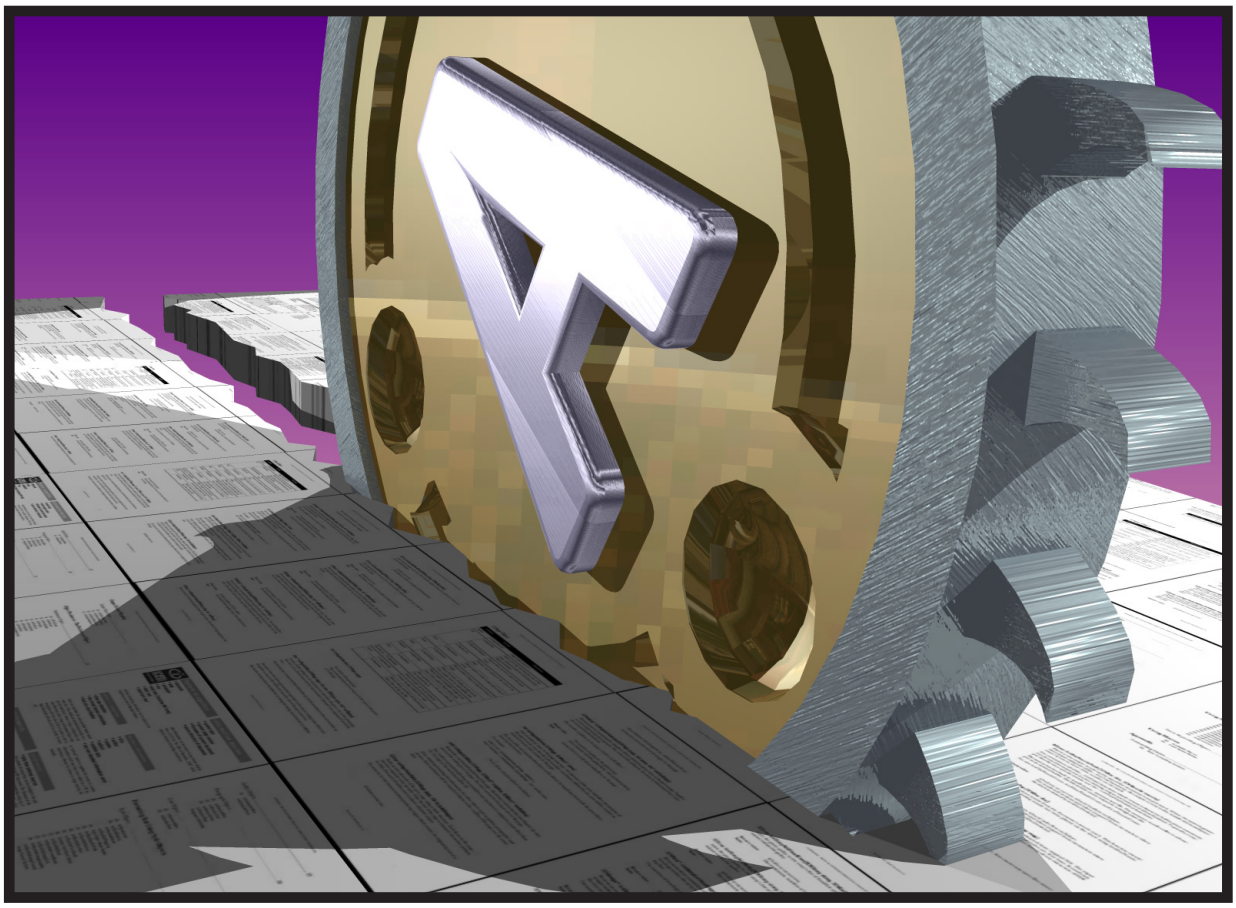


Argus Reference Manual
Version 5.1

Copyright Notice

Notice Of Change

This document represents the state of version 5.1 of Argus. 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.

January 20, 2005

Introduction 15

Using This Manual 15

What's New In Version 5.1 16

New for Version 5.0 16

Installation 18

Installing the Software 18
Obtaining a License Key 18
SDK 18

Compatibility 18

Operation 19

Page Geometry Dump 21

Pre-Programming Annotations 22

Adding Annotations To Every Page 22

Automatically Cropping Pages 23

Using Text Order Templates 23

Dealing With Iconic Font Characters 25

Processing Structured Documents 26

Not All Documents Are Structured 26
Processing Documents 26

Extraction Options 28

Bookmarks 28
Bookmark Path 28
Synth Bookmarks 28
Built In Fonts Dir 28
CMap Path 28
Gridded Text 29
Hyperlinks 29
Image Output 29

Image Path 29
Output Threads 29
Remove Text Items 30
Rotation 30
Structured Output 30
Tabulated Output 30
Text Output 30
Text Path 31
Word List 31
Word Map Path 31

The Configuration File 33

Supported Escape Sequences 33

Sections 34

[SPOOLER] 35

Batch Mode 35
Input Dir 35
Error Dir 35
Completed Dir 35
Filter 36
Log File 36
Error Halt 36
Keep Log 36
Poll rate 36
Stable time 36
Max. Log Size (kB)
Log Trunc. Size (kB) 37

[Document Features] 37

Article Sorting 37
Auto Crop Render 37
Captions 37
ClassMarkup 37
Collapse Spaces 38
Combine Slivers 38
Crop White 38
De-hyphenation 38
Full Page Render Zone 38
Hebrew 39
Illustrations 39
Ignore Text 39
Layout 39
Line Breaks 39
Para Gap 39
Raw Map Words 40
Span Galleys 40
Speechmark Recognition 40
Tab Table 40
Unicode 40
XY Sorting
Column Sorting
No Sorting 41

[Image Style] 41

Alias Limit 41
Colour Format 41
Copyright 41
Comment 42
File Format 42
Greek Limit 42
Image Quality 42
Scale Mode, XScale, YScale 42

[ZONE CONTROL] 43

Boxes 44
Vertical Lines, Horizontal Lines 44

[FONT MAP] 44

Unicode & Double Byte Mapping 45
The “Ignore” flag 45
Font Attributes 46
Glyph Name Output 46

[CHAR MAP] 47

Dbl Byte Esc Start
Dbl Byte Esc End 48
Dbl Byte Format 48
Encoding 48
Removing Line Breaks 48

[RESET ON] 48

[LIMITS] 49

MinFontSize 49
Table Spacing Factor 49
FontSizeGapRatio 50

Interactive Forms 51

[WIDGET] 51
[WIDGET:TEXT] 52
[WIDGET:BUTTON] 52

[WIDGET:CHECKBOX] 52
[WIDGET:RADIO] 52
[WIDGET:LIST]
[WIDGET:COMBO] 53

Character Markup 54

Entity Markup 55

Document, Page 55
Story 55
Headline, Byline, Story Text 55
Bookmark 55
Caption 56
Para 56
Table, Row, Cell 56
Pic 56

[CLASS SPAN] 56

[CSS STYLE] 57

Path 58
Format 58
.....ColorDef 58
FontDef 58
GalleyDef
ParaDef 59
PicDef 59
GenSetDef 60
INFOTABLE 60

[FONT BANDS] 61

USING MACROS 63

Documentation Conventions 63

Utility Macros 63

CHCASE 63
COUNTER 64
FTRUNC 64
HEAD 64
INCLUDE 64
MARKUP 64
MEDIACHANGE 65
REPEAT 65
SECTION 65
TAIL 65
TODAY
TOMORROW
YESTERDAY 65
VALUE 65

Document Information Macros 65

AUTHOR
CREATED
MODIFIED 65
CREATOR
KEYWORDS
PRODUCER
SUBJECT
TITLE 66
CLEAN_FILENAME 66
FILENAME 66
KEY 66
PAGENUM 66
PAGELABEL 66
PAGECOUNT 66
PATHNAME 67
THREAD 67
THREAD_TITLE
THREAD_SUBJECT
THREAD_AUTHOR
THREAD_KEY 67
THREAD_ID 67

Font & Character Information Macros 67

COLOR 67
FONTFAMILY
FONTBASEFAMILY 67
FONTNAME 68
FONTSIZE 68

FONTBASESIZE 68
HTMLFONTFAMILY 69
LEADING 69
SYLKFORMAT 69

Geometry Macros 69

ALIGN 69
INDENT 69
INDENTSIZE 69
MARGINSIZE 69
WIDTH.....PAGEWIDTH
HEIGHT ..... PAGEHEIGHT
POSX ..... PAGEPOSX
POSY ..... PAGEPOSY
POSYR 70

Hyperlinks & Bookmarks 70

BOOKMARK_TITLE 70
HYPERDEST
IFHYPERDEST 70
NESTING 71
PAGEREF 71
TARGET 71
URL
RTF_URL 71
BYLINE 71

Table Macros.....71

CELLWIDTH 71
COLINDEX 71
COLRIGHT 72
COLSPAN 72
ISEMPTY 72
ISINTABLE 72
ISROWSPAN 72
ISTABBORDER 72
NUMCOLS.....
NUMROWS 72
ROWINDEX 72
ROWSPAN 72
RTFCOLORTABLE 73
RTFFONTTABLE 73

Style Sheet Construction & Reference.....73

COLORINDEX 73
FONTINDEX
FONTBASEINDEX 73
GALLEYINDEX 73

GENSETINDEX 74
IGISITALIC
IGISBOLD
IGISMONOSPC
IGISBASECOL 74
IINDEX 74
IMAGEINDEX 74
INFOTABLE 74
PARAINDEX 75

Image Macros 75

BYTES_PER_LINE 75
CAPTION 75
NUM_BYTES 75
OPI 75
OPINAME 76
PIC_COUNT 76
PIC_DEPTH 76
PIC_EXT 76
PIC_NUM 76
PIC_REZ 77
PIC_SPACE 77
PIX_WIDTH
PIX_HEIGHT 77

Interactive Forms Macros 77

WIDGETKEY 77
WIDGETINDEX 77
WIDGETOPTION 77
WIDGETEXPORTVALUE 78
ISEQUAL 78

Formatting Dates 75

Font Encodings 76

Symbol Font Encoding 81

Error Return Values 84

Encoding Names 85

..... 86

Notes 94

Introduction

Argus Server is the most versatile automatic content extraction system available today for PDF documents. It is built upon Iceni's Runway content extraction engine and our cross-platform PDF interpreter.

Argus can extract text, forms data, images, vectors and structural information from PDF as well as render any part of a page at any resolution. Text can be extracted on a page-by-page basis or from article threads when present in a document. The output can be placed in single files or on a file-per-page or file-per-article basis.

Embedded images or rendered areas can be output in EPS, Tiff, JPEG, PNG or BMP formats and scaled arbitrarily. Alternatively entire pages can be rendered to any supported image format including multi-page Tiff.

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.

Argus also comes with an associated software development kit (SDK) enabling its functionality to be embedded in your own C, C++ or Visual Basic programs.

Argus is a highly sophisticated and flexible tool. In fact, the degree of configurability 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 what each configuration option and macro available, but not necessarily how they may be employed. For a more rounded understanding please consult the configuration files included with the Argus distribution in conjunction with this manual.

What's New In Version 5.1

• **New macro for logical page numbers**
The PAGELABEL macro can be used to retrieve page labels from those PDFs that use them. See "PAGELABEL" on page 66.

• **New object geometry macro**
The POSYR macro gives the y position of the current object relative to the bottom of the page. See page 70.

• **New image macro for determining the resolution of embedded images**
See "PIC_REZ" on page 77.

• **Text geometry-dump command-line flag**
See "Page Geometry Dump" on page 21.

• **Improved text rendering**

• **New compiler environments used for Argus on Solaris and Linux**
Both platforms use gcc 3.x and are now linked statically (no additional dynamic libraries are required at runtime).

• **Build date reporting**
Running argus -v now displays the build-date of the software as well as version number.

• **cmaps included with all builds**
The cmaps folder includes Adobe translation tables for many Chinese, Japanese, Korean & Vietnamese font encodings. These files were missing from the standard distribution for 5.0

• **Expanded unifont file**
Argus includes a wider though not exhaustive selection of substitute Asian characters. These characters are crude in design but are sufficient for reading when no embedded font is included in an Asian PDF.

This file is only required if rendering PDF and is not used for extraction of content.

New for Version 5.0

• **Interrogation of OPI dictionaries where available**
For documents containing embedded OPI information, Argus provides a selection of macros to extract various meta data from the OPI dictionaries. See "OPI" on page 75.

• **Extraction of named OPI images**
Argus can be used as an image server extracting specific named images from OPI enabled PDF documents. See "Extraction Options" on page 28.

• **Encapsulated PostScript image output**
Argus can output embedded images as Photoshop compatible EPS retaining clip paths and original colour space. It can also convert entire pages into EPS. See "File Format" on page 42.

• **Handling of image "mosaics"**
Argus will optionally merge thousands of tiny image slivers to form complete images. This is useful when dealing with PDF produced by certain applications which shred images into thousands of single-line slivers to reduce file size. See "Combine Slivers" on page 38.

• **Output ordering templates for text**
A new kind of annotations - "Iceni Order Box" gives template-based control over text output order. A little like applying automatic article threads across a range of pages, this feature can solve problems with multi-column documents that may not otherwise output

correctly. See “Using Text Order Templates” on page 23.

• **Finer control over zoning**
The ability of Argus to maintain discrete area across which text cannot flow has been extended to include areas defined by vertical and horizontal lines as well as boxed areas. This is helpful when exporting newspaper and magazine layouts. See “[ZONE CONTROL]” on page 43.

• **Access to internal limits**
Some of the obscure internal limits have been exposed to allow fine adjustment of aspects of Argus such as minimum font size (below which text is discarded). See “[LIMITS]” on page 49.

• **Output of PDF Forms (Interactive Forms)**
The text within form fields in forms-based PDFs can now be exported. See “Interactive Forms” on page 51.

• **Initial support for right-to-left reading languages**
Documents written in Hebrew can now be output in logical and visual formats. See “When EPS is selected as the image output format, Argus performs a true EPS conversion rather than a render.” on page 38.

