Text Token="telephone" galleyId="10"
type="single"></Text>
         <Text Token="price" galleyId="11" type="single"></Text>
      </Command>
      <Command Name="Replace Named Images">
         <Image Name="Picture"></Image>
      </Command>
   </Commands>
</InfixServer>
```

Infix Server has produced an XML Command File that can easily be altered to perform the text and image replacements required by the PDF. Therefore PDFs can be considered as templates for producing new PDFs.

If Infix Professional was used to add tags/fields to CarAd.pdf with names **title**, **price, telephone** & **description** and the description field was set to have an “Edit Type” of “Multiple Lines” then the file CarAd.xml produced will contain

```
<InfixServer>
   <Input>
      <File>CarAd.pdf</File>
   </Input>
   <Commands>
      <Command Name="Replace Fielded Text">
         <Text Token="title" galleyId="5" type="single">A
Car</Text>
         <Text Token="description" galleyId="6" type="multi"><p
style="text-align: justify"><span style="font-size:8.0pt">On very
rare occasions we come across an original </span><strong><span
style="font-size:8.0pt">DB5 </span></strong><strong><em><span
style="font-size:8.0pt">Auto</span></em></strong><span
style="font-size:8.0pt">, this example is in excellent overall
shape having had much work carried out over the
years.</span></p></Text>
         <Text Token="telephone" galleyId="10" type="single">01603
765477</Text>
         <Text Token="price" galleyId="11"
type="single">£100</Text>
      </Command>
      <Command Name="Replace Named Images">
         <Image Name="Picture"></Image>
      </Command>
   </Commands>
</InfixServer>
```

The content of the Text tags have been filled with the text from the original PDF and in the case of the description field the formatting has also been extracted since it is a multi line field.

Import / Export Bookmarks

Exports the document's bookmark tree.

```
<Command Name="Export Bookmarks">
</Command>
```

A description of all the bookmarks in the document will be stored in the file named in the **<File>** tag.

```
<bookmark
      title=[title]
      page=[page number]
      rect=[left bottom top right]
      fitzoom=[fit zoom]>

         [<bookmark ...></bookmark>]*
</bookmark>
```

Example output:

```
<bookmark title="Contents" page="1"
        rect="0 0 419 595" fitzoom="1 0"></bookmark>
<bookmark title="Quick Start Guide" page="6"
        rect="0 0 419 595" fitzoom="2 0">
   <bookmark title="New features for version 6" page="7"
          rect="0 0 419 595" fitzoom="2 0">
   </bookmark>
</bookmark>
```

The Import Bookmark command expects the bookmarks to be in this format.

Before bookmarks are imported, any pre-existing bookmarks are erased.

Example:

```
<Commands>
   <Command Name="Import Bookmarks">
      <bookmark title="New Contents" page="1"
           rect="0 0 419 595" fitzoom="1 0"></bookmark>
      <bookmark title="New Quick Start Guide" page="6"
           rect="0 0 419 595" fitzoom="2 0">
         <bookmark title="Old features for version 6" page="7"
              rect="0 0 419 595" fitzoom="2 0">
         </bookmark>
      </bookmark>
   </Command>
</Commands>
```

Impose Booklet

Impose a documents pages into a booklet format ready for printing

```
<Command Name="Impose Booklet">
   <Width>[number]</Width>
   <Height>[number]</Height>
   <Duplex/>*
   <PagesPerBooklet>[number]</PagesPerBooklet>
   <Margin>*
      <Top>[number]</Top>*
      <Bottom>[number]</Bottom>*
      <Left>[number]</Left>*
      <Right>[number]</Right>*
      <Border>[number]</Border>*
   </Margin>*
</Command>
```

<Width> and **<Height>** specifies the width and height of the output imposed document.
<Duplex> flips back pages upside down for printing on duplex printers.
If omitted all pages are printed the with the same orientation.

<PagesPerBooklet> specifies number of input pages per output booklet. For example if you have a 32 page document and specify 4 pages per booklet, Infix will produce a document with 16 pages that, when printed, will produce 4 booklets that can be folded separately and then bound together to make a single complete booklet. Values divisible by 4 are advisable for this option.
If omitted a single booklet containing all pages in the input document will be produced

<Margin> specifies the minimum margin around the imposed pages
If omitted, no minimum margins is added.

<Top> , **<Bottom>** , **<Left>** and **<Right>** specify the top, bottom, left and right margins respectively in points.
If omitted, they default to zero.

<Border> specifies the margins border. It's value is the border width in points.
If omitted, no border is added.

Impose Files

Imposes two documents. The resulting document is the combination of both documents with each page containing the first document on the left and the second document on the right

```
<Command Name="Impose Files">
   <File>[filename]</Width>
   <File>[filename]</Height>
</Command>
```

Impose PDF

Impose a documents pages into a new document ready for printing

```
<Command Name="Impose PDF">
   <Width>[number]</Width>
   <Height>[number]</Height>
   <PagesPerSheet>[2 | 4 | 6 | 9 | 16]</PagesPerSheet>
   <Layout>[LeftToRight | RightToLeft | TopToRight |
TopToLeft]</Layout>
   <Margin>*
      <Top>[number]</Top>*
      <Bottom>[number]</Bottom>*
      <Left>[number]</Left>*
      <Right>[number]</Right>*
      <Border>[number]</Border>*
   </Margin>*
</Command>
```

<Width> and **<Height>** specifies the width and height of the output imposed document.
<PagesPerSheet>is the number pages from the input document to be imposed on each page of the output document. This can be either 2, 4, 6, 9 or 16.

<Layout> specifies the imposition layout of the output document.
This can be one of four formats:

- **LeftToRight:** Imposed Pages flow from left to right then top to bottom.
- **RightToLeft:** Imposed Pages flow from right to left then top to bottom.
- **TopToRight:** Imposed Pages flow from top to bottom then left to right.
- **TopToLeft:** Imposed Pages flow from top to bottom then right to left.

<Margin> specifies the minimum margin around the imposed pages
If omitted, no minimum margins is added.

<Top> , **<Bottom>** , **<Left>** and **<Right>** specify the top, bottom, left and right margins respectively in points.
If omitted, they default to zero.

<Border> specifies the margins border. It's value is the border width in points.
If omitted, no border is added.

Insert Page

Add a number of new, blank pages to the current document.

```
<Command Name="Insert Page">
   <Before />*
   <PageNum>[Last | number]</PageNum>*
   <PageCount>[1 | number]</PageCount>*
</Command>
```

<PageNum> specifies the page at which new pages should be inserted.

<Before /> indicates that the new pages should be inserted before page **<PageNum>**. If omitted, the new pages are inserted after page **<PageNum>**.

<PageCount> is the number of new pages that should be inserted.

To insert a page from another PDF:

```
<Command Name="Insert Page">
   <File>[filename]</File>
   <InsertPageNum>[All | number]</InsertPageNum>*
   <PageNum>[Last | number]</PageNum>*
</Command>
```

<PageNum> is the page number in the destination PDF. The default is the last page.

<InsertPageNum> is the page to be inserted from the source PDF. The default is to insert all pages.

Example of merging PDFs together:

```
<Commands>
   <Command Name="Insert Page">
      <File>part1.pdf</File>
   </Command>
   <Command Name="Insert Page">
      <File>part2.pdf</File>
   </Command>
</Commands>
```

Join

Join together PDFs and output the combined PDF.

```
<Command Name="Join" [Sort="false"]>
   <FilePattern>[file path with wildcard chars]</FilePattern>*
   <File>[filename]<File>*
   <Output>[file path]</Output>
</Command>
```

This command works completely independently of the Input PDF and other Commands so should be used in isolation.

<FilePattern> a file path that can contain wildcard characters e.g.
<FilePattern>Infix Server Manual-*.pdf</FilePattern>

<**File**> the explicit name of a single file.

Both <**FilePattern**> and <**File**> can be repeated as needed within the command.

At least one <**File**> or <**FilePattern**> should be present.

The assembled list of filename is sorted alphabetically before the join is executed. To disable the sort, use the optional **Sort='false'** attribute.

Files that match the pattern will be sorted into alphanumeric order before they are joined together.

Example:

```
<InfixServer>
   <Commands>
      <Command Name="Join">
         <File>quickstart6-en-repair2/fontreport.pdf</File>
         <FilePattern>preview*.pdf</FilePattern>
         <Output>/fr-preview.pdf</Output>
      </Command>
   </Commands>
</InfixServer>
```

Optimise

Optimise a PDF and output the optimised PDF.

```
<Command Name="Optimise">
   <Input>[file path]</Input>
   <Output>[file path]</Output>
</Command>
```

This command works completely independently of the Input PDF and other Commands so should be used in isolation.

Num Pages

Returns in the Results XML the number of pages in the PDF.

