Infix Server Version 2

Reference Manual

iceni technology

Copyright Notice

Notice Of Change

This document represents the state of version 2 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.

09 March 2010

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 CD-Rom which comes with this package are all current versions of InfixServer: **Linux** (Intel), **Solaris** (Sparc) 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, Solaris 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.

-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 1.0, this header looks as follows:

```
ICENI PREFS
OWNER:InfixServer
VERSION:100
```

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.

The sections of a config file can be grouped by key:

SPOOLER	SETTINGS
Input Dir	Log File
Output Dir	Max. Log Size (kB)
Error Dir	Log Trunc. Size (kB)
Completed Dir	Default Font Name
Filter	Built in Fonts Dir
Helios dt	System Font Path
Error Halt	CMaps Path
Keep Log	Max Pages Per Block
Batch Mode	New Document Path
Poll rate	Key Functions
Stable time	CMYK Profile
	Hyphenation Dict
	Do Underlining
	Pay As You Go Key
	Translation Config
	Stamp Pdf

The rest of this chapter consists of a reference to the format of each of these sections. Each entry in a configuration section is immediately followed by its associated arguments in italic.

[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. Care should be taken when setting this parameter since it will have the effect of hiding any font problems encountered.

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

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

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 Infix-Server. 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.littlec-ms.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.

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. **NB the TransXML.cfg file should not be edited.**

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.

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
Document
Manifest
Data
DocumentInfo
Render
Output
PostFile

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: 1.52 Build Date: 13:10:38 Aug 13
2008)</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 the first page of 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 on its first page.

For Example:

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

Document

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 specfied 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.

Manifest

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.

Data

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.

<**br /**> - Insert a soft break

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

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

<**span**> - The style attribute can be used to underline the text with <span style="text-

decoration: underline;">. 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.

DocumentInfo

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>
```

Render

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.

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

File
Scale
SubsetFonts
SizeToGalley
LineCount
BBox
OutlineFonts
FlightCheck
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.

FlightCheck

Perform flight checking on 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 FlightCheck tag, which will be in Output tag, i.e.

```
<Output>
   <FlightCheck>
      <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>
   </FlightCheck>
   <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.

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>
```

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		