• **Support for forced page rotation**
Pages can now be rotated prior to analysis - useful for pages that print with a non-upright rotation. See “Rotation” on page 30.

• **Support for password encrypted documents**
A password may now be supplied on the command line to cope with documents locked with standard encryption. See “Operation” on page 19.

• **Improved rendering speed**

Installation

Included on the CDRom which comes with this package are all current versions of Argus: **Linux** (Intel), **Solaris** (Sparc), **MacOSX** and **Windows**, plus the SDK and associated resources for each platform.

Installing the Software

Copy the relevant Argus distribution folder (Linux, Windows, MacOSX, or Solaris) to the host computer.

The program will run in demonstration mode only until a valid license key is provided. In demonstration mode, all images and random words are corrupted to render them unusable for all but testing purposes.

Obtaining a License Key

All versions of Argus 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 create the unique host-id for the machine running Argus execute the command “argus -v”. As well as printing some version information this will create a file called “hostid.txt”. E-mail this file to sales@iceni.com and we will create a new license key based upon it.

When Argus runs it searches for a file called “argus.key” which should contain 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, Argus will operate in demo mode.

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

SDK

The “C” and Visual Basic SDK is contained in a separate folder with a sub-folder for each platform plus a “common items” folder containing some of the third party libraries required.

Complete documentation for the SDK is contained in the “SDK Reference.pdf”.

Compatibility

Argus is compatible with all versions of from up to PDF 1.4 (Acrobat 5). Beyond this there are some features not yet implement:

- Transparency - Argus cannot render transparency
- Object streams (highly compressed PDF, Acrobat 6 onwards)
- JBIG compression for b/w images (Acrobat 6)
- JPEG2000 image compression (Acrobat 6)

It is anticipated that some if not all of these features will appear in later releases of the product.

Operation

argus -[aefhmpsuv] [-pw <password>] [-c <config file>] <pdf files...>

-a Causes Argus to scan the first input PDF and output a summary of all Iceni annotations found within it. An Iceni annotation is (at the time of writing) an “Iceni Table Box” and “Iceni Image Box”. Both of these annotation types may be added to a document using a demo version of the Gemini plug-in for Acrobat.

The annotation summary (the Annotation Plan) is written out to the Console along with the contents of the current configuration file.

See “Pre-Programming Annotations” for more information on using this feature.

-c Specifies the name of a configuration file to use. If omitted, the program will generate a default configuration file called “argus.cfg”. This is often a good place to start when preparing a new configuration.

-e Argus will scan the first given document and take note of any logical structure tags found in it. These tags are then used to create a set of entity definitions which are written to a file called “entity.cfg”.

When dealing with a Structure/Tagged PDF these entity definitions can be used to map entity names in the original document to those desired on output.

See “Processing Structured Documents” on page 26 for a step-by-step guide to using this feature.

-f Causes Argus to output a list of fonts used in the document. The list covers only those pages processed. This facility is useful when creating FONT MAP sections in a configuration file which relate to a specific font. See “[FONT MAP]” on page 44 for further information.

-g Output geometry of all text on all pages in the range. Also output width & height of each page. See “Page Geometry Dump” on page 21.

-h Specifies an alternate file name for the host id key file. The default is “argus.key”.

-m No messages. By default Argus will output various status messages during document processing. When this option is used progress output is disabled except in the case of an error (or when using -a).

-p Specifies a page range for processing. Applies only when page-based output mode is selected (See “Argus Controls”)

Some example page ranges are given below:

1Process only the first page
1-#Process all pages
15-23Process page 15 to 23
#Process only the last page
1,#Process the first page and the last page
1-3,#Process the first 3 pages and the last page

Do not include spaces in a page range. Argus will assume anything after the space is the start of another argument or file name.

-pw Password. Enables Argus to open (decrypt) documents that have been encrypted with a password. The given password will be used for all documents given on the command line. Do not supply a password for unencrypted documents or those that have no user password.
Argus can cope with both 40 and 128-bit standard encryption.

-s Spooler operation. When spooling, Argus polls a specified input directory for files to

process. These are processed and placed into an output directory. The parameters controlling spooler operation are held in the “Spooler” section of a configuration file. See “[SPOOLER]” on page 35

-u When this flag is given during normal operation, Argus will update the supplied configuration file if its version number is less than the current version of the program.

This used to happen automatically whenever an old configuration file was loaded prior to Argus version 4.

-v Causes the program to output a simple version message. Any additional parameters will be ignored and no PDF processing will occur. A default configuration file will be generation called “argus.cfg” if no ‘-c’ has yet been encountered.
This option will also verify that a valid dongle is connected to the host machine and audits the number of “clicks” remaining on the dongle.

-xopi <opi image name>[,<opi image name>]*
Using this option causes Argus to export only images whose OPI name matches one of the names given.The list should not contain spaces - if it must, enclose the entire list in quote marks.

If for example a document contained a image whose OPI dictionary state that its original filename was “MBUK161.group.pic20” then a command such as:

```
argus -c imageConfigs/tiff.cfg -xopi MBUK161.group.pic20 myDoc.pdf
```

would export only the named image.
Name matching ignores case and will match any part of a name. So the following names would also match:

```
mbuk161.group.pic20
mbuk161
pic20
```

Page Geometry Dump

If the -g command line flag is provided, Argus will output a summary of all text on a page with coordinates. A typical dump is illustrated below:

```
> argus -g -p 25 referenceManual.pdf

Page 25 595x842

275    87     23     10     Page
298    87     9      10     24
274    797    24     10     Argus
298    797    11     10     5.0
96     765    109    27     Processing
204    765    103    27     Structured
306    765    112    27     Documents
125    747    24     11     From
148    747    21     11     PDF
169    747    15     11     1.3
183    747    39     11     onwards,
222    747    46     11     documents
267    747    16     11     can
283    747    32     11     contain
315    747    43     11     additional
357    747    42     11     meta-data
399    747    26     11     called
425    747    37     11     structure
462    747    16     11     tags
125    735    18     11     that
142    735    40     11     describes
181    735    20     11     both
202    735    15     11     the
216    735    6      11     <fb02>
222    735    12     11     ow
.....
```

The first line of the dump contains the page number and dimensions (relative to any page crop) of the page in PostScript points. If there is no page crop in use, these dimensions will match those of the page's media box.

The rest of the dump is split into 5 columns:

```
[x co-ord] [y co-ord] [width]  [height]   [text]
```

The x,y coordinates are relative to the bottom left of the crop box[1] (or page media box is there is no crop box).

Where text includes double-byte (wide) characters they are represented as hexadecimal and placed inside angle brackets. For example:

```
216    735    6      11     <fb02>
```

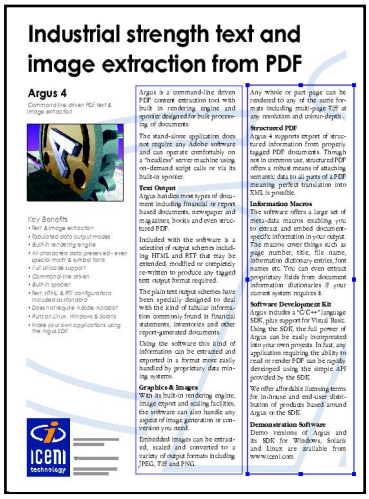
1. In the "Amazon" build of Argus the x,y coordinates are relative to the top, left of the crop box (or page media box is there is no crop).

Pre-Programming Annotations

By adding special PDF annotations to documents, Argus can be given “hints” on how a document should be extracted. These hints can include the locations of tables, images and regions to be ignored within documents.

At the current time, Argus supports four annotation types-

Iceni Image Box
Iceni Crop Box
Iceni Table Box
Iceni Order Box



These can be added to PDF’s manually using the demo version of Gemini for Macintosh or Windows (available from our web site).

In order to pre-program annotations into Argus, an annotated PDF file is required for input. The annotations are harvested from the PDF and written into an argus configuration file. The format of an annotation in the configuration file is:

<number>.<type>=<page range> [<top> <left> <bottom> <right>] <flags>;

For example, if Argus is required to process 10,000 documents each with a table in the centre of every page the following procedure should be followed:

- Open an example of the documents in Solo. Using the table box tool, add boxes where you require them, setting the page range of each accordingly.
- Save this document.
- Using Argus, issue the following command to process the document just saved:

  argus -a savedExample.pdf > newConfig.cfg

  This will cause Argus to read the PDF and output a new configuration file called “newConfig.cfg” containing an *annotation plan* detailing all of the annotations to be added when processing future documents.
- Incorporate the [ANNOT PLAN] section from this config in your own config files (or use the #include mechanism).

To now extract all tabular information from similar documents, use this new configuration file.

Note that documents processed in this way are not modified in any way.

Do not process the document used to create the new configuration file. Since that document already contains annotations, Argus will add them plus those listed in the annotation plan. This will cause incorrect output where two annotations clash. Instead ensure that all Iceni annotations are removed from the document before it is processed or process it using a config file with no annotation plan.

Adding Annotations To Every Page

It is possible to edit the annotation plan to ensure that an annotation is mapped to every

page of the document even if the example document from which the plan was created only contained the annotation on a single page.

This avoids the need to manually annotate an entire document before creating the annot plan.

The following is a simple ANNOT PLAN section taken from an Argus configuration file:

```
[ANNOT PLAN]
0.Iceni Table Box =1, [732.25, 30.82, 224.70, 593.09];
[-- END --]
```

This plan instructs Argus to add an “Iceni Table Box” annotation on the first page of every document with the coordinates (732.25,30.82) (224.7, 593.09).

The page number (1 in the example above) may be a fully specified page-range as used in the “-p” command-line flag for Argus. See “Installation” on page 18 for information about the format of a page range.

For example, to process a hypothetical set of documents with tables on pages 5-10 and 12 as well as ignoring headers from page 2 onwards, the following annotation plan might be used:

```
[ANNOT PLAN]
0.Iceni Table Box =5-10,12 [732.25, 30.82, 224.70, 593.09];
1.Iceni Image Box =2-# [0.00,600.00, 400.00,740.00];
[-- END --]
```

If you require assistance setting up an annotation plan, please email support@iceni.com. Demo versions of Iceni’s “Gemini” plug-in and “Gemini Solo” application are available from our web site - www.iceni.com.

Automatically Cropping Pages

By creating an “Iceni Crop Box” annotation, Argus can be instructed to apply a temporary crop box to a range of pages. This may be useful for removing repeated headers and footers (such as page numbers) from the output.

Adding a crop box to the annotation plan is done in the same way as both table and image boxes. However, the Gemini plug-in which can be used for this purpose doesn’t list “Iceni Crop Box” as one of its link types. Instead use either the table or image box to define the cropping rectangle. Once a new configuration file has been created with a new annotation plan, edit the annotation name to be “Iceni Crop Box”.

Any text outside of the crop box will not be output. You can only have one crop box per page. If you specify two for the same page, only the first listed will be used.

Example:

```
[ANNOT PLAN]
0.Iceni Crop Box =1-# [341.15, 91.78, 259.87, 364.84];
[-- END --]
```

Using Text Order Templates

Sometimes Argus does not output text in the correct order. This can occur with multi-column documents for example. You can use Article Threads to remedy these problems but these have to be added manually to each document before processing. A more automated, though less sophisticated approach is to use “Iceni Order Box” annotations.

These are similar to crop, table and image annotations and are stored in the [ANNOT

PLAN] section of a config file. Typically a number of order boxes will be placed on a page and made to span for a number of pages. Each order box is numbered and behaves in a manner similar to an article thread. Text falling into order box 0 for example, is output before text within order box 1 and so on. There is no limit to the number of order boxes that may be added to a page.,

For a two column document only two boxes need to be added. These should each be made to enclose one column of text. Any text not enclosed such as headers and footers will be treated as normal. If your document layout changes on later pages, add further order boxes and adjust the page-range of all boxes so the entire document is covered.

There is no tool specifically for creating order boxes in versions of Gemini Solo prior to 2.0. An example of using a pre-2.0 version of Solo to create an order plan is show below.

Step 1: Open your document in Solo and add image box annotations around columns or blocks of text you want to control. A example screen shot is shown below of the Macintosh version of Solo.

This screenshot shows two table boxes (shown in blue) placed around two columns of text. If this document has following pages in a similar layout, the page range should be set appropriately for each box so that is covers these pages.

If the layout changes, add new annotations to reflect this adjusting the page-range of all annotations so that they do not overlap.

Step 2: Save the document.

Step 3: Run the command:

```
argus -a doc.pdf > annot.cfg
```

where ‘doc.pdf’ is the document just marked up. This will generate a new config file called ‘annot.cfg’ with an [ANNOT PLAN] section something like the following:

```
[ANNOT PLAN]
0.Iceni Table Box =1-10 [666.21, 400.88, 30.43, 580.21] 0;
1.Iceni Table Box =1-10 [665.13, 212.87, 30.43, 392.19] 0;
[-- END --]
```

Step 4: Change ‘Iceni Table Box’ to ‘Iceni Order Box’.

Step 5: Specify the order of the two boxes. Do this by looking at the coordinates of the boxes and deciding which should be output first. The format of the coordinates shown is:

```
[top left bottom right].
```

Remember that the origin in PDF documents is the bottom left of the page. So after renumbering, the example above would be:

```
[ANNOT PLAN]
0.Iceni Order Box =1-10 [666.21, 400.88, 30.43, 580.21] 2;
1.Iceni Order Box =1-10 [665.13, 212.87, 30.43, 392.19] 1;
[-- END --]
```

The numbers you use are not important as long as the are in order (you could have numbered then 0 and 1 for example).

Step 6: Manually add the annot plan to your own config file or use the #include facility to import it.

Step 7: Remember to remove the table boxes from your original document since they would cause problems with the text output if you processed the document with your newly modified config.

Dealing With Iconic Font Characters

Symbol fonts such as Monotype-Sorts, MathPi, ZapfDingbats etc. are difficult to convert in a way that is suitable for a wide range of output formats. Due to the irregular nature of the characters in such fonts, a general translation from PDF into HTML or RTF may not always be possible.