```
<Command Name="Num Pages"></Command>
```

The Results XML will contain the following.

```
<Commands>
   ...
   <Command Name="Num Pages">
      <NumPages>[number]</NumPages>
      <Status>OK</Status>
   </Command>
   ...
</Commands>
```

Page BBox

Returns in the Results XML the bounding boxes requested from the PDF.

```
<Command Name="Page BBox">
   <StartPage>[1 | number | LastPage]</StartPage>*
   <EndPage>[number | LastPage]</EndPage>*
   <RestrictTo>[Odd | Even | All]</RestrictTo>*
   <BBox Name="[TrimBox | MediaBox | BleedBox | TrimBox |
VisibleBox | LargestBox | LargestClipBox | AutoTrimBox |
AutoBleedBox]"/>
</Command>
```

The <BBox> tag can be repeated for each bounding box required. The **<BBox>** Name attribute can be used to request a bounding box already set in the PDF: "MediaBox", "BleedBox", "TrimBox", "VisibleBox", "LargestBox", "LargestClipBox" or these heuristic methods "AutoTrimBox", "AutoBleedBox" that look for marks on the page.

The Results XML will contain the following.

```
<Commands>
   ...
   <Command Name="Page BBox">
      <Page PageNum="[number]">
         <BBox Name="[TrimBox | MediaBox | BleedBox | TrimBox |
VisibleBox | LargestBox | LargestClipBox | AutoTrimBox |
AutoBleedBox]">
            <Top>[float]</Top>
            <Left>[float]</Left>
            <Bottom>[float]</Bottom>
            <Right>[float]</Right>
         </BBox>
      </Page>
   </Command>
   ...
</Commands>
```

The **<Page>** tag will be repeated for each page requested. The **<BBox>** tag will be repeated for each bounding box requested.

Place Image

Import a image from another file and place it on to pages in the current PDF. The current images supported are JPEG, TIFF, PNG and BMP

```
<Command Name="Place Image">
  <File>[filename]</File>

  <Top>[number]</Top>
  <Left>[number]</Left>
  <Width>[number]</Width>*
  <Height>[number]</Height>*

  <Scale>[None | Fit | Crop | Stretch]</Scale>*

  <StartPage>[1 | number | LastPage]</StartPage>*
  <EndPage>[number | LastPage]</EndPage>*
  <RestrictTo>[Odd | Even | All]</RestrictTo>*
</Command>
```

The page is inserted with a top-left given by the **<Top>** & **<Left>** tags.

If the **<Width>** tag is specified, the **<Height>** tag must also be specified.

Infix Server will try to fit the given page into the destination page at the given top, left coordinates.

If a height & width have been specified, the page will be scaled to fit into the destination rectangle. The type of scaling used in this case is defined by the **<Scale>** tag. For the default scale mode (None), any width & height values will be ignored.

<PlaceBBox> specifies how the bounding box of the new page is determined.

Place PDF

Import a page from another PDF and place it on to pages in the current PDF.

```
<Command Name="Place PDF">
  <File>[filename]</File>
  <Password>[text]</Password>*
  <SrcPageNum>[1 | number | LastPage]</SrcPageNum>*

  <Top>[number]</Top>
  <Left>[number]</Left>
  <Width>[number]</Width>*
  <Height>[number]</Height>*

  <Scale>[None | Fit | Crop | Stretch]</Scale>*
  <PlaceBBox>[Bleed | Crop | Media | Trim]</PlaceBBox>*

  <Rotate>[-270 | -180 | -90 | 0 | 90 | 180 | 270]<Rotate>*

  <UseExistingFonts>[true | false]</UseExistingFonts>*

  <StartPage>[1 | number | LastPage]</StartPage>*
  <EndPage>[number | LastPage]</EndPage>*
  <RestrictTo>[Odd | Even | All]</RestrictTo>*
  <Crop>
     <Left>[number]</Left>
     <Right>[number]</Right>
     <Top>[number]</Top>
     <Bottom>[number]</Bottom>
  </Crop>
</Command>
```

The optional **<Password>** field is used when the PDF being placed requires a password in order to open it.

The page is inserted with a top-left given by the **<Top>** & **<Left>** tags.

If the **<Width>** tag is specified, the **<Height>** tag must also be specified.

Infix Server will try to fit the given page into the destination page at the given top, left coordinates.

If a height & width have been specified, the page will be scaled to fit into the destination rectangle. The type of scaling used in this case is defined by the **<Scale>** tag. For the default scale mode (None), any width & height values will be ignored.

The example XML file **examples/placePDF.xml** shows the results of each of the different scaling modes.

The optional **<Crop>** tag can specify a crop box to crop the placed page after it is placed on the destination page

<PlaceBBox> specifies how the bounding box of the new page is determined.

Place Text

Add a text box containing text to the page. The supplied text is reflowed to fit within the text box.

```
<Command Name="Place Text">
  <Left>[number]</Left>
  <Top>[number]</Top>
  <Right>[number]</Right>
  <Bottom>[number]</Bottom>

  <Contents>[text]</Contents>
  <FontName>[font-name]</FontName>
  <Size>[point-size]</Size>

  <FillColour col1=[colour] ... col4=[colour] />*
  <StrokeColour col1=[colour] ... col4=[colour] />*

  <Align>[Left | Centre | Right | Full]</Align>*

  <StartPage>[1 | number | LastPage]</StartPage>*
  <EndPage>[number | LastPage]</EndPage>*
  <RestrictTo>[Odd | Even | All]</RestrictTo>*
</Command>
```

The first line of text will sit on the specified **<Top>** coordinate. If there is too much text to fit into the text box, the overset text will be hidden just as it would if it were edited using Infix.

Infix Server supports only certain kinds of fonts. To determine which of your system fonts can be used, use the '-f' command line parameter to generate a list of all the compatible fonts Infix Server can find.

Render

Render pages from the current document and output as JPEG, TIFF or PNG images.

```
<Command Name="Render" FileFormat="[PNG | JPG | TIF]">
   <File>[filename]</File>

   <StartPage>[1 | number | LastPage]</StartPage>*
   <EndPage>[number | LastPage]</EndPage>*
   <RestrictTo>[Odd | Even | All]</RestrictTo>*

   <BBox>[BleedBox | CropBox | MediaBox | TrimBox]</BBox>*

   <Patch>
        <Left>[number]</Left>
        <Top>[number]</Top>
        <Right>[number]</Right> | <Width>[number]</Width>
        <Bottom>[number]</Bottom> | <Height>[number]</Height>
   </Patch>*

   <Width>[number]</Width>*
   <Height>[number]</Height>*
   <Stretch />

   <DPI>[number]</DPI>*
   <ColourDepth>[Bitmap | Dither | Gray | RGB]</ColourDepth>*

   <ShowEdits />
</Command>
```

The **<File>** tag specifies the output name of the rendered images and can include macros. This is useful when rendering a range of pages. For example, the following will generate a separate image file for each page rendered:

```
<File>[FILENAME]/page[PAGENUM].jpg</File>
```

<BBox> specifies the bounding box to use when calculating the scale and rendering area of the page. The default is to use the crop box.

<Width> is the maximum width in pixels of the output image.

<Height> is the maximum height in pixels of the output image.

If a width is specified without a height, Infix Server will compute an appropriate output height to ensure the aspect ratio of the image is retained.

If a height is specified without a width, Infix Server will compute an appropriate output width to ensure the aspect ratio of the image is retained.

If both a height and width are specified, Infix Server will compute an appropriate scale to fit the image inside a box of the given width and height.

<Stretch> causes the width and height to be computed without retaining the original aspect ratio.

<DPI> is the resolution value stored in the image - it does not affect the content or size of the image but may effect display if the image viewer respects suggested DPI values.

<ShowEdits> causes modified text to show up red in the rendered output.

<Patch> specifies the area of the page to be rendered. **<BBox>** and **<Patch>** are mutually exclusive – use one or other, or neither.

Replace Fielded Text

Replace fielded/tagged text in the PDF.