One approach to help in solving this problem is to combine the word mapping facility (see “Word Map Path” on page 31) with the glyph name output feature of the FONT MAP section (See “[FONT MAP]” on page 44).

For example, given a document with an iconic style font called “WoodOrnamets” containing:

Text with a ✂ symbol and a ❤ icon

the following FONT MAP will cause the glyph names of characters to be output rather than the actual character codes:

```
[FONT MAP:WoodOrnaments]
Glyph Names =true;
Glyph Start =[.;
Glyph End =];
[-- END --]
```

When output, a document may yield:

Text with a [.scissors] symbol and a [.heart] icon

To then convert these glyph sequences into something more useful in HTML for example, a word map file called “substitutes.txt” is created with the following entries:

```
[.scissors]:<IMG SRC="scissors.jpg">
[.heart]:<IMG SRC="heart.jpg">
```

Argus needs to be instructed to make use of this word map file. In the ARGUS CONTROLS sections the following line is added:

```
Word Map Path=substitutes.txt;
```

When processing the same document, the output will now be:

Text with a <IMG SRC="scissors.jpg"> symbol and a <IMG SRC="heart.jpg"> icon

This is how the text would appear in an HTML browser. Argus will have mapped the HTML special characters < and > to ensure that they do not inadvertently effect the HTML formatting. However, in this case, this mapping is undesired. Instead Argus must be told to bypass the CHAR MAP for substituted words. This can be done in the DOCUMENT FEATURES section with the following:

```
Raw Map Words = true;
```

Note that this kind of use of the word-map feature is only intended for documents containing a few special characters. For documents consisting mainly of glyph fonts, it is better to use a post process search-and-replace procedure after Argus has finished.

Processing Structured Documents

From PDF 1.3 onwards, documents can contain additional meta-data called structure tags that describes both the flow of a document and arbitrary attributes for each component of it.

For PDF 1.4, a properly tagged document must have a tree-structure with the document itself at the root of the tree. Branches and nodes of the tree are represented as chapters, sections and paragraphs or any other hierarchic arrangement you care to use.

The tagging mechanism is so flexible that any tag can be used for any part of a document, there being no need to stick to rigid monickers such as “paragraph” or “chapter”. A Document could just as easily be a collection of “flibberts”, grouped by “mickels” and “trandons” for example.

Argus version 4 is able to interpret the tree structure of a tagged document. However, it does not offer access to all of the attributes assigned to each particular component. This facility will be included in later versions.

The program can output the text of a document by performing an in-order traversal of the structure tree. This means that in theory, the output order of text may no longer bare any relationship to the order of pages in the document but will instead follow that of the structure tree.

Not All Documents Are Structured

Even though PDF may contain any kind of structure information required, it is the responsibility of the generating application to include this information. As at the time of writing, few applications support full structure tagging. Microsoft Word and Adobe’s InDesign support it to an extent and others will no-doubt follow suit in the future.

Processing Documents

Using Argus to output a document via a structure tree traversal is typically a three stage process.

Stage 1

The first stage is to catalogue the structure tags used in the document. Since these can be arbitrary names, the use of a particular name such a “paragraph” cannot be taken for granted.

Using the *-e* flag to Argus will cause it to dump a new configuration file containing formatting entries for every structure tag found in the document.

By default the configuration file is created in such as way as to output the tag start and end in an XML fashion such as <section>, </section> with the textual content in-between. However, you will probably want to tailor this to your particular requirements as follows.

Stage 2

Edit the configuration file produced. Alter the *Start* and *End* formatting for each tag listed to generate the final output format you require. For HTML output for example, you may wish to format the “<P>” and “</P>”.

You must also enable structured output in the “DOCUMENT FEATURES” section of the new configuration file by setting the “Use Structure” flag to true.

Stage 3

Now your original document and any others like it can be processed using the new con-

figuration file created. Since the “Use Structure” option has been enabled, Argus will output the text of the document by traversing the structure tree (there is no need for the -e flag now).

The program will output the text content of each structural element encountered, bounded by the *Start* and *End* formatting you prescribed for the element.

You may use this same configuration file for all documents produced by the same program with the same tags. Different PDF producers will probably need different configuration files tailored to their own tag names.

Extraction Options

The way in which Argus analyses and processes documents can be controlled to a degree via a series of behaviour flags and path settings. These settings are all listed in a section entitled [ARGUS CONTROLS] in the configuration file. See “The Configuration File” on page 33 for a description of the overall layout of a standard configuration file.

The flags in the [ARGUS CONTROLS] section can be enabled by either of “true” or “1”. Similarly they can be disabled using “false” or “0”.

When specifying a pathname that includes slashes, Argus makes no distinction between forward (Unix) and backward (Windows) slashes. In fact, since a backslash must be escaped as ‘\\’ it is more convenient to use forward slashes (Unix style).

There now follows a description of each setting available.

Bookmarks

true | *false*

Bookmark Path

pathname

Synth Bookmarks

true | *false*

Controls the output of bookmarks. These are sometimes present in PDF documents and consist of a hierarchic list of destinations within the document body.

When enabled, the output is sent to the destination described in “Bookmark Path”. It also causes the definition of hypertext anchors within the document body to act as destination for the bookmarks.

If a document contains no bookmarks of its own Argus (from 3.1 onwards) will synthesise bookmarks based upon the content of the document. It does this by looking at the headings in a document and sorting them into a hierarchy based upon the font size of the heading.

This feature can be disabled via the “Synth Bookmarks” flag.

See “HYPERLINKS” and “BOOKMARKS”

Built In Fonts Dir

pathname

Specifies the folder containing the standard Type1 fonts used by the renderer during font substitution. If not set, Argus will not be able to render text when the document does not include the font for the text.

CMap Path

pathname

Specifies the location (directory) of the standard Adobe CMAP translation files. These files enable Argus to translate fonts encoded in locale specific schemes such as Big-5 into Unicode. Once in unicode format, the program can then convert then into any of the supported output encodings - see “[CHAR MAP]” on page 47 for details of output encoding.

Gridded Text

true | *false*

Outputs text using a grid system which attempts to retain the layout of the original page. Words are positioned on the page using spaces.

Useful for both plain text output and HTML if the <pre> tag is used.

This mode may be useful for exporting invoices or other simply constructed PDF pages into a plain text format while preserving the layout. Depending upon the complexity of the input PDF, it is not always possible to replicate the layout exactly.

When viewing output generated in this way it is important that a mono-space font is used such as Courier since the layout effect relies on the fact that each character has the same width.

Hyperlinks

true | *false*

Controls the output of hypertext links within a document. When enabled hypertext links in the original PDF are re-created in the output within the limits of the chosen output format.

HTML output is quite suitable for recreating most link types although Argus does not create image map links.

See the HYPERLINKS sections for more information.

Image Output

true | *false*

When enabled, images are output when encountered in the original PDF document. The exact format of the images output is controlled by the “IMAGE STYLE” section. The location of the output images is controlled by the “Image Path” pathname (see below).

Argus version 3 now outputs all images. Previous versions did not output very small images (less than 4k bytes in size) or images held in Forms.

Image Path

pathname

Sets the destination for image output. The definition may include macros (& in fact makes little sense if it does not).

For example:

```
Image Path =/users/home/images/[IMAGECOUNT].jpg;
```

This causes each image to output to a file named by the image count of the image. i.e. 0.jpg, 1.jpg, 2.jpg etc.

Any subdirectories mentioned in the pathname will be automatically created by Argus as necessary.

Output Threads

true | *false*

When true the documents are no longer output on a page-by-page basis but on a thread-by-thread basis. “Thread” is another name for “Article” as used within the Acrobat application.

An article thread can be defined by the user within Acrobat or by the program producing the PDF document (as is the case with FrameMaker). Each thread can span many pages and serves to guide the reader through one complete article within the document.

Hence extracting by thread ensures that articles are exported complete with all their paragraphs. Even when an article spans multiple columns and pages, Argus will reflow the article to produce one complete text block.

The destination for output of each thread is controlled by the “Text Path” pathname.

Remove Text Items

true | *false*

When true, no text items from the original page will be output. This is not the same as disabling text output. A typical use of this is to create a config file which produces a summary or catalogue of images in document. Argus will find only images and illustrations in its display list after removing text items but will still produce the output specified in the [PIC] section and send it to whatever file is specified for text output.

To see this in use please refer to **imageConfigs/opihtml.cfg** which output no text items from PDF but just an inventory of images, formatted as HTML.

Rotation

0 | *90* | *180* | *270*

Forces Argus to treat documents as if they were rotated by the given number of degrees. This is most useful for processing documents that are rotated but that do not indicate the fact in their page dictionaries.

The default position is to specify no angle.

Structured Output

true | *false*

When enabled Argus will ignore the physical order of text and pages in a document and will instead output text in the order dictated by a document’s structure tree. This setting should only be used for documents known to have valid and useful structure trees i.e. tagged PDF only.

See “Processing Structured Documents” on page 26 for more information.

Tabulated Output

true | *false*

When true causes each page to be treated as if it were laid out using a complex table structure. This table is then translated into the output format using the definitions of TABLE, ROW and CELL in order to better preserve page layout.

Note that this option does not normally retain the layout of tables within the original PDF.

Due to the possible complexity of PDF pages, this feature rarely results in an accurate preservation of layout but may provide an improvement over non-tabulated output.

Text Output

true | *false*

When false no text is output from Argus. Set to false when only image output is required (as is the case in the tiff.cfg example configuration file included with the distribution).

Text Path

pathname

Controls the output destination for text. Both page text and thread text make use of this value. The pathname itself may be made up of several macros in order that it reflects the documents or pages etc. from which it came.

Examples of path names are:

a) Text Path = docs/[FILENAME]/page[PAGENUM].html
b) Text Path = docs/[FILENAME].html
c) Text Path = docs/[FILENAME]/thread-[THREAD_ID].html

a) Outputs each page of a document in a separate file, all contained within a directory with the same filename as the original document.

b) Outputs entire text of document in one file with the same filename as the document.

c) Outputs each thread of the document in a separate file, all within a directory with the same filename as the document. (Only works when “Output Threads” is enabled.)

Although many macros may be used within the pathname, not all are appropriate when taken out of context and in fact may cause an error. For example “FONTSIZE” cannot be used in this way since it is only in context during output of text from the document.

Any subdirectories contained within the pathname will be created by Argus if they do not exist.

In order to force text to be sent to the console rather than to a file, set the pathname to nothing. That is:

```
Text Path =;
```

Word List

pathname

Points to a file containing a list of words. These words are use by Argus when deciding if a hyphenated word found at the end of a line should be de-hyphenated. If the de-hyphenated form of the word exists in the dictionary, Argus will de-hyphenate it. If one of the words each side of the hyphen is found in the list, Argus will not de-hyphenate.

The list, which exists as a single line of space delimited, lower case words, may be modified manually. However it must always remain in sorted alphabetic order otherwise word look up will fail.

Words entered into the list do not need plural equivalents - Argus automatically deals with plurals. It should be noted that the default English language list supplied with Argus does contain some plurals and that these are redundant.

Word Map Path

pathname

Points to a file containing word mappings. The mappings are from one word to another word or sequence of words. This facility is mainly used for mapping glyph names into more useful text sequences.

The format of a word map list is:

```
<target word>:<replacement text>
```

For example, the following is a valid word map list:

```
[H9254]:<SYMBOL>delta</SYMBOL>
[H9278]:<SYMBOL>phi</SYMBOL>
```

When Argus decides to output the glyphname of a character because it is a non-standard name, the software will consult the word map to see if the name can be mapped to any-

thing more meaningful. This is of most use for fonts containing non-standard glyph names.

By default all replacement text is itself subject to character mapping via the CHAR MAP. This second mapping can be avoided using the “Raw Map Words” feature flag. See “Raw Map Words” on page 40.

The Configuration File

The configuration file is the main way in which Argus is controlled whether it is the command line or SDK version being used. 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 Argus. For Argus 4, this header looks as follows:

```
ICENI PREFS
OWNER:Argus
VERSION:500
```

At any point throughout the file, additional sub-files can be included. This is useful for including such things as standard character maps shared amongst a number of different output formats.

To include a sub-file, use the following:

```
#include filename
```

This directive must be the only thing on the line and cannot span multiple lines. Everything after the “include” is taken to be the file or path name.

Included sub-files can themselves include further sub-files. Subfiles do not require the three line version header as described above.

For example, the html css1 configuration supplied with the distribution includes the following:

```
#include charMaps/htmlSimple.cmap
```

This causes a commonly used character map to be included. Changes to this character map will be reflected in every configuration file which references it.

Supported Escape Sequences

The following escape sequences are supported by Argus 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

For example:

```
[FONT MAP: StrangeFont]
d55 =\157;          Octal character 157 (111 decimal);
d56 =\x2001;        Hex character 2001 (8193 decimal);
```

The hexadecimal escape mechanism is currently the only way to represent double-byte characters in Argus.

When specifying a pathname that includes slashes, Argus makes no distinction between forward (Unix) and backward (Windows) slashes. In fact, since a backslash

must be escaped as '\\' it is more convenient to use forward slashes (Unix style).

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 Argus 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 function:

CONTROL & BEHAVIOUR	CHARACTER MARKUP	ENTITY MARKUP
CHAR MAP	BOLD MARKUP	BOOKMARKS
DOCUMENT FEATURES	COLOUR	CAPTION
FONT MAP	FONT BANDS	CLASS SPAN
IMAGE STYLE	ITALIC MARKUP	CSS STYLE
PROFORMA	MONO SPACED MARKUP	DOCUMENT, PAGE
RESET ON	SANS SERIF MARKUP	GALLEY
SPOOLER	SERIF MARKUP	HEADLINE
ZONE CONTROL	SUB SCRIPT MARKUP	HYPERLINKS
LIMITS	SUPER SCRIPT MARKUP	PARA
		PIC
		STORY, STORYTEXT
		TABLE, ROW, CELL
		WIDGET

Though there may be other sections available in the file, these are not used by this version of Argus.

The rest of this document consists of a reference to the format of each of these sections. For examples of using particular features of Argus, please also see the configuration files

included with the distribution.