```
<Command   Name="Replace Fielded Text"
           Reflow="[Text | Line | Para]"*
           Align="[Left | Right | Center | Full]"*
           >

   <StartPage>[1 | number | LastPage]</StartPage>*
   <EndPage>[number | LastPage]</EndPage>*
   <Token Start="[text]" End="[text]"></Token>*
   <MissingCharFormat>[format]</MissingCharFormat>*

   <Text Token="[text]"
         Reflow="[Text | Line | Para]"*
         Align="[Left | Right | Center | Full]"*
         Repeat="[End | Next | Delete]"*
         >[text]</Text>
</Command>
```

If the **<Token>** tag is defined then any text found that starts with the Start attribute and ends with End attribute will be replaced with the contents of the sub-tag **<Text>** that has a Token attribute equal to the text found between the Start and End attributes. If a suitable Text sub-tag is not defined then the text will be deleted and a warning will appear in the Results XML. When matching PDF text to the Token attribute spaces are ignored. The Text Token attribute should therefore not contain spaces. If Infix Pro has been used to define fields/tags within a PDF then any fielded/tagged text with same name as specified in the Token will be replaced.

For Example

```
<Command Name="Replace Fielded Text">
   <Token Start="[" End="]"></Token>
   <Text Token="Title">A Lovely Car</Text>
   <Text Token="Price">£123,567</Text>
</Command>
```

Will replace the text **[Title]** with **A Lovely Car** and **[Price]** will be replaced with **£123,567**.

Both the Replace **<Command>** tag and its **Text** subtags can have Reflow and Align attributes. The Reflow and Align attributes of the Replace **<Command>** are the default values for all its Text subtags.

The Reflow and Align attributes should not be used if the PDF being processed has been edited using one of the Infix family of products and structure defining galley sizes and paragraph styles has been added to the PDF.

The Reflow attribute can be Text, Line, Para:

Text - only the text will be replaced and nothing else will reflow.
Line - The line will be reflowed to accommodate the text.
Para - The Para will be reflowed to accommodate the text. This is the default.

The Align attribute can be Left, Right, Center, Full (fully justified).

Left - The replaced and reflowed text will be left aligned. This is the default.
Right - The replaced and reflowed text will be right aligned.
Center - The replaced and reflowed text will be centre aligned.
Full - The replaced and reflowed text will be full justified.

For Example

```
<Command Name="Replace Fielded Text" Reflow="Text" Align="Left">
   <Token Start="|" End="|"></Token>
   <Text Token="CustomerName">
      <![CDATA[Iceni Technology Ltd]]>
   </Text>
   <Text Token="Name" Reflow="Text" Align="Right">
      <![CDATA[Simon Crowfoot]]>
   </Text>
</Command>
```

The text **|CustomerName|** will be replaced with the text **Iceni Technology Ltd** and will be left aligned, i.e. the left hand side of the new text will be where the left hand side of the old text was. The text **|CustomerName|** will be replaced with the text **Simon Crowfoot** and will be right aligned, i.e. the right hand side of the new text will be where the right hand side of the old text was.

The Text subtag can also have an attribute called **Repeat**. The Repeat attribute is used to instruct Infix Server to copy the paragraph containing the text specified to either the next paragraph or to the end of the current galley chain.

The Repeat attribute can be:

End - Copy the paragraph that contains this tag to the end of the text before doing the replace.
Next - Copy the paragraph that contains this tag after this paragraph before doing the replace.
Delete - Default behaviour i.e. don't copy para. If a para contains two or more fields the first one should do the Repeat.

Using this attribute it is possible to generate lists in the generated PDF. See the linerepeat.xml example that is in the ExampleXMLCommandFiles folder included as part of the InfixServer distribution. Repeats will only work if there is text after the Fielded or tagged text that is to be repeated in the paragraph.

Controlling Infix Server replacement behaviour from the PDF text.

The text contained within the start and end tokens in the PDF can also control the behaviour of InfixServer. Adding a full stop (.) at the end of the text followed by a single character invokes the following functionality.

d: If this text is not replaced or supplied as empty then delete the paragraph that contained this text, i.e. the text |email.d| will be matched by the following xml.

```
<Command Name="Replace Fielded Text" Reflow="Text" Align="Left">
   <Token Start="|" End="|"></Token>
   <Text Token="email">
      <![CDATA[]]>
   </Text>...
```

But because the replacement is empty the containing paragraph in the PDF will be deleted. This functionality is useful for removing empty lines that would otherwise result from an empty replacement.

l, r, c, f: Force the reflow mode for the replacement to be **Text** and align the new text left (l), right(r), centre(c) or full(f). This method of control will override the values for Reflow and Align specified in the XML.

Formatting replaced text

To insert a paragraph break into the replaced text then '\n' character code 10 decimal can be used. To insert a softbreak then '\r' character code 13 decimal can be used.

If the replaced text is not supplied inside a CDATA tag then the following html like formatting commands are available:

<p> - Mark text as a paragraph. A paragraph break will be inserted at the end. The style attribute can be used to force the justification of the paragraph, i.e. <p style="text-align: justify;">. **left**, **right**, **center** and **justify** (fully) are supported.

**
** - Insert a soft break

**** - Change text to a bold version of the current font if available.

**** - Change text to a italic version of the current font if available.

**** - The style attribute can be used to underline the text with . The size of the font used can be changed using etc. The font size can be specified relatively using xx-large, x-large, large, medium, small, x-small, xx-small. Where medium is the current size. It can also be specified explicitly using <number>pt, i.e. 21pt.

Please be aware that the formatting is part of the Command XML and should therefore be well formed, i.e.

```
<Command Name="Replace Fielded Text">
   <Text Token="description" galleyId="6" type="multi"><p
style="text-align: justify;">On very rare <span style="text-
decoration: underline;">occasions</span> we come across an
original <strong>DB5</strong> <em>Auto. </em></p><p style="text-
align: left;">This example is in <span style="font-size:
large;"><span style="text-decoration:
underline;">excellent</span><br /></span>overall shape having had
much work carried out over the years.</p></Text>
</Command>
```

See also the CarAdFormat.xml example that is in the ExampleXMLCommandFiles folder included as part of the InfixServer distribution.

MissingCharFormat can be used to define what to do when a character in the replacement text cannot be represented in the font used for the field.

If present, the format string is used to convert the character code of the problematic character into a simpler numeric format.

For example, given the missing character FFEE:

"<0x%04d>" → <0xffee>
"&#%d;" → ￮
"&#x%X;" → ￮

If the **MissingCharFormat** tag is not specified, a error occurs whenever an unrepresentable character is encountered.

Replace Hyperlinks

Search for and replace hyperlink destination text throughout a range of pages.

```
<Command Name="Replace Hyperlinks">
  <WholeWords>[true | false]</WholeWords>*
  <UseRegExp>[true | false]</UseRegExp>*
  <MatchCase>[true | false]</MatchCase>*
  <UseWildcards>[true | false]</UseWildcards>*
  <Replace>
     <From>
        <Text>[target text]</Text>
     </From>
     <To>
        <Text>[text]</Text>
     </To>
  </Replace>

  <StartPage>[1 | number | LastPage]</StartPage>*
  <EndPage>[number | LastPage]</EndPage>*
  <RestrictTo>[Odd | Even | All]</RestrictTo>*
</Command>
```

UseWildCards

When enabled, the following wildcards and tokens are available:

? - match a single character
***** - match zero or more characters
[[- literal '['
]] - literal ']'

If the target text includes square brackets, the entire target text is matched but only the text within the brackets is replaced. For example, to remove the first sentence in this paragraph, the search could be:

```
<Replace>
      <UseWildCards>true</UseWildCards>
      <From>
         <Text>[If the target*]For example<Text>
      </From>
      <To>
         <Text></Text>
      </To>
   </Replace>
```

which would match the sentence beginning "If the target..." all the way up to the start of the next sentence "For example" but would only replace the first sentence.

Replace Named Images

Replace named/tagged Images in the PDF.

```
<Command Name="Replace Named Images">
   <Image Name="[text]">[file path]</Image>
</Command>
```

The **<Image>** can be repeated for each image to be replaced.

Each of the **<Image>** subtags of this command will specify the path to an Image file. The image replaced will be the one that is named the same as the Name attribute of the **<Image>** subtag. If the PDF has been edited by Infix Professional and the Image has been tagged with a name attribute then the name attribute will be used. If not then images will be named as follows:

image<page index>-<image index>

Where <page index> is the page number where the image is located. This index starts at 0. <image index> is the index of the image on the page and starts at 0. i.e., if there is one image on the first page then its name will be image0-0.

For Example

```
<Command Name="Replace Named Images">
   <Image Name="image0-0">c:/temp/BlueBMW_4.jpg</Image>
</Command>
```

OR if the image has a name attribute set to **Car** using Infix Professional.