Each entry in a configuration section is immediately followed by its associated arguments in italic.

[SPOOLER]

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

In *batch mode* a list of all files to be processed is gathered then Argus 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 files added to the input folder are detected by Argus 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 arbitrary order until none is 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 Argus. 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.

Batch Mode

true | false

When true, Argus will operate in batch mode.

In Batch Mode, Argus 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.

Error Dir

pathname

If an error is encountered during processing, a PDF job 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 PDF jobs 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 Argus 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=*.pdf;
Filter="Legal? Text.pdf";
```

Only one filter may be supplied in a configuration file.

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

Log File

pathname

The location of the log file. All activity of the spooler 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, Argus will truncate it periodically to any length required. See Max. Log Size below.

Error Halt

true | *false*

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

Keep Log

true | *false*

When true, Argus maintains a log file of activity. 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 Argus 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.

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 Argus 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.

[DOCUMENT FEATURES]

Most entries in this section control features of the underlying Runway extraction engine employed by Argus. Each is a simple on/off switch. Use '1' for on and '0' for off (or "true" and "false").

Article Sorting
Related paragraphs and headings are grouped into a "Story" object. This may alter the reading order of the page.

Useful for article-based documents such as newspapers and magazines. This feature should be disabled for documents known not to follow this pattern, such as books and memos since a different sorting method is employed in these cases.

Do not confuse "Article Sorting" with threads and threaded documents. The latter is a more effective method of ordering or grouping text. See "Output Threads" on page 29 for more information.

Auto Crop Render
When true, Argus will apply a crop to each rendered page cutting out unoccupied space. The crop is formed by merging the bounding boxes of every item rendered on the page.

See "Full Page Render Zone" on page 38 for details of page rendering.

Captions
Associates text with images. On occasion body text may be mis-classified as an image caption.

Disable this feature for documents that contain little or no true picture captions.

ClassMarkup
This flag changes the way Argus outputs markup characters. When off, Argus will output the text defined for Bold, Italic, font size etc. whenever one of these attributes changes.

When enabled, Argus outputs a tag which represents the combination of attributes for the current text. It does this by first cataloguing all combinations of character style used in a document. A reference to this catalogue is then all that is output when a style change is required.

This behaviour means Argus can output correct Cascading Style Sheets tags when required. See "[CSS STYLE]" on page 57 and "[CLASS SPAN]" on page 56 for more information.

Collapse Spaces

When true, removes repeated spaces from words and omits words containing only spaces. If a document uses the space character for layout, there may be a lot of repeated spaces in the output unless this flag is enabled.

Combine Slivers

Enabling this flag may help when dealing with documents containing images that have been split into many (possibly thousands) of single-pixel strips.

Images are sometimes broken up in this way by the PDF producer in an attempt to reduce file size. When enabled (the default position) Argus will attempt to identify and merge image slivers into a single larger image.

Since a PDF contains no information positively identifying image strips, this procedure may in some instances join images that should not be joined or fail to merge those that should.

Crop White

When true, causes white coloured text to be output as light grey so that it may still be read when viewed against a white background.

De-hyphenation

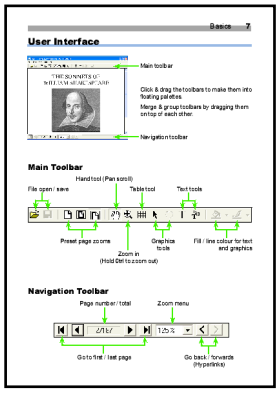
Words which are broken at line endings by a hyphen are normally rejoined into a complete word. This may sometimes result in two words which should remain hyphenated (such as “high-powered”) being incorrectly joined.

Disable this feature to avoid such an error.

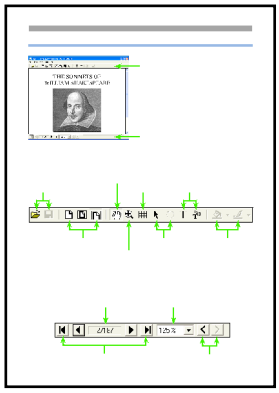
Full Page Render Zone

When true causes Argus to automatically add an “Iceni Image Box” annotation to each page processed. This will cause the entire page to be rendered. When true, the value of “Ignore Text” is especially important since it will dictate whether or not any text is output in either the rendered page or the text extracted from it.

When EPS is selected as the image output format, Argus performs a true EPS conversion rather than a render.



Ignore Text=true (text is rendered but not exported)



Ignore Text=false (text is exported but not rendered)

Hebrew

Causes Argus to treat documents as if written right-to-left (as are Hebrew and Arabic). Lines of text are constructed right-to-left and paragraphs are linked in top-to-bottom, right-to-left order.

To achieve this Argus flips all x coordinates on the page then treats the page as if it were western (left-to-right). This means all coordinates including those of images will be mirrored horizontally. It will not however, effect the order of image data.

This flag will cause processing to take approximately twice as long per page.

Illustrations

When true, Argus will try to identify regions of pages containing vector artwork - illustrations & line drawings rather than photographs. Once recognised, the program can render the area into an image an output is as a JPEG, PNG etc.

Argus looks for groups of curved segments in attempting to identify such regions. Due to the simple nature of this detection scheme Argus may identify the wrong area or miss an area altogether. If this is the case, it can be told about an illustration via an “Iceni Image Box” annotation. These may be created in the ANNOT PLAN section (see “Page Geometry Dump” on page 21) or added using a demonstration version of Iceni’s Gemini Solo application - available from www.iceni.com.

Ignore Text

When Argus renders an area of a page that contains text this switch controls whether or not the text appears in the rendered version or as true text in the text output stream.

When true, text contained in a rendered region is not included in the text output stream, just in the rendered image.

When false, the text is included in the text output stream but not in the rendered image.

“Full Page Render Zone” on page 38 for an example illustrating this effect.

Layout

Modifies the way that paragraphs and galleys of text are formed to ensure that a larger number of discontinuous objects are formed. This may improve output when attempting to preserve layout. It should not be enabled when text is to be reflowed.

The bundled configuration file html/super.cfg uses the layout feature to break-up the text into as many objects as possible. It also employs “Shatter Document” and “Split Lines” to further splinter the objects. It then uses line-by-line positioning to attempt to duplicate the original layout of the PDF in HTML.

Line Breaks

When set to true (the default is false) Argus will preserve the line breaks observed in the original document. The “force line break” character sequence (see “Para” on page 56) is used cause the line-break when needed.

See “[FONT MAP]” on page 44 for a font-specific approach to removing line-breaks.

Para Gap

Vertical gaps between paragraphs are normally discarded on output. If this feature is on, these gaps cause the forced line-break phrase to be output in an attempt to replicate the gap. This occurs in addition to normal paragraph formatting. See “Entity Markup” on page 55

Raw Map Words

When using the word mapping facility (see "Word Map Path" on page 31) Argus will pass any substituted words through the CHAR MAP (see "[CHAR MAP]" on page 47) prior to output. In some cases it may be useful to avoid this step when a substituted word includes HTML formatting commands for example.

Setting this flag to true (the default is false) will cause Argus to omit the final CHAR MAP stage for substituted words, instead letting characters pass through to output untouched.

In order to use this facility effectively the word map file must be carefully constructed to ensure it does not contain any characters or phrases which are illegal or reserved in the output format being used.

See "Dealing With Iconic Font Characters" on page 25 for more information.

Span Galleys

A galley as defined in Argus is a vertical group of paragraphs. Argus attempts to link similar galleys into reading order.

When false, Galley formatting is output at the start and end of every galley. When true, it is only output at the start and end of a group of linked galleys.

Speechmark Recognition

The rules of reported speech in English language dictate that a line-break indicates an alternate speaker. Hence it is vital that Argus preserves breaks when they occur in speech.

Enabling this flag causes Argus to detect the presence of quotation marks and preserve line-breaks in their vicinity. This may fail if the fonts used have non-standard encodings.

Disable this feature for documents not expected to contain reported speech.

Tab Table

When true, Argus assumes that all pages are mainly tabular in nature. It decomposes pages into individual cells rather than trying to forms lines and paragraphs.

When dealing with heavily tabular pages such as stock results, financial reports and invoices this flag should be enabled.

The "tabText.cfg" supplied with the distribution relies upon this flag.

Do not used this flag if you wish to export normal pages that contain a few tables. In this case it is better to mark-up the tables concerned and use the normal processing options.

Unicode

When enabled, all characters will be translated into Unicode (where possible). Ensure that any character maps used when enabled, are written for Unicode and not for PDFDoc encoding.

When false, characters are encoded internally as PDFDoc encoding. In this case, ensure all character maps used are compatible with PDFDoc encoding.

If any output text encoding is used such as UTF-8 or Shift-JIS, this option must be set to true since the encoding mechanism expects all text to be encoded as Unicode internally to Argus. See "[CHAR MAP]" on page 47 for details of using encodings.

XY Sorting
Column Sorting
No Sorting

true | *false*

These options change the sorting used by Argus prior to outputting text. They should be used only when the default method is producing poor results. None of these methods should be used if "Article Sorting" is enabled.

XY Sorting: a rigid x-y sort is applied to the text on each page. Text at the top-left of the page will be output first

This is especially useful when text is printed out-of-order on the page - a common occurrence when pages include footnotes, superscripts, lists or tables. Pages produced by newer PDF generators should always be generated in-order but some software still does not do this.

Column Sorting: text is sorted into columns with text at the top of the left-most column being output first. This may be useful for processing book-like texts containing two columns of text on each page for example.

No Sorting: text is output in printing order. Adobe Inc. recommends modern PDF creators produce PDF so that the reading order of text is the same as the printing order. Hence for modern PDF documents, this approach may produce good results.

[IMAGE STYLE]

This sections contains various flags and settings for controlling the output of images and rendered regions by Argus.

Alias Limit

number

Text rendered with a size larger than the Alias Limit will not be smoothed. All text smaller than the limit will be subject to smoothing (except for monochrome output formats).

As the output resolution is increased, more a more text will exceed the given limit until none is smoothed.

Colour Format

Bitmap | *Dithered* | *Grey* | *RGB* | *CMYK*

This setting specifies the colour depth of rendered images only - it does not affect embedded images such as photos and scans.

Some colour depths are unsupported in some file formats. For example, CMYK is only supported by Tiff. Argus will report an error when an inappropriate colour format is specified for a particular file format.

For EPS file formats, this setting effects only the preview included in the EPS, not the main image itself.

Copyright

text

Specifies the copyright text to be included inside the image. This is applied to the Copyright tag in a TIFF file or the comment tag within JPEG & PNG files. BMP does not support associated text information.

The definition may include macros.

Example:

```
Copyright =[AUTHOR];
```

Comment

text

Specifies the comment to be included within the Comment field of a Tiff, JPEG or PNG image on output. BMP does not support associated text information.

The definition may include macros.

Example:

```
Comment =Taken from page [PAGENUM] of [TITLE];
```

File Format

EPS | *Tiff* | *TiffM* | *PNG* | *BMP* | *JPEG* | *Progressive* | *ConvertCMYK* | *ConvertLab*

This option selects the file format used to write images to disc.

"TiffM" selects multi-page Tiff output. This format will generate a single file containing multiple images. Though few image viewers support multi-page Tiff, it is commonly used by computer-based facsimile systems.

Use "EPS" for preserving the clip-path and original colour space of embedded images.

Options may be combined together as shown below:

```
File Format = JPEG Progressive;
File Format = TIFF ConvertCMYK ConvertLab;
```

By default Tiff output will retain CMYK or Lab colour spaces unless either of the conversion flags shown above is used.

Greek Limit

number

Text whose size is less than the Greek Limit will not be fully rendered but will instead appear as a grey box (so called "Greeked" text).

As the render resolution is reduced, more a more text will appear Greeked.

Image Quality

1..100

Dictates the compression level used when writing JPEG files. 100% represents the best quality available with the JPEG engine included within Argus and is never loss-less.

Scale Mode, XScale, YScale

These three parameters can be used to specify a range of image scaling operations. "Scale Mode" can be one of the following values:

- None
- Max
- Set
- Percent
- Rez

When scale mode is "**Max**" the values of XScale and YScale specify the maximum allowed pixel width or height of an image. Images with a side greater than either one of these limits will be scaled proportionally to ensure it fits within the bounds.

If either bound is zero, it is ignored. For example,

```
Scale Mode =Max;
XScale =320;
YScale =240;
```

Ensures that images always fit into a rectangle of 320,240 pixels - useful for preparing images for the world wide web.

When scale mode is "**Set**", the values of XScale and YScale specify exact pixel dimensions to which the image will be re-sampled. This allows non-proportional scaling of an image. However, either dimension may be specified as zero, in which case it is scaled proportionally with respect to the other dimension.

For example:

```
Scale Mode =Set;
XScale =320;
YScale =0;
```

Ensures that all images have an exact width of 320 pixels. Images will be enlarged or reduced to fit this measure. The images will retain the correct aspect ratio.

```
Scale Mode =Set;
XScale =320;
YScale =240;
```

Will force an image to fit into 320,240 pixels. This will not preserve the aspect ratio of the image.

When scale mode is "**Percent**" an image can be scaled by a factor. This may be useful for example to reduce the size of all images by 50%.

```
Scale Mode =Percent;
XScale =50;
YScale =0;
```

Reduced the width (& height proportionally) by 50%.

When scale mode is "**Rez**" the XScale and YScale values specify the desired resolution of the output image. In this case, the value of YScale is always ignored - an image cannot have a distinct x & y resolution.

When adjusting resolution, Argus will sub-sample the image to ensure that it remains the same physical size when viewed but consists of more or less pixels.