```
<Command Name="Replace Named Images">
   <Image Name="Car">c:/temp/BlueBMW_4.jpg</Image>
</Command>
```

If **[file path]** is omitted, then the named image will be removed.

For example, the following will delete **"**BottomImage"**:**

```
<Command Name="Replace Named Images">
      <Image Name="BottomImage"></Image>
   </Command>
```

Replace Properties

Search for and replace text throughout the document properties

```
<Command Name="Replace Properties">
  <WholeWords>[true | false]</WholeWords>*
  <UseRegExp>[true | false]</UseRegExp>*
  <MatchCase>[true | false]</MatchCase>*
  <UseWildcards>[true | false]</UseWildcards>*
  <Replace>
     <WholeWords>[true | false]</WholeWords>*
     <UseRegExp>[true | false]</UseRegExp>*
     <MatchCase>[true | false]</MatchCase>*
     <UseWildcards>[true | false]</UseWildcards>*
     <From>
        <Text>[target text]</Text>
     </From>
     <To>
        <Text>[text]</Text>
     </To>
  </Replace>
</Command>
```

Multiple replacements can be supplied in one command by repeating <Replace> tag and it's children as required. When a tag can be supplied as a child of both the <Command> tag and the <Replace> tag then the one that is a child of a replace tag will be used if specified for that particular replace. Tags that are children of the <Command> can therefore be used to specify settings to be used across all replacements unless an individual replacement also specifies that setting.

Replace Text

Search for and replace text throughout a range of pages. Searches do not cross paragraph boundaries unless **<MakeSinglePara>** is true.

```
<Command Name="Replace Text">
  <WholeWords>[true | false]</WholeWords>*
  <UseRegExp>[true | false]</UseRegExp>*
  <MatchCase>[true | false]</MatchCase>*
  <UseWildcards>[true | false]</UseWildcards>*
  <Reflow>[Text | Line | Para]</Reflow>*
  <Alignment>[Left | Centre | Right | Full]</Alignment>*
  <Fitting>...</Fitting>*
  <ResetLetterSpacing>[true |false]</ResetLetterSpacing>*
  <MakeSinglePara>[true | false]</MakeSinglePara>*
  <Replaceproperties>[true | false]</Replaceproperties>*

  <Replace>
     <WholeWords>[true | false]</WholeWords>*
     <UseRegExp>[true | false]</UseRegExp>*
     <MatchCase>[true | false]</MatchCase>*
     <UseWildcards>[true | false]</UseWildcards>*
     <Reflow>[Text | Line | Para]</Reflow>*
     <Alignment>[Left | Centre | Right | Full]</Alignment>*
     <Fitting>...</Fitting>*
     <ResetLetterSpacing>[true |false]</ResetLetterSpacing>*
     <Replaceproperties>[true | false]</Replaceproperties>*

     <From>
        <Text>[target text]</Text>
        <FontName>[name]</FontName>*
        <FontSize>[point-size]</FontSize>*
        <FontFamily>[name]</FontFamily>*
        <FontWeight>[[Normal | Bold | Italic |
BoldItalic]]</FontWeight>*
        <FillColour None | col1=[colour] ... col4=[colour] />*
        <StrokeColour None | col1=[colour] ... col4=[colour] />*
     </From>
     <To>
        <Text>[text]</Text>
        <FontName>[name]</FontName>*
        <FontSize>[point-size]</FontSize>*
        <FontFamily>[name]</FontFamily>*
        <FontWeight>[[Normal | Bold | Italic |
BoldItalic]]</FontWeight>*
        <FillColour None | col1=[colour] ... col4=[colour] />*
        <StrokeColour None | col1=[colour] ... col4=[colour] />*
     </To>
  </Replace>

  <StartPage>[1 | number | LastPage]</StartPage>*
  <EndPage>[number | LastPage]</EndPage>*
  <RestrictTo>[Odd | Even | All]</RestrictTo>*
</Command>
```

Multiple replacements can be supplied in one command by repeating <Replace> tag and it's children as required. When a tag can be supplied as a child of both the <Command> tag and the <Replace> tag then the one that is a child of a replace tag will be used if specified for that particular replace. Tags that are children of the <Command> can therefore be used to specify settings to be used across all replacements unless an individual replacement also specifies that setting.

UseWildCards

When enabled the following wildcards and tokens are available in the from text:

? - match a single character
***** - match zero or more characters
[[- literal '['
]] - literal ']'

If the target text includes square brackets, the entire target text is matched but only the text within the brackets is replaced. For example, to remove the first sentence in this paragraph, the search could be:

```
<Replace>
      <UseWildCards>true</UseWildCards>
      <From>
         <Text>[If the target*]For example<Text>
      </From>
      <To>
         <Text></Text>
      </To>
   </Replace>
```

which would match the sentence beginning "If the target..." all the way up to the start of the next sentence "For example" but would only replace the first sentence.
If UseRegExp is enabled then this will be ignored.

UseRegExp

When enabled Perl Compatible Regular Expressions are supported with the following exceptions:

Look-ahead and Look-behind is not supported
The use of ^$ to match the start and end of text is not supported.

Please refer to the Perl Compatible Regular Expressions documentation for more information. Regular expressions will be matched globally. If enabled UseWildCards, MatchCase and WholeWord will be ignored. As well as the **from** text supporting regular expressions the **to** text can contain references to matched elements in the regular expression in the form \0, \1 etc.

MakeSinglePara

If you need to search across paragraph boundaries, set this flag to true. Infix Server will place all the text of a page into a single paragraph and text box. None but the most minimal of reflow (reflow text) will work in this mode. Full reflow would probably cause the page layout to be destroyed.

For example, the following XML locates a legal notice spanning multiple paragraphs on the first page of a document and removes it:

```
<Commands>
   <Command Name="Replace Text">
      <StartPage>1</StartPage>
      <EndPage>1</EndPage>
      <WholeWords>false</WholeWords>
      <MatchCase>true</MatchCase>
      <UseWildcards>true</UseWildcards>
      <Reflow>Text</Reflow>
      <MakeSinglePara>true</MakeSinglePara>
      <Replace>
        <From><Text>[THIS*]IMPORTANT</Text></From>
        <To><Text></Text></To>
      </Replace>
   </Command>
</Commands>
```

Specify no-fill or no-stroke for the **<To>** or **<From>** text by using using the keyword **None**.

```
<FillColour None />
```

Reflow & Alignment

These two settings are the same as is offered in the Find & Replace dialogue inside Infix. As a general rule, to minimise the amount of changes made to the document, use **<Reflow>Text</Reflow>** which only re-positions the characters replaced. This can give rise to overprinted text if the new text requires more space than the old.

Fitting

Infix Server can automatically alter various text metrics to ensure replaced text fits into an existing text box. The auto-fit operation can be controlled using the <Fitting> command. Fitting has no affect on single-line text boxes which are automatically extended to accommodate any new text.

When the **<Fitting>** tag is absent, no fitting is done at all.

```
<Fitting>*
   <Shrink>*
      <FontSize Min=[min]>[true | false]</FontSize>*
      <Leading Min=[min]>[true | false]</Leading>*
   </Shrink>

   <Stretch>*
      <FontSize Max=[max]>[true | false]</FontSize>*
      <Leading Max=[max]>[true | false]</Leading>*
   </Stretch>
</Fitting>
```

The presence of **<Shrink/>** will enable fitting by tracking (character/word spacing) adjustment. The minimum values specified in the optional **<WordSpace>** and **<CharSpace>** tags will be used.

If **<FontSize>** is present, the size of text will be reduced to fit down to the optional minimum specified. This only occurs if the tracking adjustment fails to achieve fitting first.

If **<Leading>** is specified the line-spacing and space-after-paragraphs is reduced down to the optional minimum specified. This only occurs if the tracking adjustment fails to achieve fitting first.

If both **<FontSize>** and **<Leading>** are present, the font size adjustment is tried first.

If after a small reduction, fitting is still not achieved, line spacing then tracks the font size and is reduced in proportion to it until the minimum of both is reached.

Optional minimum values may be specified as a scale factor in the range 0 to 1.0. A factor of 0 is ignored.

The **<Stretch>** behaviour is identical except that scaling is in the opposite direction in order to make the existing text fill more space that it would otherwise. Scaling is only applied to text boxes containing 3 or more lines of text.

Example of gentle fitting, adjusting tracking only, possibly resulting in over or underset text:

```
<Fitting>
   <Shrink/>
   <Stretch/>
</Fitting>
```

Example of aggressive shrinking ensuring text is rarely overset baselines remain unchanged (no leading adjustment):

```
<Fitting>
   <Shrink>
      <FontSize min="0.1">true</FontSize>
   </Shrink>
</Fitting>
```

Example of aggressive shrink and stretch of both font and leading:

```
<Fitting>
   <Shrink>
      <FontSize min="0.1">true</FontSize>
      <Leading min="0.5">true</Leading>
   </Shrink>

   <Stretch>
      <FontSize max="2">true</FontSize>
      <Leading max="1.5">true</Leading>
   </Stretch>
</Fitting>
```

ReplaceProperties

If this tag is present then the Replace will be performed within the document properties too.

Security

Change the security settings for a document.

```
<Command Name="Security">
   <Encryption>[None | Acrobat3 | Acrobat5 | Acrobat 6
               | Acrobat7]</Encryption>
   <Password>[password]</Password>*
   <UserPassword>[password]</UserPassword>*
   <MasterPassword secret=[True |
False]>[password]</MasterPassword>*
   <AllowAll />*
   <AllowAccessibility />*
   <AllowCopying />*
   <AllowPrinting>[LowQuality | HighQuality]</AllowPrinting>*
   <AllowEditing>[Any | Content | Commenting | Assemble
                 | FormFill]</AllowEditing>*
   <AllowSearchMeta />*
</Command>
```

<Password> is the password required to make changes to the document's security settings (or the password required to open the document).

<UserPassword> sets the user password (the one required to view the document). If absent, any user password is removed.

<MasterPassword> sets the password required for making changes to the document's security settings. If absent, the master password is removed from the document. If the optional attribute `secret` is set to `True` then a secret master password is added to the document

If any of the **<Allow*>** tags are omitted, their values are assumed to be **false** or **None**.

Set All Meta

Sets the key/value pairs in the documents Info dictionary. These are typically 'Subject', 'Author' etc but can include any custom key.

Certain keys are read-only such as Created, CreationDate, Producer.

```
<Command Name="Set All Meta">
        <Value Key="Author">Iceni Technology Ltd</Value>
        <Value Key="Creator">Adobe InDesign CS2 (4.0.4)</Value>
        <Value Key="Subject">QtInfix v2.02</Value>
        <Value Key="Title">德□ □公开□学□笔□20170324</Value>
        <Value Key="Trapped">False</Value>
</Command>
```

In the example above, the Creator line will be silently ignored.

See **Dump All Meta**

Set BBox

Modify the various different bounding box dimensions of a range of pages.

```
<Command Name="Set BBox">
   <Type>[CropBox | TrimBox | BleedBox]</Type>*
   <Left>[number]</Left>
   <Top>[number]</Top>
   <Right>[number]</Right>
   <Bottom>[number]</Bottom>
   <StartPage>[1 | number | LastPage]</StartPage>*
   <EndPage>[number | LastPage]</EndPage>*
   <RestrictTo>[Odd | Even | All]</RestrictTo>*
</Command>
```

Set Object BBox

Modify the bounding box of an object by OID.

```
<Command Name="Set Object BBox" OID="[number]">
   <PageNum>[number]</PageNum>*
   <BBox>
      <Left>[number]</Left>
      <Top>[number]</Top>
      <Right>[number]</Right>
      <Bottom>[number]</Bottom>
   </BBox>
</Command>
```

Set Para Attribs

Sets the paragraph alignment attributes of a paragraph of text in the PDF

```
<Command Name="Set Para Attribs" OID="[number]">
   <LineIndex>[number]</LineIndex>*
   <Leading>[number]</Leading>*
   <LeadingMode>[AtLeast | Exactly | Auto]</LeadingMode>*
   <SpaceAfter>[number]</SpaceAfter>*
   <TextAlign>[Start | Center | End | Justify |
Auto]</TextAlign>*
   <TextIndent>[number]</TextIndent>*
   <StartIndent>[number]</StartIndent>*
   <EndIndent>[number]</EndIndent>*
</Command>
```

The paragraph specified by **OID** and **LineIndex** will have it's attribute updated to reflect the attributes specified by the tags in the XML.

All of the tags are optional.

The **<Leading>** tag sets the line space of the text in the paragraph.

The **<LeadingMode>** tag sets the mode for the line space in the paragraph.

- **AtLeast**: The current line space is the minimum line space used.
- **Exactly**: The current line space is always used.
- **Auto**: The line space is calculated using the font sizes in the paragraph.

The **<SpaceAfter>** tag sets the space after the paragraph.

The **<TextAlign>** tag set the horizontal alignment of the text in the paragraph.

- **Start**: Left aligned.
- **Center**: Center aligned.
- **End**: Right aligned.
- **Justify**: Full aligned.

The **<TextIndent>** tag sets the left indent of the text in the paragraph.

The **<StartIndent>** tag overrides the left indent of the text and sets the left indent of the first line of text in the paragraph.

The **<EndIndent>** tag sets the right indent of the last line of text in the paragraph.

Set Text Align

Sets the vertical alignment of a paragraph of text in the PDF

```
<Command Name="Set Text Align" OID="[number]">
   <Valign>[T | M | B | F]</Valign>*
   <WordSpace min=”[number] max=”[number]”></WordSpace>*
   <LetterSpace min=”[number] max=”[number]”></LetterSpace>*
   <LeadingScale min=”[number] max=”[number]”></LeadingScale>*
   <SpaceAfter min=”[number] max=”[number]”></SpaceAfter>*
   <FontSizeAdjust min=”[number]
max=”[number]”></FontSizeAdjust>*
   <OnlyFitIfOverSet>[1 | 0]</OnlyFitIfOverSet>*
   <IfSpaceAlign>[T | M | B]</IfSpaceAlign>*
</Command>
```

The text specified by **OID** will have it's vertical alignment set to **Valign**.

T = Top Align.

M = Middle Align.

B = Bottom Align.

F = Full Align.

The fitting limits **WordSpace**, **LetterSpace**, **LeadingScale**, **SpaceAfter**, **FontSizeAdjust**, **OnlyFitIfOverSet** and **IfSpaceAlign** are only applied if **Valign** is set to '**F**'.

IfSpaceAlign is only applied if **OnlyFitIfOverSet** is set to **1**.

Translate Import

Imports translated XML back into a PDF from which XML has previously been exported via the **Translate Export** command.

```
<Input>
<File>[Input PDF]</File>
</Input>
```

```
<Command Name="Translate Import">
   <File>[Input XML]</File>
   <AutoFit />*
   <Fitting>...</Fitting>*
   <HyphenationDictionary>[Hyphen Dict]</HyphenationDictionary>
   <Substitute font=[Document Font]>[Disc Font]</Substitute>*
   <ResetLetterSpacing />*
   <TransXML>[trans xml config file]</TransXML>*
</Command>
```

The Input PDF must be one that has been produced from the execution of a previous **Translate Export** command. It will have been tagged so that the location of the original text to which each piece of XML corresponds can be determined

During import, InfixServer may hyphenate words to achieve better layout. It is therefor essential that the correct hyphenation dictionary is specified in **[Hyphen Dict]** to match the language contained in the XML being imported. The available hyphenation dictionaries are stored in the 'hyphenation' folder in the InfixServer folder.

Imported text that is too long to fit into the original space will become overset. You can avoid this by enabling **<AutoFit />** which will attempt to adjust the spacing and size of the text in order to make it fit. It may not work in some extreme cases so a check for overset text should be routinely performed on PDF resulting from this command.

The optional **<Fitting>** section can be used to specify fitting behavior and is the same as that used in ReplaceText. See Fitting on page 75.

In most cases (unless using auto-fit), the character spacing used in the original text is maintained after import. Use **<ResetLetterSpacing />** to reset this spacing to 0 – the default for new text.

<Substitute> causes Infix Server to use a font from disc whenever the named in the ***font*** tag is encountered. This is useful when the document's existing fonts do not contain all of the characters required by the translated text being imported.

This tag may be repeated as many times as necessary to cater for any number of substitutions.

The optional **<TransXML>** tag specifies the configuration file to use. This file dictates the XML tags that are expected and should not generally be edited by the user. If not specified, the setting is taken from the main `infixserver.cfg` file.

It is important to use the same **<TransXML>** file for both export and input.

Translate Export

Exports the contents of the Input PDF as XML. Once exported, the XML can be translated by 3rd party tools and then re-imported into the PDF using the associated **Translate Import** command.

```
<Input>
<File>[Input PDF]</File>
</Input>

   <Command Name="Translate Export">
      <File>[filename]</File>
      <MarkedContent />*
      <HyphenationDictionary>[Hyphen Dict]</HyphenationDictionary>
      <StartPage>[1 | number | LastPage]</StartPage>*
      <EndPage>[number | LastPage]</EndPage>*
      <TransXML>[trans xml config file]</TransXML>*
   </Command>

<Output>
      <File>[Output PDF]</File>
</Output>
```

If **<MarkedContent />** is present, only text marked for export will be included in the export. The facility to mark-for-export is available in the Infix desktop editor available from Iceni's website.

During export, InfixServer tags the PDF to associate each piece of XML with its original text in the document. Once complete, this tagged version of the PDF will be saved as [Output PDF]. When it comes time to import new XML into the document (using **Translate import**), it must be imported into this tagged version, not the original PDF.

The optional **<TransXML>** tag specifies the configuration file to use when exporting. This file dictates the XML tags that are output and should not generally be edited by the user. There may however be multiple files installed with the InfixServer system in which case, you may specify one of them here. If not specified, the setting is taken from the main `infixserver.cfg` file.

Update Annots

Update details of annotations within the PDF or add new annotations to the pdf.