```
Scale Mode =Rez;
XScale =72;
YScale =0;
```

Ensures all images are subsampled to achieve 72dpi resolution.

[ZONE CONTROL]

In version of Argus prior to 4.2 all zoning was controlled with a single [DOC FEATURES] flag - "Zoning". Later version of Argus allow finer control of the way in which the program zones text.

If Argus reads any configuration file which does not contain a [ZONE CONTROL] section, all zoning is activated irrespective of the setting of the old-style 'Zoning'

feature flag.

Boxes

When true, Argus will detect outlines boxes drawn on the page and prohibit text flow connection which cross the boundaries of the box. It is rare that this flag should be set to false.

Vertical Lines, Horizontal Lines

These flags control the way in which Argus uses any straight lines it finds in a page. When true, a vertical/horizontal line of significant length will be treated as a text flow barrier, that is, Argus will assume text cannot flow across such a lines.

Newspaper pages often use such lines to given visual clues to the reader about the bounds of an article. In this case these flags are of most use.

Some newspaper styles use vertical between each galley of text on a page - effectively revealing to the reader the original galley margins used during page layout. In this case, "Vertical Lines" should be turned off otherwise Argus will fail to reflow text across columns.

[FONT MAP]

Font maps can be used to ensure that characters from fonts which cannot be converted to PDFDoc (the internal character encoding used by Argus) are converted to an alternative form.

For example, the greek letter Delta (Δ) available in the "Symbol" font cannot be represented by a single character in PDFDoc encoding.

However, using a font map it could be converted to "<delta>" for example. It is then a simple matter to change the text output from Argus using the search and replace features of any text editor.

The format of a font map is:

[**FONT MAP**:*font name*]
Serif = *false | true;*
Monospace = *false | true;*
Symbol = *false | true;*
Ignore = *false | true;*
Glyph End = *text;*
Glyph Start = *text;*
Glyph Names = *false | true;*
Preserve Line Breaks = *false | true;*
octal code = text;
d*decimal code = text;*
xhexadecimal code =text;
'a' =text;/* the single ascii character 'a' is the target */
[-- END --]

Using a font map every single character of a font can be mapped to another character or sequence of characters.

The following is a section taken from the "text.cfg" configuration file. It is part of the font map for the Symbol font (mapping Symbol encoding to ASCII).

[FONT MAP: Symbol]
Symbol =true;
d032 = ;
d033 =!;

```
d034 =[.universal];
d035 =#;
d036 =[.existential];
d037 =%;
d038 =&;
d039 =[.suchthat];
d040 =(;
d041 =);
d042 =\x107A;/* Double byte character */
....
[-- END --]
```

Note that the fontname may include the wildcard characters "*" and "?" which match a span and a single character respectively. This is useful for matching against an entire family of fonts such as "Times*" which will catch Time-Roman, TimesBold, TimesBoldItalic etc.

To avoid ambiguity, wildcard fontname matching will always use the longest match found. For instance, given two font maps for "Times*" and "TimesR*", the font "Times-Roman" would match against the "TimesR*" map even though it could also match the "Times*" pattern too.

It is important to note that there should be no space after the '=' sign unless you wish to map a character to 'space'.

Unicode & Double Byte Mapping

Since Argus will normally operate in Unicode mode, it is possible that some fonts will be represented as Unicode rather than their native encodings. This is especially true of Symbol and ZapfDingbats which both have standardised Unicode mappings. In this case many of the entries on the left hand size of a font mapping will be double-byte values. For example:

```
x2021 =[bullet];
```

If left as Unicode, these characters will appear correctly in any output format capable of supporting Unicode (not ASCII for example). However, a problem may occur with the following HTML:

```
<FONT FACE="Symbol">&#2021;</FONT>
```

On Windows platforms in particular, trying to output a valid Symbol font character by referencing its Unicode value will fail to produce the correct character if it is rendered in the Symbol font. The following HTML will work:

```
<FONT FACE= "Times">&#2021;</FONT>
```

The way to avoid this problem is to map the Unicode values of Symbol characters back into the standard Symbol font encoding. This mapping is provided with the distribution and is called "UniSymbol.fmap".

The equivalent mapping for ZapfDingbats is not supplied.

The "Ignore" flag

If the "Ignore" flag is set to true, Argus will discard any characters found in the matching font. This is especially useful for ignoring special fonts used only to provide ornamentation within a document such as a border.

```
[FONT MAP:Symbol]
Ignore = true;
Symbol = true;
```

```
[-- END --]
```

Font Attributes

Within a FONT MAP, a number of font attributes may be declared. These attributes tell Argus about various characteristics of a font and are set to either true or false (the default).

The currently supported attributes are: “Symbol”, “Serif”, “Monospaced”, “Preserve Line Breaks”.

If “Preserve Line Breaks” is set to true Argus will respect the line breaks found in any line of text which ends with a word written in the particular font.

Hence, if a paragraph is written entirely in “Courier” and preserve line breaks is true, Argus will output a line break sequence at the end of each line of text output in the paragraph.

Examples uses of attributes:

```
[FONT MAP: Courier*]
Serif = true;
Preserve Line Breaks = true;
Monospaced = true;
Ignore = false;
[-- END --]
```

The attributes of the standard base 13 fonts (Courier, Times, Helvetica, Symbol and ZapfDingbats) are set by default on start-up. These may however be overruled by an appropriate entry in a configuration file.

Glyph Name Output

It is possible that two different PDF documents may share the same font but the encodings of the font in each case are completely different.

For example, given two documents which make use of the ✂ and ✿ symbols from the “Monotype Sorts” font, it could happen that in one document character code 1 corresponds to ✂ yet in the other, code 2 is used.

This is often as a result of font-subsetting: a process which minimises file size by only including those characters of a font which are used in a document.

Subsetting fonts is not normally a problem. However when the font in question is a glyph font - containing non-readable characters such as Symbol and Zapf Dingbats, re-ordering the font can cause problems.

In order to deal with such instances, a font map would have to be constructed for each document since each may contain a different ordering of characters.

To avoid this, Argus can output the names of the character glyphs directly instead of the character codes.

Any glyph names can then be quickly found and replaced using a simple text editor afterwards.

For example, the line

```
ΑΒΧΔΕΦΓ
```

in “Symbol” font would be output as

```
[.Alpha][.Beta][.Chi][.Delta][.Epsilon][.Phi][.Gamma]
```

A search through the output of Argus for “[.” would quickly find any glyph name. It could then replaced by something more appropriate.

To enable glyph name output for any font use:

```
Glyph Names = true;
```

In the above example, each glyph name was surrounded by "[." and "]". This is the default but can be overridden as follows:

```
Glyph Start =start text;
Glyph End =end text;
```

See "Dealing With Iconic Font Characters" on page 25 for further examples of uses of this feature.

[CHAR MAP]

A character map is used to map Argus internal representation of a character (either Unicode or PDFDoc) into one suitable for the desired output format.

This is useful for mapping characters such as "©" to the proper equivalent in the output format such as "©" in HTML. Or for mapping documents into international encodings such as Shift-JIS, Big-5 or UTF-8.

The format of a mapping entry can take one of four styles:

```
octal char code = text sequence;
ddecimal char code = text sequence;
xhexadeciaml char code = text sequence
'<char>' = text sequence
```

Examples of these four are:

```
'}' =\\};
030 =\x0306;
xfb02 =fl;
d24 =';
```

The numbers on the left hand side correspond to Unicode values (when the "Unicode" Document Feature flag is set) or PDFDoc encoding numbers (see "Font Encodings" on page 80).

For octal entry of a text sequence on the right hand side of the equals sign, type a backslash '\' character followed by three octal digits. For hex entry, use '\x' prior to the four hex digits.

Character mapping is only performed on text extracted from PDF documents and not from text included in any of the macro definitions used throughout such as 'INCLUDE'.

For example, a portion of a typical HTML character map may look like:

```
[CHAR MAP]
326 =&Ouml\;;
325 =&Otilde\;;
323 =&Oacute\;;
322 =&Ograve\;;
321 =&Ntilde;
320 =--;
317 =&Icirc\;;
315 =&Iacute\;;
314 =&Igrave\;;
313 =&Euml\;;
312 =&Ecirc\;;
d65 =\300;
...
[-- END --]
```

Notice the use of an escaped semi-colon to include the HTML semi-colon in the mapping.

Dbl Byte Esc Start
Dbl Byte Esc End

text

When a character is encountered whose value cannot be expressed in the chosen output format, Argus will output the character bracketed by the given escape sequence start and end phrases. In HTML for example, the tokens &#x and ; are used to bracket the hexadecimal representation of a Unicode character.

Use these settings to specify the particular escape tokens used in the chosen output format.

Dbl Byte Format

hex | dec

This setting dictates the way Argus outputs the numerical representation of a character - as either hexadecimal or decimal. HTML unicode characters are specified using hexadecimal. In RTF output, the specification should be in decimal.

This setting and the two above, only come into play when an out-of-range character is being output. For normal output encoding this means any character whose value is greater than 255.

Encoding

Encoding name

When an encoding name is specified, all output is sent through an encoding module which is included in Argus. This module support a wide variety of different output encodings - see “Encoding Names” on page 89 for a complete list of those supported.

To select an encoding, find the name of the encoding in the encoding names appendix and enter it after the equals sign. For example:

Encoding = utf-8;

The MacOSX version of Argus supports None and UTF-8 encodings only.

Removing Line Breaks

Line breaks are always preserved if the “Line Breaks” flag is true. See “Line Breaks” on page 39.

Line breaks may be preserved on a font-by-font basis using the FONT MAP mechanism. See “[FONT MAP]” on page 44 for more information.

[RESET ON]

This section controls the reset behaviour of the text output sub-system of Argus. When reset, the current state of text output is assumed to be plain, unformatted text and the relevant mark-up phrases needed to achieve this state are output. The meaning of each entry is given below:

Reset Modes

MODE	ACTION
Page	Reset at the start & end of each page output
Story	Reset at the start & end of each story output

Reset Modes

MODE	ACTION
Galley	Reset at the start & end of each galley of text - a grouped column of paragraphs
Headline	Reset at the start & end of each headline output
Caption	Reset at the start & end of each caption output
Para	Reset at the start & end of each paragraph output
Table	Reset at the start & end of each table output
Row	Reset at the start & end of each row of each table
Cell	Reset at the start & end of each cell within each table
Pic	Reset mark-up style when a picture link is output
StoryText	Reset at the start of the body text of a story

The default behaviour is for all to be "on" ensuring that whenever a logical section boundary occurs, all mark-up is switched off, then on again. Although this produces verbose output for formats such as HTML, it ensures correct syntax is maintained for more strict formats such as XML.

[LIMITS]

This section allows control over various internal limits governing specific aspects of behaviour such as kerning and paragraph formation. An explanation of the meaning of every limit is beyond the scope of this manual. However the few that are more understandable are documented. The rest are mentioned for completeness only and should not be altered unless under guidance from a member of the technical support staff at Iceni Technology.

MaxWordsInHeading
MinBuffSize
BigGapRatio
BreakOutLines
BreakOutLen
FontSizeDiff
ParaFontDiff
TableSpacingFactor
MinFontSize
MinAnalAccuracy
FontSizeGapRatio

All values integers.

MinFontSize

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

Table Spacing Factor

Dictates the amount of space required between words before Argus assumes that a table column exists. Only required when using table output.

A higher figure (1 or 2 for example) means Argus will create table columns even for small inter-word gaps. This may be useful when dealing with tables formatted in such a way

that the inter-column gap is occasionally very small.

The downside is that it may create columns when none is needed. However this will probably not effect the overall appearance of the table, just it's complexity.

FontSizeGapRatio

A ratio used by Argus 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.

Interactive Forms

[WIDGET]

This section can be used to dictate the output format used for elements of interactive forms embedded within documents. Using a combination of macros, both the form elements and their contents can be output - or any combination of the two.

Form elements are output as they are encountered. This order may not correspond to the reading order of the page - the elements may all be output at the end of a page for example. If this is the case, you may wish to consider enabling sorting of the output. See "XY Sorting" on page 41.

When Argus encounters a form, the formatting specified in the [WIDGET] section is invoked.

For example:

```
[WIDGET]
Start = <FORM>;
End </FORM>;
[-- END --]
```

The above example will output the HTML elements to signal the start and end of an interactive form.

To output individual elements of the form, append the name of the element type to the end of WIDGET: to create a new section. Argus supports six element types named as follows:

[WIDGET:TEXT]
[WIDGET:BUTTON]
[WIDGET:CHECKBOX]
[WIDGET:RADIO]
[WIDGET:LIST]
[WIDGET:COMBO]

Each type of section controls the output of that particular kind of form element (widget).

In order to output anything within these sections, you will have to employ the widget macros created specially for form element output. These are mentioned here but detailed fully in the Macros section of the manual.

WIDGETKEY <keyname>
Outputs the named value obtained from the widget's dictionary.

WIDGETOPTION
Outputs the definition of the current option in a multiple-choice widget such as a list or combo box.

WIDGETINDEX
Identifies the current option within a multiple-choice element.

WIDGETEXPORTVALUE
The value exported by the form element when it is in its 'on' state.

ISEQUAL <a> <b> <true expression> <false expression>
If <a> equals <b> then <true expression> is output. Otherwise the false expression is output.

Using the WIDGETKEY macro you can access any of the numerous values contained within each field element. However, a little explanation of the following values may be of help:

"T" The name of a field. For example "surname" or "age group"

"TU" An optional user-friendly name used for navigation assistance

"V" The value of a field.

For a complete list of all form element keys, see the current version of the official PDF specification available from Adobe's web site.

[WIDGET:TEXT]

The following will output the HTML definition of a text input field. The field will be named as in the PDF and will be pre-filled with any value found in the PDF.

```
[WIDGET:TEXT]
Start    =<INPUT type=text name=[WIDGETKEY T] value="[WIDGETKEY V]">;
End      =</INPUT>;
[-- END --]
```

To output just the contents of the field as text (not as an active HTML form element) use the following:

```
[WIDGET:TEXT]
Start    =[WIDGETKEY V];
End      =;
[-- END --]
```

[WIDGET:BUTTON]

The following example outputs the definition of a interactive button in HTML.

```
[WIDGET:BUTTON]
Start    =<INPUT type=button name=[WIDGETKEY T] value="[WIDGETKEY MK.CA]">;
End      =</INPUT>;
[-- END --]
```

[WIDGET:CHECKBOX]

The following example outputs the HTML definition of a checkbox and checks it if it was checked in the PDF.