```
<Command Name="Update Annots">
      <Annot PageNum="[page number]" id="[annotation id]"*>
         [annotation XML definition]
      </Annot>
      <Annot>
         [annotation XML Definition]
      </Annot>
</Command>
```

Each annotation specified by an **<Annot>** tag will be added to the pdf. An annotations definition should match the equivalent annotation definition outputted via the **"Dump annots"** command.

If an **id** attribute is defined then the annotation matching that id will be updated on the page. If this attribute is not present then a new annotation will be added to the page

Page numbers in an annotation definition should be preceded by the string **"ICNPN 22"** this is so page numbers can be matched to page references when creating/updating the annotation.

An example annotation definition for a Link annotation

```
<Annot PageNum="3" id="A">
  <Border>0</Border>
  <Border>0</Border>
  <Border>0</Border>
  <Dest>ICNPN 22</Dest>
  <Dest>/XYZ</Dest>
  <Dest>95.8</Dest>
  <Dest>759.3</Dest>
  <Dest>0</Dest>
  <Rect>105.1</Rect>
  <Rect>80.6</Rect>
  <Rect>486.9</Rect>
  <Rect>93.3</Rect>
  <Subtype>/Link</Subtype>
  <Type>/Annot</Type>
</Annot>
```

The **<Subtype>** tag restricts the annotations output to be of a specific type. The Subtype can be any valid Subtype of annotation as specified in the pdf reference manual.

Version

Returns the Infix Server version information in the Results XML.

```
<Command Name="Version"></Command>
```

The Results XML will contain.

```
<Commands>
   …
   <Command Name="Version">
      <Version>[float]</Version>
      <BuildDate>[time and date]</BuildDate>
   </Command>
   ...
</Commands>
```

Watermark

Add a watermark to a range of pages. Places a page from the input PDF in front of or behind the existing page contents of the current PDF.

```
<Command Name="Watermark">
   <File>[filename]</File>
   <Password>[text]</Password>*
   <SrcPageNum>[1 | number]</SrcPageNum>*

   <PlaceAtBack />*
   <FitToPage />*
   <Opacity>[percentage]</Opacity>*

   <StartPage>[1 | number | LastPage]</StartPage>*
   <EndPage>[number | LastPage]</EndPage>*
   <RestrictTo>[Odd | Even | All]</RestrictTo>*
</Command>
```

<filename> is the watermark PDF to be used.

If **<PlaceAtBack>** is omitted, the watermark is placed over existing objects on the page(s).

If **<FitToPage>** is omitted, no scaling is done of the watermark contents, otherwise they are resized (isomorphically) to fit the destination page(s).

<Opacity> is used to dictate the density of the placed watermark with 100% being completely opaque. If absent, the value is assumed to be 100%.

<Password> is required if the document to be watermarked is password protected.

The Results XML

Infix Server will write the results of any operation it performs in XML format. When Infix Server is not run as a spooler the XML will be written to stdout if the command line parameter -l is not supplied. When running as a spooler the results xml will be written to a file in the Output Dir called results_<XML Command File Name>.xml.

The top-level tag in the XML results will be InfixServerResults. It will have the following subtags

Args
Config
LogFile
LicenseKey
RunwayInit
FontPaths
XMLFileLoad

Plus subtags which are the same as any tags specified under the InfixServer tag in the XML Command File being processed.

Each of these tags and their subtags should be checked for any warnings or errors that may have occurred. If everything is OK then they will contain

```
<Status>OK<Status>
```

When a warning occurs then the tag whose operation caused a warning will contain

```
<Status>Warning</Status>
<Detail Code="|warning code|">|Description|</Detail>
```

Where |warning code| is a number from 1 to 5 and |Description| is a description of the warning. The warning codes are defined as

1 - Text has been overset as a result of the operation
2 - Config file is for an older version of Infix Server
3 - A value is missing from the XML Command File.
4 - The license key is invalid.
5 - PDF not the same size as the rectangle it is to be placed in.

Infix Server will continue processing the XML Command File after a warning.

When an error occurs then the tag whose operation caused an error will contain

```
<Status>Error</Status>
<Detail Code="|error code|">|Description|</Detail>
```

Where |error code| is a number whose meaning is described below and |Description| is a description of the error.

Infix Server will stop processing the XML Command File when an error occurs.

Although there is no detailed explanation of all the possible error conditions, the following is a list of mnemonics used internally by Iceni to define the error codes. This list may be useful in producing some text explanation of the error.

0	kNoError	20	kStackOverflow
1	kBuffTooSmall	21	kStackUnderflow
2	kBadlyFormedExpression	22	kWriteFailed
3	kNoSuchObject	23	kBadNozzle
4	kOutOfMemory	24	kInvalidArgument
5	kOpenFailed	25	kNotInScope
6	kBadFilename	26	kNotAvailable
7	kTooManyArguments	27	kToolkitNotReady
8	kMissingQuote	28	kLimitCheck
9	kUnknownMacro	29	kMismatch
10	kMissingArguments	30	kAtomicObject
11	kReadFailed	31	kUserCancel
12	kSyntaxError	32	kSystemError
13	kFileNameTooLong	33	kUnmatchedMark
14	kUnexpectedFormat	34	kMissingKey
15	kDoesNotExist	35	kTypeCheck
16	kCreateFailed	36	kParsingError
17	kInvalidImage	37	kInvalidDongle
18	kAcroError	38	kEndOfData
19	kOutOfRange		

Deprecated Sections

The following sections are still available for use in legacy installations though their use for future tasks is not recommended. All of the functionality they used to provide is now provided by equivalents entries in the Command section.

- Document
- Manifest
- Data
- DocumentInfo
- Render

Data *(Deprecated)*

Specifies text or images to be replaced in the PDF. It can have the following sub tags.

ImageReplace
Replace
File

ImageReplace

Each of the Image subtags of this tag will specify the path to an Image file. The image replaced will be the one that is named the same as the Name attribute of the Image subtag. If the PDF has edited by one of the Infix product family that has added structure to the PDF and a name attribute has been set on the image then the name attribute will be used. If not then images will be named as follows:

image<page index>-<image index>

Where <page index> is the page number where the image is located. This index starts at 0. <image index> is the index of the image on the page and starts at 0. i.e., if there is one image on the first page then its name will be image0-0.

For Example

```
<ImageReplace>
   <Image Name="image0-0">c:/temp/BlueBMW_4.jpg</Image>
</ImageReplace>
```

OR if the image has a name attribute set to **Car** using one of the Infix product family

```
<ImageReplace>
   <Image Name="Car">c:/temp/BlueBMW_4.jpg</Image>
</ImageReplace>
```

Replace

Use this tag to replace text in a PDF document.

Replace operates in two modes. If the attributes StartToken and EndToken are not used then it will perform a straight replace of text supplied in the subtag **From** to the text specified in the subtag **To** across the PDF document.

For Example

```
<Replace>
   <From>Cat</From>
   <To>Dog</To>
</Replace>
```

Will replace all instances of the text **Cat** with **Dog** in the PDF.

If the StartToken and EndToken attributes are defined then any text found that starts with StartToken and ends with EndToken will be replaced with the contents of the subtag Text that has a Token attribute equal to the text found between the Start and End Token. If a suitable Text subtag is not defined then the text will be deleted and a warning will appear in the Results XML. When matching PDF text to the Token attribute spaces are ignored. The Text Token attribute should therefore not contain spaces. If Infix Pro has been used to define fields within a PDF then any fielded text with same name as specified in the Token will also be replaced.

For Example

```
<Replace StartToken="[" EndToken="]">
   <Text Token="Title">A Lovely Car</Text>
   <Text Token="Price">£123,567</Text>
</Replace>
```

Will replace the text **[Title]** with **A Lovely Car** and **[Price]** will be replaced with **£123,567**.

Both the Replace tag and its Text subtags can have Reflow and Align attributes. The Reflow and Align attributes of the Replace Tag are the default values for all its Text subtags.

The Reflow and Align attributes should not be used if the PDF being processed has been edited using one of the Infix family of products and structure defining galley sizes and paragraph styles has been added to the PDF.

The Reflow attribute can be Text, Line, Para:

Text - only the text will be replaced and nothing else will reflow.
Line - The line will be reflowed to accommodate the text.
Para - The Para will be reflowed to accommodate the text. This is the default.

The Align attribute can be Left, Right, Center, Full (fully justified).

Left - The replaced and reflowed text will be left aligned. This is the default.
Right - The replaced and reflowed text will be right aligned.
Center - The replaced and reflowed text will be centre aligned.
Full - The replaced and reflowed text will be full justified.

For Example

```
<Replace StartToken="|" EndToken="|" Reflow="Text" Align="Left">
   <Text Token="CustomerName">
      <![CDATA[Iceni Technology Ltd]]>
   </Text>
   <Text Token="Name" Reflow="Text" Align="Right">
      <![CDATA[Simon Crowfoot]]>
   </Text>
</Replace>
```

The text **|CustomerName|** will be replaced with the text **Iceni Technology Ltd** and will be left aligned, i.e. the left hand side of the new text will be where the left hand side of the old text was. The text **|CustomerName|** will be replaced with the text **Simon Crowfoot** and will be right aligned, i.e. the right hand side of the new text will be where the right hand side of the old text was.

The Text subtag can also have an attribute called **Repeat**. The Repeat attribute is used to instruct Infix Server to copy the paragraph containing the text specified to either the next paragraph or to the end of the current galley chain.

The Repeat attribute can be:

End - Copy the paragraph that contains this tag to the end of the text before doing the replace.
Next - Copy the paragraph that contains this tag after this paragraph before doing the replace.
Delete - Default behaviour i.e. don't copy para. If a para contains two or more fields the first one should do the Repeat.

Using this attribute it is possible to generate lists in the generated PDF. See the linerepeat.xml example that is in the ExampleXMLCommandFiles folder included as part of the InfixServer distribution. Repeats will only work if there is text after the Fielded or tagged text that is to be repeated in the paragraph.

Controlling Infix Server replacement behaviour from the PDF text.

The text contained within the start and end tokens in the PDF can also control the behaviour of InfixServer. Adding a full stop (.) at the end of the text followed by a single character invokes the following functionality.

d: If this text is not replaced or supplied as empty then delete the paragraph that contained this text, i.e. the text |email.d| will be matched by the following xml.

```
<Replace StartToken="|" EndToken="|" Reflow="Text" Align="Left">
   <Text Token="email">
      <![CDATA[]]>
   </Text>...
```

But because the replacement is empty the containing paragraph in the PDF will be deleted. This functionality is useful for removing empty lines that would otherwise result from an empty replacement.

l, r, c, f: Force the reflow mode for the replacement to be **Text** and align the new text left (l), right(r), centre(c) or full(f). This method of control will override the values for Reflow and Align specified in the XML.

Formatting replaced text

To insert a paragraph break into the replaced text then '\n' character code 10 decimal can be used. To insert a softbreak then '\r' character code 13 decimal can be used.

If the replaced text is not supplied inside a CDATA tag then the following html like formatting commands are available:

<p> - Mark text as a paragraph. A paragraph break will be inserted at the end. The style attribute can be used to force the justification of the paragraph, i.e. <p style="text-align: justify;">. **left**, **right**, **center** and **justify** (fully) are supported.

**
** - Insert a soft break

**** - Change text to a bold version of the current font if available.

**** - Change text to a italic version of the current font if available.

**** - The style attribute can be used to underline the text with . The size of the font used can be changed using etc. The font size can be specified relatively using xx-large, x-large, large, medium, small, x-small, xx-small. Where medium is the current size. It can also be specified explicitly using <number>pt, i.e. 21pt.

Please be aware that the formatting is part of the Command XML and should therefore be well formed, i.e.

```
<Replace StartToken="[" EndToken="]">
<p style="text-align: justify;">On very rare <span style="text-
decoration: underline;">occasions</span> we come across an
original <strong>DB5</strong> <em>Auto. </em></p><p style="text-
align: left;">This example is in <span style="font-size:
large;"><span style="text-decoration:
underline;">excellent</span><br /></span>overall shape having had
much work carried out over the years.</p>
</Replace>
```

See also the CarAdFormat.xml example that is in the ExampleXMLCommandFiles folder included as part of the InfixServer distribution.

File

If supplied then the contents of this tag will be the path to where an XML Command

File will be written by Infix Server. It is best to describe this function by example. If the following XML Command File is input to Infix Server.

```
<InfixServer>
   <Input>
      <File>CarAd.pdf</File>
   </Input>
   <Data>
      <Replace StartToken="[" EndToken="]">
      </Replace>
      <ImageReplace>
      </ImageReplace>
      <File>CarAd.xml</File>
   </Data>
</InfixServer>
```

And the PDF CarAd.pdf contains the text **[title]**, **[price]** & **[description]** along with an Image that has had its image name attribute set to **Picture** using one of the Infix family of products. The file CarAd.xml produced will contain

```
<?xml version="1.0"?>
<InfixServer>
   <Input>
      <File>CarAd.pdf</File>
   </Input>
   <Data>
      <Replace StartToken="[" EndToken="]" Token="True">
         <Text Token="title" type="single"></Text>
         <Text Token="description" type="multi"></Text>
         <Text Token="price" type="single"></Text>
      </Replace>
      <ImageReplace>
         <Image Name="Picture"></Image>
      </ImageReplace>
   </Data>
</InfixServer>
```

Infix Server has produced an XML Command File that can easily be altered to perform the text and image replacements required by the PDF. Therefore PDFs can be considered as templates for producing new PDFs.

If Infix Pro was used to add fields to CarAd.pdf with names **title**, **price** & **description** and the description field was set to have an "Edit Type" of "Multiple Lines" then the file CarAd.xml produced will contain

```
<?xml version="1.0"?>
<InfixServer>
   <Input>
      <File>CarAd.pdf</File>
   </Input>
   <Data>
      <Replace StartToken="[" EndToken="]" Token="True">
         <Text Token="title" type="single">1964 Aston Martin
DB5</Text>
         <Text Token="description" type="multi"><p style="text-
align: left">On very rare occasions we come across an original
<strong>DB5 <em>Auto</em></strong>, this example is in
<em>excellent</em> overall shape having had much work carried out
over the years.</p></Text>
         <Text Token="price" type="single">£119,950</Text>
      </Replace>
      <ImageReplace>
         <Image Name="Picture"></Image>
      </ImageReplace>
   </Data>
```

```
</InfixServer>
```

The content of the Text fields has been filled with the text from the original PDF and in the case of the description field the formatting has also been extracted since it is a multi line field.

Document *(Deprecated)*

The Document tag specifies commands that are to be performed at a document level. It can contain the following tags

DeletePage
PDFInsert
AddNewPage
SetBBox

The commands specified in these tags will be performed in the order indicated by the above list regardless of the order that they appear in the XML.

DeletePage

Instructs Infix Server to delete the page specified by its sub tag PageNum.

For Example

```
<DeletePage Name="DeletePage 1">
   <PageNum>1</PageNum>
</DeletePage>
```

Will delete the first page.

The Name attribute specifies an attribute that is replicated in the corresponding tag in the Results XML.

PDFInsert

Instructs Infix Server to insert a PDF. It can have the following sub tags

File - path to the PDF to insert.

PageNum - Number of the page into which the PDF will insert. This page number will become the first page of the inserted PDF. If not specified then insert PDF will be inserted after the last page.

InsertPageNum - Page number of the page from the PDF to be inserted. If not specified then all the PDF is inserted.

For Example

```
<PDFInsert Name="test.pdf">
   <File>test.pdf</File>
   <PageNum>1</PageNum>
   <InsertPageNum>3</InsertPageNum>
</PDFInsert>
```

The Name attribute specifies an attribute that is replicated in the corresponding tag in the Results XML.

AddNewPage

Instructs Infix Server to add a new blank page to the PDF. The new page will be the same size as page 1 of the PDF that it is being added to. It can have the following sub tags

PageNum - Page number. Default if not supplied will be 1.

After - If supplied page will be added after the page specified in PageNum. Otherwise it will be added before.

Last - If supplied indicates that the page should be added at the end. PageNum and After will be ignored.

For Example

```
<AddNewPage Name=”Add new page 3”>
   <PageNum>2<PageNum>
   <After></After>
</AddNewPage>
```

The Name attribute specifies an attribute that is replicated in the corresponding tag in the Results XML.

SetBBox

Set the crop, bleed or trim box on the specified page in the PDF. It can have the following sub tags. Multiple BBox may be set using this tag repeatedly.

Name – Either “Crop”, “Bleed” or “Trim”
PageNum - Page number. Default if not supplied will be 1.
Left – Left hand coord of the bbox.
Right – Right hand coord of the bbox.
Top – Top coord of the bbox.
Bottom – Bottom coord of the bbox.

DocumentInfo *(Deprecated)*

Used to return information in the Results XML about the PDF that has been processed by Infix Server. It can have the following sub tags.

PageSize
NumPages
ImageBBoxes

PageSize

Return the size of the crop box and media box of a PDF page. The number of the page is specified using the <PageNum> sub tag. The Results XML will contain XML similar to the following within the InfixServerResults tag.

```
<DocumentInfo>
   <Page>
      <PageNum>1</PageNum>
      <MediaBox>
         <Top>792.0000</Top>
         <Left>0.0000</Left>
         <Bottom>0.0000</Bottom>
         <Right>612.0000</Right>
         <Height>792.00</Height>
         <Width>612.00</Width>
      </MediaBox>
      <CropBox>
         <Top>792.0000</Top>
         <Left>0.0000</Left>
         <Bottom>0.0000</Bottom>
         <Right>612.0000</Right>
         <Height>792.00</Height>
         <Width>612.00</Width>
      </CropBox>
   </Page>
</DocumentInfo>
```

NumPages

Return the number of pages in the PDF.

ImageBBoxes

Return the bounding boxes of all the images in the PDF that have been given a name attribute by one of the Infix family of products. The Results XML will contain XML similar to the following within the InfixServerResults tag.

```
<DocumentInfo>
      <ImageBBoxes>
      <Image PageNum="1" Name="Picture">
         <Top>821.5541</Top>
         <Left>25.4268</Left>
         <Bottom>750.1870</Bottom>
         <Right>138.1537</Right>
      </Image>
   </ImageBBoxes>
</DocumentInfo>
```

Manifest *(Deprecated)*

The Manifest tag specifies commands that are to be performed at a page level. It can contain the following tags:

PDFPlace
DrawRect
Text

PDFPlace

Allows a PDF to be placed on a PDF page. It can have the following sub tags:

File - Path to the PDF to be placed.

PageNumIn - The page number of the page to be placed from the PDF indicated by File. Will default to 1 if not supplied.

PageNumOut - The Page number on which the PDF is to be placed. Will default to 1 if not supplied.

X & Y - Coords in points of the top left corner of where the PDF should be placed.

Width & Height - The width and height in points of the area into which the PDF should be placed. The Scale tag dictates how the PDF will be scaled. If Width and Height are not specified then the PDF will be placed at original size, i.e. unscaled.

Scale - Ignored if Width and Height are not supplied.
Can be Fit (Default), Crop or Stretch. Fit means that the PDF will be resized but not distorted to fit the rectangle by padding with white space. Crop means that the PDF will be resized but not distorted to the fit the rectangle by cropping it where necessary. Stretch will stretch the PDF to fit the space exactly and it may be distorted.

XTolerance & YTolerance - Infix Server will error when scaling if either the Width or Height has to be changed by more than XTolerance or YTolerance percent of the original width and height.

Rotate - Can be 90, 180 or 270. The PDF will be rotated clockwise by 90, 180 or 270 degrees before scaling and placing.

PlaceBBox – The PDF bounding box whose top left coord after any rotation and scaling should become **X, Y** when the PDF is placed. Can be “crop”, “media”, “bleed” or “trim”. The default is “crop”. Will also default to this if a bounding box is specified that is not defined in the PDF. The placed pdf will also be cropped to this bounding box once placed. Please refer to the PDF spec for details about crop, media, bleed and trim boxes.

For Example

```
<PDFPlace Name="TopLeft">
   <File>1254404.pdf</File>
   <PageNumIn>1</PageNumIn>
   <PageNumOut>1</PageNumOut>
   <X>36</X>
   <Y>833</Y>
   <Width>287</Width>
   <Height>375</Height>
   <Scale>Stretch</Scale>
   <PlaceBBox>crop</PlaceBBox>
</PDFPlace>
```

The Name attribute specifies an attribute that is replicated in the corresponding tag in the Results XML.

DrawRect

Draws a Rectangle on a PDF page. It has the following sub tags.

PageNum - The number of the page to draw rect on

Left - The X Coordinate of the left side of the rectangle.

Right - The X Coordinate of the right side of the rectangle.

Top - The Y Coordinate of the top of the rectangle.

Bottom - The Y Coordinate of the bottom of the rectangle.

FillColour - Fill colour. Please see start of XML Command File section for details of how colours are specified. If not supplied then the rentangle will not be filled.

StrokeColour - Stroke colour. Please see start of XML Command File section for details of how colours are specified. If not supplied then the rentangle will not be stroked.

StrokeWidth - Width in points of the stroke. Setting to 0 will mean that the stroke will always be 1 pixel wide on any display that the PDF is rendered.For Example

```
<DrawRect Name="Recttest">
   <PageNum>1</PageNum>
   <Left>500</Left>
   <Top>1000</Top>
   <Right>750</Right>
   <Bottom>750</Bottom>
   <FillColour col1="0" col2="0" col3="255" col4="0"></FillColour>
   <StrokeColour col1="0" ></StrokeColour>
   <StrokeWidth>0</StrokeWidth>
</DrawRect>
```

The Name attribute specifies an attribute that is replicated in the corresponding tag in the Results XML.

Text

Place a line of text on a PDF Page. It has the following sub tags

X - Specifies the horizontal anchor point.

Y - Specifies the text baseline.

String - The text to place.

FontName - is the name of a of font either in the input PDF, in the ResourcePage or in the System Font Path specified in the infixserver config file

Size - Font size in points.

FillColour - Text Fill colour. Please see start of XML Command File section for details of how colours are specified. If not supplied then the text will not be filled.

StrokeColour - Text Stroke colour. Please see start of XML Command File section for details of how colours are specified. If not supplied then the text will not be stroked.

Align - Left, Right, or Center. If Left then the X will specify the left side of the text. If Right then the X will specify the right side of the text. If Center then the X will specify the centre of the text.

For Example

```
<Text Name=”TestText”>
   <PageNum>1</PageNum>
   <X>70</X>
   <Y>50</Y>
   <String><![CDATA[Hello]]></String>
   <FontName>Felix Titling</FontName>
   <Size>18</Size>
   <FillColour col1="0"></FillColour>
   <Align>Right</Align>
</Text>
```

The Name attribute specifies an attribute that is replicated in the corresponding tag in the Results XML.

Render *(Deprecated)*

Use this section to instruct Infix Server to render the PDF processed by Infix Server to a JPEG or PNG file. The attribute **FileFormat** is used to specify the output file formats. It can be one of the following values.

png – for PNG image output.

jpg – for JPEG image output.

The **Render** tag has the following sub tags.

File
BBox
Scale
PageNum
ShowEdits
Stretch
Width
Height
DPI

File

This is the Jpeg file path. If not supplied this can be supplied on the command line using the -r command line parameter.

BBox

Can be one of the following:

MediaBox
ClipBox
TrimBox
BleedBox

Specifies the bounding box within the PDF to render. Will default to ClipBox and then MediaBox if not specified or if specified and the bbox is not defined in the PDF.

Scale

Should not be used in conjunction with Stretch, Width & Height. Set the percentage scale, the output height or the output width of the rendered PDF. The attribute **ScaleMode** specifies the type of scale to be applied to the rendered PDF. If no **ScaleMode** is supplied then render will be at the original PDF size. It can be one of the following values.

percent – Scale as a percentage of the original PDF size to render at. If not supplied here then it can be specified on the command line using the -s command line parameter.

width – Specifies the output width in pixels of the rendered PDF, the height is scaled accordingly.

height – Specifies the output height in pixels of the rendered PDF, the width is scaled accordingly.

max – Specifies the output height or width in pixels of the rendered PDF. Scale so that largest dimension of the rendered pdf is set to **max** pixels. The other dimension will be scaled accordingly.

PageNum

This is the number of the page that is to be rendered. If not supplied then first page will be rendered.

ShowEdits
If specified then any text that has been edited by any of the Infix products, including Infix Server, will be coloured red when rendered.

Stretch
Should not be used in conjunction with Scale. If specified along with Width and Height then the rendered image will stretched to be those dimensions.

Width & Height
Should not be used in conjunction with Scale. Either or both of these dimensions can be supplied and are in pixels. If both are supplied and **Stretch** is not specified then the image will be rendered so that it is the largest size that fits inside the dimensions specified by Width and Height without changing its aspect ratio. If one dimension is supplied then the rendered image will be sized so that it has that dimension without altering its aspect ratio.

DPI
Used to set horizontal and vertical resolution of the rendered image in dots per inch. This just sets the image header resolution values and has no effect on the actual size of the image in pixels