```
[WIDGET:CHECKBOX]
Start    =<INPUT type=CheckBox \
         name=[WIDGETKEY T]
         [ISEQUAL [WIDGETKEY V] [WIDGETEXPORTVALUE] CHECKED]>;
End      =</INPUT>;
[-- END --]
```

In this example the ISEQUAL macro is used to ensure the checkbox is checked if the value of the checkbox is "yes".

```
[ISEQUAL [WIDGETKEY V] [WIDGETEXPORTVALUE] CHECKED]
```

This reads as "if (checkbox value) is (its 'on' value') then output 'CHECKED'"

By default the 'on' value for a checkbox is 'yes'. However, this can be set to any text by the form's creator which is why the [WIDGETEXPORTVALUE] macro is used.

If you just want to output the value rather than define a working HTML checkbox then the following would be sufficient:

```
...
Start    =[WIDGETKEY V];
...
```

[WIDGET:RADIO]

Radio buttons are typically grouped so that only one button in the group may be selected. PDF stores each button separately but includes in each a reference to the group to which it belongs. The record for the group itself includes a list of all its members plus the identifier of the currently selected member.

The following will output a definition of a radio button in HTML. It will use the groupname to name buttons so they respond correctly within a group. It will also ensure the cur-

rently chosen button is chosen in the HTML output.

```
[WIDGET:RADIO]
Start    =<INPUT type=radio \
         name=[WIDGETKEY T] [ISEQUAL [WIDGETKEY V] [WIDGETINDEX] CHECKED]>;
End      =</INPUT>;
[-- END --]
```

In this example the ISEQUAL macro is used to ensure only the one button per group is selected.

```
[ISEQUAL [WIDGETKEY V] [WIDGETINDEX] CHECKED]
```

[WIDGETKEY V] gives the id of the group's chosen button. [WIDGETINDEX] gives the id of the current button being output within the group.Hence the ISEQUAL expression can be read as:

"if (group's chosen button) is (this button) then output 'CHECKED'"

[WIDGET:LIST]
[WIDGET:COMBO]

These multiple choice elements differ from all others in that only one element exists but it contains multiple options within it. This means the Start/End formatting will be used only once for the whole element irrespective of the number of options it contains.

To cope with this, these sections have another formatting control called "Middle" which is output for each option.

```
[WIDGET:LIST]
Start    =<SELECT MULTIPLE name=[WIDGETKEY T]>;
Middle   =<OPTION [ISEQUAL [WIDGETKEY V] [WIDGETINDEX] SELECTED]>\
         [WIDGETOPTION];
End      =</SELECT>;
[-- END --]
```

In this example, the 'Start' format defines a new HTML list box with the name given to it in the original PDF form.

The 'Middle' format then causes each option of the list box to be defined.It also ensures that the option chosen in the PDF is also marked as "CHECKED' in the output. This is done using the same technique as was used for radio buttons.

Character Markup

The physical mark-up sections control the text to output when styled text is encountered. Styled text is text that deviates from plain black such as bold, italic or coloured text.

The mark-up sections are:

- BOLD MARKUP
- ITALIC MARKUP
- MONO SPACED MARKUP
- SERIF MARKUP
- SANS SERIF MARKUP
- COLOUR

None of these sections is used when class based markup is enabled. See "Class-Markup" on page 37.

Each section has the following format:

```
Start = <formatting>;
End = <formatting>;
Nesting Level = <number>;
```

The value of the start tag will be output when such styled text is encountered. The end tag will be output either at the end of such a section of text or when nesting rules requires it. Both the tags may include arbitrary text as well as macros.

The nesting Level parameter is a small number indicating the nesting order of the phrases. The outermost nesting level is 0. Higher numbers occupy smaller and smaller nesting scopes. A higher numbered phrase cannot extend beyond the scope of a lower numbered phrase.

Example of HTML mark-up:

```
[BOLD MARKUP]
Start =<B>;
End =</B>;
Nesting Level = 3;
[-- END --]

[ITALIC]
Start =<i>
End =</i>;
Nesting Level =2;
[-- END --]
```

In this example, the nesting level of the italic mark-up phrase is lower than that of the bold mark-up. This means that a section of bold mark-up will not overlap the italic mark-up. Instead of getting output such as this:

plain then <B>**bold, then** **italic**, **bold and** </B>plain

the output is more modular and less open to misinterpretation by HTML browsers:

plain then <B>**bold, then** </B><B>**bold italic**</B><B>**bold and** </B>plain

With careful use of nesting levels, the system can produce output which conforms to quite complex nesting conventions such as those of XML.

Entity Markup

The entity sections control the text that is output at the start and end of discrete parts of a document such as page, paragraph etc.

The sections are:

BOOKMARK	PARA
BYLINE	PIC
CAPTION	ROW
CELL	STORY
DOCUMENT	STORY TEXT
GALLEY	CSS STYLE
HEADLINE	TABLE
PAGE	

Each section has the following format:

```
Start = <formatting>;
End = <formatting>;
```

With the exception of the style section, each of these sections is described below. The style section is rather more complex. See “[CSS STYLE]” on page 57 for a description.

When dealing with a document that has a valid structure tree (a “tagged” PDF), Argus will create new entities from all of the entity names listed in the structure tree. See “Processing Structured Documents” on page 26 for information on accessing these imported entity definitions.

Document, Page

Output at the start and end of each page and each document. When thread-based output is in effect, the behaviour of the Page section changes.

Story

Only occurs when Argus determines that a body of text should be grouped with a headline and byline to form an article or ‘story’.

The ability of Argus to achieve this grouping depends entirely upon the style of document given as input and the setting of the “Article Sorting” flag.

A typical newspaper page may group very well. On the other hand, a book or memo which does not naturally divide into articles would not yield any story groupings. If “Article Sorting” is disabled, no story grouping is done.

Headline, Byline, Story Text

Only generated from a story grouping. Hence only available when “Article Sorting” has been enabled and then only when suitably formatted input is encountered.

Bookmark

Only used when “Bookmarks=true” in the [ARGUS CONTROLS] section - See “Bookmarks” on page 28. The bookmarks for document can form a tree structure with sub-sections containing further bookmarks or further sub-sections (as the bookmarks in this document do).

There are four parts to this section controlling the way the bookmark (outline) tree is output:

Tree Start, Tree End

These specify the formatting to be output at the start and end of the entire bookmark tree and for any nested trees within it.

Entry Start, Entry End
These specify the formatting to output for leaf nodes of the bookmark tree.

An example of how to output a bookmark tree using basic HTML (the list operator) is shown below:

```
[BOOKMARKS]
Tree Start =<UL>;          /* start new (sub) list */
Tree End =</UL>;           /* end of (sub) list */
Entry Start =<LI>;         /* list item - has no sub list */
Entry End =</LI>;          /* end list item */
[-- END --]
```

Caption
When argus has determined that a picture has a related caption, this formatting is used for the caption. Only available when 'Captions' is enabled in the "Document Features" section.

Para
The formatting used to render the start and end of a paragraph. The PARA section has an additional key called "Forced Break". This may be used to define the formatting to output when a visible line-break is to be rendered. This may be used for example, to create paragraph gaps when the "Para Gaps" feature is enabled in the "Argus Controls" section.

Table, Row, Cell
When "Tabulated Output" is enabled, these are the formatting sections used to render the table in the desired output format.

Pic
The formatting to output when a picture is encountered. This is typically used to generate HTML links to image data in the text. Only the 'Start' tag is used within this section, the "End" tag is ignored.

For example, producing an HTML link with the picture section:

```
[PIC]
Start =<IMG SRC='[CLEAN_FILENAME][PAGENUM] -[PIC_NUM].jpg'>;
End =;
[-- END --]
```

Producing paragraph output in HTML:

```
[PARA]
Forced Break =<BR>;
Start =<P>;
End =</P>;
[-- END --]
```

No binary picture data is output by this macro. That is done separately from text output when the "Image Output" flag is enabled in the [ARGUS CONTROLS] section. See "Processing Structured Documents" on page 26

[CLASS SPAN]

This section is used to specify the text to be output at the start and end of a class markup

change. It is used only when “Class Markup” is enabled and when a class catalogue has been constructed using definitions from the [CSS STYLE] section.

When Class Markup is enabled none of the character markup sections are used by Argus. So any definitions for Bold, Italic, Font size etc. are ignored. Instead these effects are rendered using the class markup mechanism.

The only exception to this is super/sub-script markup which is not available at all when using the class mark-up mechanism.

An example of a class span section suitable for use with HTML output is show below:

```
[CLASS SPAN]
Nesting Level =10;
End =</SPAN>
;
Start =<SPAN ID="textStyle[GENSETINDEX]">;
[-- END --]
```

See “[CSS STYLE]” on page 57 for more information on using this mechanism.

[CSS STYLE]

The css style section has been designed to allow creation of a wide variety of style sheets for use in displaying context extracted by Argus.

The distribution includes configuration files which generate style sheets for HTML CSS1 and CSS2 as well as RTF. However, other types of style sheet may be created using the same mechanism.

Four basic style sheets may be created; these are colours, galleys, pictures, fonts. In addition a special style sheet based upon class-based markup can be created.

For example, to create a list of all text colour used within a document, the following section definition could be used:

```
[CSS STYLE]
Path =None;
ColorDef ={\\red[COLOR red]\\green[COLOR grn]\\blue[COLOR blu]\;};
[-- END --]
```

The value of ‘ColorDef’ defines a single row of the eventual colour table. It will be output once for each text colour present in the document.

Though the above example defines the format for a row of a table, to actually output the entire table the INFOTABLE macro is required. See for a full explanation of this and other related macros.

The following is a definition of a RTF [DOCUMENT] section which includes the colour table:

```
[DOCUMENT]
Start =
{\\rtf1
        {\\colortbl
        {\\red0\\green0\\blue0\;}
        [INFOTABLE ColorDef]
        }
};
[-- END --]
```

For the basic four table types, the table output has the same number of rows as there are relevant items. So for a colour table there will be enough rows to catalogue all text colours used in a document. For the font table, enough rows to catalogue every font and size com-

bination etc.

The current “row number” for each table as it is being output is given by the IINDEX macro.

For the special table called “GenSetDef” it will contain as many rows as their are combinations of font, size, colour and text style (bold, italic etc.). Hence, this special table can be used instead of the basic four to catalogue every variation of text used throughout a document - exactly what is required for a well constructed cascading-style-sheet.

See the various configuration files included with the distribution for examples of using this section.

Path

pathname

The full pathname of the output file for style information. One output file will be generated for each document converted by Argus.

The pathname may include macros.

If no external style sheet is required, set Path to “None”.

Example:

```
Path =c:\\argusout\\[FILENAME].css;
```

Format

text

The text to be written in the output file specified by “Path” (see above). When no external file is being created (Path=None) this setting is ignored.

Example:

```
Format =
BODY {font-size: [FONTBASE]px\;}
[INFOTABLE FontDef]
[INFOTABLE ColorDef]
;
```

ColorDef

text

Defines the format text to be output for each font colour used within the document.

The format text will be output numerous times as needed by the INFOTABLE macro.

For example:

```
ColorDef =.cstyle[IINDEX]
{ color: #[COLOR red hex][COLOR grn hex][ICOLOR blu hex]\; };
```

If a document uses three text colours - black, red & white for example, then three colour definitions will be output by expanding the “ColorDef” formatting text three times.

Within the ColorDef format, normal document macros may be used.

FontDef

text

Defines the format text to be output for each font and font size combination used within the document.

The format text will be output numerous times as needed by the INFOTABLE macro.

For example:

```
FontDef =.fstyle[IINDEX]
{
        font-family: [FONTNAME], [HTMLFONTFAMILY]\;
        font-size: [FONTSIZE]px\;
        line-height: normal\;
}
;
```

Assuming a document consists of two fonts and uses each at two different sizes, the INFOTABLE macro will expand the format of FontDef 4 times - once for each font and font size combination.

Within the FontDef format, normal document macros may be used.

GalleyDef
ParaDef

text

Defines the format text to be output for each galley or paragraph of text created on the current page being output.

A galley is the name given to a vertical group of paragraphs. In a single column essay-style document for example, each page would probably be treated as a single galley. In a newspaper style document, each page would probably contain numerous galleys corresponding to the columns of text in the original printed version.

Since the GalleyDef text is output for each galley on a page, there needs to be a current page in scope when the text is parsed. This means that the GalleyDef format text cannot be invoked in places which do not implicitly define a current page.

Hence you cannot use the GalleyDef text in the [DOCUMENT] section or the [CSS STYLE] section “Format” text.

For example, the following will produce a “Not In Scope” error:

```
[DOCUMENT]
Start =[INFOTABLE GalleyDef];
End=;
[-- END --]
```

Example usage of GalleyDef:

```
GalleyDef =.gstyle[PAGENUM]X[IINDEX]
{
        position: absolute\;
        left: [GALLEYPOSX]px\;
        top: [GALLEYPOSY]px\;
        width: [GALLEYWIDTH]px\;
        height: [GALLEYHEIGHT]px\;
}
;
```

This is the HTML4 definition required for exact positioning of galleys on a page.

Within the GalleyDef format, normal document macros may be used.

PicDef

text

Defines the format text to be output for each image box created on the current page being output.

The format text will be output once for each image in the current page by the INFOTABLE macro.

Since the PicDef text is output for each image on a page, there must be a current page in

scope when the text is expanded. This means that the PicDef text cannot be invoked in places which do not implicitly define a current page.

Hence you cannot use the PicDef text in the [DOCUMENT] section or the [CSS STYLE] section “Format” text.

Example usage of PicDef:

```
PictureDef =.pstyle[PAGENUM]X[IINDEX]
{
        position: absolute\;
        left: [POSX]px\;
        top: [POSY]px\;
        width: [WIDTH]px\;
height: [HEIGHT]px\;
        clip: rect(0, [WIDTH], [HEIGHT], 0)\;
}
;
```

This is the HTML4 definition required for exact positioning of pictures on a page.

Within the PicDef format, normal document macros may be used.

The macros returning width and height detail the visible area of the image. That is, the portion of the image remaining after clipping. This may be different from the actual size of the picture stored in the document.

GenSetDef

t*ext*

Defines the text to output on each row of the class-based table. The INFOTABLE macro will output one row for each font, size, colour & style combination found in the document.

This table is intended for use in constructing class-based style sheets such as those required for the CSS2 format.

The following example shows a definition suitable for use in a CSS1 style sheet:

```
GenSetDef =
#textStyle[IINDEX] {
font-size: [FONTSIZE %]%\;
[IGISBOLD "font-weight: bold\;"]
[IGISITALIC "font-style: italic\;"]
[IGISSUPSCR "vertical-align: super\;"]
[IGISSUBSCR "vertical-align: sub\;"]
[IGISBASECOL "*" "color: #[COLOR]\;"]
};
```

See the html configurations css1.cgf and css2.cfg included with the distribution for further examples of using this kind of table.

INFOTABLE

format-name

This macro is the mainstay of the style output mechanism. Given the name of a format such as “ColorDef”, the macro will output the format text for each item relevant to that format.

The currently supported format names are

- GalleyDef
- PicDef
- ParaDef
- FontDef
- ColorDef
- GenSetDef

These table definitions are described in the proceeding pages.

[FONT BANDS]

This section allows the definition of size-bands representing ranges of font sizes. All text encountered within the PDF will be categorised into these bands on the basis of font size, with band 1 representing the largest font size in the document. The normal band is deemed to represent the body-text of a document or page.

The format of the section is:

[FONT BANDS]
Normal Band = *number;*
Number of Bands = *number;*
Font Band*N* Nesting = *number;*
Font Band*N* Start = *formatting text;*
Font Band*N* End = *formatting text;*
[-- END --]

For example, given that simple HTML output is required, a number of bands may be used to map input fonts to the standard H1, H2 etc. of HTML:

```
[FONT BANDS]

Normal Band = 3;
Number of Bands = 4;
Font Band1 Nesting = 0;
Font Band1 Start = "h1";
Font Band1 End = "/h1";
Font Band2 Nesting = 0;
Font Band2 Start = "h2";
Font Band2 End = "/h2";
Font Band3 Nesting = 0;
Font Band3 Start = "";
Font Band3 End = "";
Font Band4 Nesting = 0;
Font Band4 Start = "<SMALL>";
Font Band4 End = "</SMALL>";
[-- END --]
```

If this style of font size classification is not required, setting the number of bands to 1 with a font information macro as the format will reproduce the fonts used in the original document.

For example:

```
[FONT BANDS]
Normal Band = 1;
Number of Bands = 1;
Font Band1 Nesting = 0;
Font Band1 Start = "<FONT SIZE=[FONTSIZE html]>";
Font Band1 End = "</FONTSIZE>";
[-- END --]
```

See the description of the FONTSIZE macro in the macros section for details of what it can do.

USING MACROS

In many of the sections detailed previously, it has been noted that macros may be included in the formatting text. These macros represent a convenient way of enriching the information output by the system.

The format for a macro is:

[**macro-name** <*arguments*>]

where <macro name> is one of those given in the table below and <arguments> is any number of arguments (depending upon the macro) or another, nested macro. In general macro names are case sensitive so care must be taken when typing them.

Multiple arguments are separated by whitespace, so to include a space within an argument, enclose it within single or double quotes.

To include the open bracket symbol itself in output, use the sequence '[[' - double open bracket. This is not necessary for the closing bracket which may appear on its own in any case.

This example show the definition of a simple information header at the top of each page:

```
[PAGE]
Start =Page [PAGE], extracted from [TITLE]
On [TODAY '%Y, %M, %d']
Bounding Box [[[POSX] [POSY] [WIDTH] [HEIGHT]]
;
End =---------- END PAGE ----------
;
[-- END --]
```

Setting up HTML-style links to images in text output:

```
[PIC]
Start =<IMG HREF='[FILENAME]-[PAGE]:[IMAGENUM].JPG'>;
End =;
[-- END --]
```

In the following pages is a brief description of every macro available. More information is presented after the summary.

Documentation Conventions

In the detailed descriptions that follow, optional macro parameters are signified using <> .

Arguments shown within quotes are literal and when used should be typed as written.

Utility Macros

CHCASE

"upper" | *"lower"* | *"title"* | *"off"*

Controls the case change filtering associated with the currently active output map. By default the filter is "off" and no changes are made. Giving any of the other three parameters will arm the filter causing all subsequent text extracted from the document and output using the current output map to be converted to the required case.

For example, to ensure that all story headlines appear in upper case:

```
[HEADLINE]

Start =[CHCASE upper];
End =[CHCHASE off];
```

```
[-- END --]
```

COUNTER

<“set” value> | <+/-value>

Without any arguments, returns the value of a counter (initially zero). The value of the counter may be reset using the “set” argument and inc/decremented by supplying an integer value.

FTRUNC

filename

Ensures that the given filename is short enough to be valid under a Macintosh system. Under MacOS Classic, filenames must be 32 characters or less.

HEAD

pathname

Returns the given pathname up to the final component - the filename. See TAIL.

For example:

```
[HEAD /one/two/three/four.txt]
```

yields:

```
/one/two/three
```

INCLUDE

filename <“true”>

Includes the contents of the specified file in the output of the current document. The filename argument may be an absolute path or relative to the current working directory. For convenience, the filename parameter can itself be made up from other macros.

If the named file does not exist or cannot be read, the default behaviour is to continue without error. Adding the true argument will force an error in this situation.

For example, given a PDF with a filename of “manual. pdf”, the following line would instruct the system to read a file called “descriptions/ manual.txt” from the current working directory:

```
[INCLUDE ‘descriptions\[FILENAME].txt’]
```

There is no upper limit on the amount of text that will be read from a file in this manner. All text included in this way bypasses the character mapping stage.

MARKUP

“on” | “off”

Controls the output of text styling phrases (mark-up phrases). By default out is enabled (is “on”). When disabled by issuing this macro with “off” as its argument, it remains disabled until a corresponding “on” is encountered.

When off, no text styling is output at all. This includes font size and colour information as well as styling such as bold and italic.

For example, to ensure that story headlines are output with no applied styling:

```
[HEADLINE]
Start =[MARKUP off];
End =[MARKUP on];
[-- END --]
```

MEDIACHANGE
true <false>

Returns the true or false parameter depending upon whether or not the page size has changed from one page to the next.

If the true parameter is "*" it is ignored.

REPEAT
count text

Outputs the given 'text' 'count' times.

SECTION
"on" | "off"

Controls the output of section phrases such as paragraph, caption and table sequences. Behaves in a similar manner to MARKUP.

TAIL
filename

Returns the final component of a pathname - the filename. See HEAD.

For example:

```
[TAIL /one/two/three/four.txt]
```

yields:

```
four.txt
```

TODAY
TOMORROW
YESTERDAY
<format>

Without the optional format parameter, the relevant date is output in a numeric day/ month/ year format (with the slashes). The format of the date is dependent upon the platform on which the system is running.

Use the format parameter to specify an alternative style of output.

VALUE
key

Returns the value of the given user-defined key. Keys may be defined using the Argus SDK call *icnArgusValueAdd***.** Please see the SDK reference manual for details of creating macro keys.

Document Information Macros

AUTHOR
CREATED
MODIFIED
<format>

CREATOR
KEYWORDS
PRODUCER
SUBJECT
TITLE

Causes the respective document information dictionary entry to be output. These keys represent the same information as is displayed in the “General Document Info” dialogue found in the File menu of Acrobat Exchange.

Note that although a standard date format is defined in the PDF Mark Reference Manual, some document generators adopt their own format (often plain English). The upshot being that when such a non- standard date is encountered, the plug- in will not be able to re-format it and will ignore any format parameter given. This only applies to the date producing macros, MODIFIED and CREATED.

To obtain values from non-standard keys within the document information catalogue, use the KEY macro - see page 66.

CLEAN_FILENAME

The filename of the current PDF with spaces replaced by underscores. Provided for use in creating image links since some web browsers will not allow image filenames to contain space characters.

FILENAME

Outputs the final component of the path name of the current PDF minus the “.pdf” extension.

KEY

keyname

Given a key name will return associated value in the document’s information dictionary. The dictionary (if present) holds information such as creation date, author, title etc. and is normally accessed using the AUTHOR, TITLE, SUBJECT... macros.

Use this macro to access other, non-standard key values - perhaps those added during some special production process.

PAGENUM

<offset>

Outputs the current page number being processed. Numbering starts at 1 but can be modified with the optional offset parameter. This may a positive or negative integer.

PAGELABEL

Outputs the label or logical page number given to a page. From PDF version 1.3 onwards pages can have a label as well as their implicit physical page number.

For example, the initial pages of a document may have page labels in Roman numerals - ii, iii, iv, etc. with all following pages being numbered using Arabic numerals - 5, 6, 7... etc.

The output of this macro is not always a number but can contain text as defined by the PDF creator. If a page has no label defined, no output is generated.

PAGECOUNT

Returns the number of pages in the document.

PATHNAME
Outputs the full path name of the current input pdf without any modifications.

THREAD
Outputs the name of a the current thread. The name of a thread is defined by the application creating the PDF or by a user from within Acrobat using the thread tool.

This macro is available within the DOCUMENT and PAGE entity sections only when outputting in thread-based mode. See “Output Threads” in “Argus Controls”.

THREAD_TITLE
THREAD_SUBJECT
THREAD_AUTHOR
THREAD_KEY
<keyname>

Obtains the Title, Subject and Author values form the current thread’s information dictionary.

To access any other non-standard key value from the dictionary use ‘THREAD_KEY’ which requires the key name as its argument.

THREAD_ID
Returns the index number of the current thread. This number is normally assigned to the thread by the application that created it.

Font & Character Information Macros

COLOR
<“red”|”grn”|”blu”> <“hex”| “%” | “dec”>

Takes two parameters the first of which can be red, grn or blu (for red green or blue) and a second parameter that can be hex, “%”.

The second parameter instructs the macro to display its result as a hexadecimal (00..ff), decimal (0..255) or percentage value (0..100). The default when nothing is specified is hexadecimal.

Default setting with no parameters is to output the concatenated values of red, green and blue (in that order) as hexadecimal.

FONTFAMILY
FONTBASEFAMILY
< “html” | “rtf” >

FONTFAMILY returns an approximate value for the family name of the current font. FONTBASEFAMILY returns the family for the most common font in the document.

The family name is derived from the font name by using only the first complete word of the name.

For example,

a font name of “Minion-Bold” returns a font family of “Minion”
a font name of “Swiss227” returns a font family of “Swiss”

Though this may not be the actual name of the font family, in many cases it will be close

enough.

When the “html” argument is supplied, the result is one of the standard font class names defined for HTML - “Serif”, “Sans-serif”, “monospace” and “symbol”

Example font band definition:

```
[FONT BANDS]
Font Band1 Nesting =0;
Font Band1 End =</FONT>;
Font Band1 Start =<FONT FACE="[FONTNAME], [FONTFAMILY], [FONTFAMILY html]"
SIZE="[FONTSIZE html]">;
One Size Per Word =false;
Normal Band =1;
Number of Bands =1;
[-- END --]
```

Similarly when the “rtf” argument is supplied, the result is one of the standard class names used in RTF - “nil”, “roman”, “swiss”, “modern”.

FONTNAME

<“*index*”>

Returns the name of the current font. This is the complete name of the font including any bold or italic nomenclature.

If the optional word “index” is supplied, then instead of the font name being produced an index number is given. This index corresponds to an entry in the font table held within the system for the pages so far analysed by the system. The font table itself can be output in an RTF compatible format using the FONTTABLE macro.

FONTSIZE

<“rtf” | “html” | “%”> <“adjust”>

FONTBASESIZE

<“*rtf*” | “*html*” | “*%*”>

With no arguments, FONTSIZE returns the size of the current font rounded off to the nearest whole Point (Postscript Point). If one of the three optional conversion parameters is supplied, the format of the size returned changes as follows:

WORD	OUTPUT FORMAT
html	A number between 1 and 7 is returned. 1 representing the smallest font size, 7 the largest and 3 size regarded as “normal” for the document. This is the format expected by the <FONT> tag in HTML 3.2
rtf	Font size in twips suitable for use in RTF output
%	Font size as a percentage of the base-font size for the document. The base-font is the font deemed to be the most commonly occurring font/size combination in the document.

When the optional literal argument “adjust” is supplied, the resulting size is modified in order to cap the size of very large fonts. This may produce better results when attempting to preserve layout.

FONTBASESIZE returns the size of the most common font in the document. In this case the “adjust” argument is nonsense and should not be used.

HTMLFONTFAMILY

Returns the generic name for the family of the current font. Names returned are suitable for use within HTML FONT tags - sans-serif, serif, monospace.

LEADING

<"in" | "cm" | "mm" | "tw" | "%" | "em" | "rtf" | mult>

Returns an approximate value for the leading (inter-line gap) between lines of text in a paragraph. Optional unit conversions are inches (in), centimetres (cm), millimetres (mm), twips (tw), percentage of font size (%), printer's "em" (em) and rtf compatible units (rtf).

An optional numerical scaling factor (mult) can be supplied.

SYLKFORMAT

Returns an index into a font format table for use with SYLK based output. See sylk.cfg for an example of its usage.

Geometry Macros

ALIGN

left centre right full

Given up to four arguments, outputs the argument corresponding to the text alignment of the current paragraph.

For example, to correctly align a table cell in HTML:

```
<TD align=[ALIGN left centre right]>
```

When processing a document with structure tags, Argus will examine the TextAlign attribute from the Layout tag.

For non-structured documents, the result is always 'left'.

INDENT

text

Outputs the given text as many times as their are characters in the current paragraphs first line indent. A typical usage would be INDENT " " to output the space character a number of times.

INDENTSIZE

<multiplier>

Output the size of the first line indent of the current paragraph. Is typically used in conjunction with the INDENT macro to render first-line indents in text.

The <multiplier> can be used as a numeric scale factor.

MARGINSIZE

"left" |"right" *<mult>*

Returns the size of either the left or right margin of the current paragraph. The margin is computed by subtracting the left or right extent of the paragraph from the corresponding edge of the containing galley.

The first parameter (which is mandatory) dictates which margin is returned - left or right side of the paragraph. The second parameter is optional and allows for a scaling value.

This may be used for example to convert the result to twips by supplying a scale of 20.

WIDTH **PAGEWIDTH**
HEIGHT **PAGEHEIGHT**
POSX **PAGEPOSX**
POSY **PAGEPOSY**
POSYR

<"cm" | "mm" | "tw"| "in"><"clip"><"adjust"><mult>

When no unit arguments are given, the return value will be Postscript Points - approximately 1/72 of an inch.

The "PAGE..." macros return information about the current page's crop box. The origin for measurement is 0,0 - the bottom-left of the page media.

POSX is the absolute left hand side of the current object.

POSYR is the absolute coordinate of the bottom of the current object. The origin of its coordinate system being 0,0 - the bottom-left of the page media.

To obtain x,y coordinates relative to the crop use the following constructs:

```
[ADD [POSX] -[PAGEPOSX]]
[ADD [POSY] -[PAGEPOSY]]
```

POSY is the distance from the top of the page's crop box to the top of the current object - this is useful for generating HTML positioning information and is maintained for backwards compatibility with older versions of Argus.

POSY is equivilent to:

```
[ADD [PAGEHEIGHT] -[ADD [POSY] [HEIGHT]]]
```

The results may be converted to other units via the optional arguments: cm - centimetres, mm - millimetres, tw - twips, in - inches.

The optional "adjust" argument causes the width result to be modified in such a way as to yield better results when preserving page layout. It should not be used if accurate measurements are required and has no effect on the "PAGE..." macros.

An optional numeric scale factor may be supplied - the <mult> argument.

When in the scope of an image, the WIDTH and HEIGHT returned relate to the entire bbox of the image not just the visible portion (though in some cases they will be the same). To obtain the bbox of just the visible portion use the "clip" argument.

Hyperlinks & Bookmarks

BOOKMARK_TITLE

Outputs the title of the current bookmark. Only available within the BOOKMARKS entity section. Can be used to help produce a formatted list of bookmarks with their titles, hypertext linked to the relevant place within a document.

HYPERDEST
IFHYPERDEST

These macros are generally used together to deal with the instance of a hyper link itself being the destination for another hyperlink. Typical usage would be:

```
<a href=[URL] [IFHYPERDEST 'name=[HYPERDEST]'>
```

which allows a href to include a name for parts of the page that are both links and destinations. This happens for example with the Quick Start Guide, page 1, when synth bookmarks is on.

NESTING
Returns the nesting level of the currently active bookmark.

PAGEREF
The page number of the destination page of a hypertext link. This macro is available for use within the HYPERLINKS entity section when bookmarks or hyperlinks are enabled (See “Argus Controls”). See the configuration files html.cfg and htmlLink.cfg for example usage.

Example:

```
Src Start =<a href="page[PAGEREF].html#[TARGET]">;
```

TARGET
The name of a hypertext reference within a document. This name is typically a unique number generated by Argus. The number is unique within the document.

This macro is available within the HYPERLINKS entity section. For an example of its usage see html.cfg.

Example:

```
[HYPERLINKS]
Dest End =</a>;
Dest Start =<a name="[TARGET]"><B>*[TARGET]*</B>;
Src End =</a>;
Src Start =<a href="page[PAGEREF].html#[TARGET]">;
[-- END --]
```

URL
RTF_URL
Gives the url of the currently active bookmark. RTF_URL returns the same information in a manner suitable for inclusion in an RTF document.

BYLINE
Outputs the author byline found within the current story. Only available when “Article Sorting” is enabled within the “Document Features” section.

The byline is located within a story by a combination of examination of the story layout and cross-reference of the byline lookup table contained in the file specified by “Byline Path” in “Argus Controls” section.

Table Macros

CELLWIDTH
Width of the current cell in characters using the base font size.

COLINDEX
The distance from the left table edge to the right of the current column.

COLRIGHT

<mult>

The distance from the left table edge to the right of the current column in points. An optional numeric multiplier may be provided to scale the result.

COLSPAN

Gives the number of columns spanned by the current cell in the table. This is a number from 1 to the number of columns in the table.

See the table definitions within html.cfg for an example of this and other table-specific macros.

ISEMPTY

true <false>

Returns the true or false argument supplied depending on whether the current cell is empty.

If the truth argument is "*" it is ignored.

ISINTABLE

intable outtable

Outputs the 'intable' text if there is an active table, otherwise 'outtable' is output.

May be used for example to achieve different paragraph formatting when within a table cell as apposed to normal body text.

ISROWSPAN

true <false>

Returns the true or false argument supplied depending on whether the current cell spans more than one row.

Use the ROWSPAN macro to determine the actual number of rows being spanned.

ISTABBORDER

true <false>

Returns the true or false argument supplied depending on whether the current table is considered to have borders.

NUMCOLS
NUMROWS

Returns the number of columns/rows in the current table. If no table is currently in scope, an error occurs.

The number of columns of computed from the maximum number of cell spans within a row not just the number of cells in each row.

ROWINDEX

Returns the row number of the current row.

ROWSPAN

Gives the number of rows spanned by the current cell. Typically this value is 1. Only available within the ROW entity section.

RTFCOLORTABLE

Outputs a definition of a colour table compatible with the RTF specification. This should be output at the start of a document to define all of the colours used for text throughout the document.

In order to reference the table within the text, use the indexed form of the COLOUR macro.

See the rtf.cfg configuration file for an example of its use.

RTFFONTTABLE

Produces a table listing each font used within a document. The format of the table is compatible with the RTF format. The table should be output at the start of an RTF document.

In order to reference the table within the text, use the indexed version of the FONTNAME macro.

Style Sheet Construction & Reference

COLORINDEX

<offset>

Returns the index number of the current color definition. This is an index into the colour table generated by the RTFCOLORTABLE macro. It will also correspond to the relevant row number for tables output via the ColorDef format in the [CSS STYLE] section.

An optional numeric argument can be supplied which will then be added to the index.

```
[COLOUR]
Nesting Level =2;
End =\\cf0 ;
Start =\\cf[COLORINDEX 1] ;
[-- END --]
```

FONTINDEX
FONTBASEINDEX

<offset>

FONTINDEX returns the index number of the current font. This number will correspond to a row number in the table output via the FontDef specification in the [CSS STYLE] section. See “[CSS STYLE]” on page 40

An optional numeric argument can be supplied which will then be added to the index.

```
[FONT BANDS]
Font Band7 Nesting =1;
Font Band7 End =};
Font Band7 Start =\\s[FONTINDEX 1]\\f[FONTNAME index]\\fs[FONTSIZE rtf] {;
...
```

The FONTBASEINDEX macro outputs the index of the most common font within a document.

GALLEYINDEX

<offset>

Returns the index number of the current galley. This number will correspond to a row number in the table output via the GalleyDef specification in the [CSS STYLE] section. See “[CSS STYLE]” on page 40.

The optional numeric offset argument will be added to the return value.

GENSETINDEX

Returns the index number of the current class-based style. This number will correspond to a row number in the table output via the GalleyDef specification in the [CSS STYLE] section. See “[CSS STYLE]” on page 40

IGISITALIC
IGISBOLD
IGISMONOSPC
IGISBASECOL

true <false>

These macros may be used during construction of the class-based style sheet table “GenSetDef”. They are out of scope at all other times.

Each macro performs a test on the currently active style’s attributes and return either the true or false argument supplied dependent upon the result.

For example:

```
GalleyDef=[IGISITALIC “<i>"];
```

would cause the truth argument “<i>” to be output whenever the style active on output of a table row was italic.

The IGISBASECOL macro outputs the truth argument whenever the current style’s text colour is the same as the most common font’s text colour - that it, the same colour as the majority of text in the document.

If the truth argument is set to “*”, it is ignored and only the false argument is output when appropriate.

IINDEX

<offset>

Returns the row number of the table being constructed. This macro is only in scope during table construction and should not be used otherwise.

The optional argument is an offset which is added to the result.

IMAGEINDEX

formatName

Returns the index number of the current image. This number will correspond to a row number in the table output via the PictureDef specification in the [CSS STYLE] section. See “[CSS STYLE]” on page 40

INFOTABLE

rowDefinitionName

This macro causes a style sheet table to be created using the named row format. The number of rows generated for the table will depend upon the type of table and the current document. For instance, the ColorDef table will contain one row for each different colour used for text in the current document.

The values of formatName can be any one of:

- GalleyDef,
- PicDef,
- ParaDef
- FontDef
- ColorDef.
- GenSetDef

Depending upon the kind of table being output, care must be taken on the placement of this macro. The GalleyDef, PicDef and ParaDef tables can only be created when a page is being analysed. This means for example, that requesting their creation within the [DOCUMENT] section (which is output before any page analysis is done) would cause a scope error.

See “[CSS STYLE]” on page 40 for information on constructing style tables using this macro.

PARAINDEX

Returns the index number of the current image. This number will correspond to a row number in the table output via the ParaDef specification in the [CSS STYLE] section. See “[CSS STYLE]” on page 40

Image Macros

BYTES_PER_LINE

Returns the number of bytes of data in a line of pixels from the current image.

CAPTION

<“left” | “right” | “top” | “bottom” | “inside”>

Returns one or more captions associated with the current picture. If a position qualifier is supplied (top, bottom etc.) then only the captions in the given position will be returned. If no qualifier is given, all captions will be returned for the current picture.

Note that this macro will only work when used within the[PIC] section since this is the only time when a picture is active. Used at any other time, it will produce an error.

For example, to output a picture as part of a table with all of its captions:

```
[PIC]
Start =<TABLE><CAPTION>[CAPTION]</CAPTION>
<TR><TD><IMG SRC='[CLEAN_FILENAME]-[PIC-COUNT].jpg'>
</TABLE>;
End =;
[-- END --]
```

NUM_BYTES

Returns the total number of bytes in the current image.

OPI

key

Returns the value of the given key in the OPI dictionary of the current image. If the key is not found or the image has no OPI information dictionary, nothing is returned. This macro is used in the imageConfigs/opieps.cfg file.

Example:

```
[PIC]
Start=OPI Information for image '[OPINAME "image [PIC_NUM]"]':
Version          : [OPI Version]
File-Spec        : [OPI F]
ID               : [OPI ID]
CropRect         : [OPI CropRect]
Size             : [OPI Size] ([PIX_WIDTH] x [PIX_HEIGHT] pixels)
Comments         : [OPI Comments]
CropFixed        : [OPI CropFixed]
Position         : [OPI Position]
| Resolution     : [OPI Resolution]
ColorType        : [OPI ColorType]
Color            : [OPI Color]
Tint             : [OPI Tint]
Overprint        : [OPI Overprint]
ImageType        : [OPI ImageType]
GrayMap          : [OPI GrayMap]
Tags             : [OPI Tags]
Transparency     : [OPI Transaparency]
;
End=;
[-- END --]
```

The values returned by this macro are representations of the PostScript objects held within the dictionary.

OPINAME

default

Returns a filename representing the original filename of the current image as stored in its associated OPI dictionary. If no filename can be found or there is no OPI dictionary, the 'default' text is output instead. Using this technique it is possible for example to output images with their original names when found or a default name when not.

Example:

```
Image Path =docs/[FILENAME]/page[PAGENUM]/[OPINAME [PIC_NUM]][PIC_EXT];
```

This macro uses the values of either the 'ID' for 'F' keys to arrive at a name.

PIC_COUNT

Unique ID assigned to each image throughout the document being processed.

PIC_DEPTH

The bit depth of the color channels in the current image. For bitmap images depth will be 1, for most other formats the depth will be 8.

The number of color components can be inferred from the COLORSPACE value.

PIC_EXT

Returns a file extension suitable for the current image including the period. For example ".jpg" for a jpeg format image.

PIC_NUM

This is a counter, incremented by one as each image is processed. It is reset at the end of each page in the document It is provided as a means for generating unique image names when outputting images.

PIC_REZ

Returns the x & y resolution of the current image measured in dots per inch. For example, an image that is 250.2 dpi(x) by 266.7 dpi(y), the result would be:

250x267

PIC_SPACE

Returns the name of the colour space of the current image as defined in the PDF Red Book. The colourspace names are:

Color Space Names

NAME	CHANNELS	NOTES
Mask	1	Bitmap
Indexed	-	Indexed color space
DeviceGray	1	Grayscale
DeviceRGB	3	
DeviceCMYK	4	
CalGray	1	Calibrated Grayscale
CalRGB	3	Calibrated RGB
CIELAB	3	Device independent colour space

PIX_WIDTH
PIX_HEIGHT

Return the width and height of the current image in pixels.

Interactive Forms Macros

For examples of how the following macros may be used, see “See the description of the FONTSIZE macro in the macros section for details of what it can do.” on page 61.

WIDGETKEY

keyname

Returns the named values from the current form element’s dictionary. Consult the PDF specification for a list of standard key names.

Some key names which may be commonly required are:

- “T” The name of a field. For example “surname” or “age group”
- “TU” An optional user-friendly name used for navigation assistance
- “V” The value of a field.

To access a value within a sub-dictionary of a form element use the notation

<keyname>.<keyname>

WIDGETINDEX

Returns the id of the currently active form element. This id may be a digit or a piece of text depending upon how the form was designed.

WIDGETOPTION

Returns the value of the current sub-element of a multi-option form element. Used for dealing with list and combo box elements which commonly have multiple sub-options.

This macro is only in scope within the ‘Middle’ section of list and combo box form elements.

For example:

```
[WIDGET:LIST]
Start     =<SELECT MULTIPLE name=[WIDGETKEY T]>;
Middle  =[WIDGETOPTION];
End       =</SELECT>;
[-- END --]
```

WIDGETEXPORTVALUE

Returns the value of the widget that would be output if it were currently selected. Note that this is not the same as the current value of the element.

For example, a checkbox with the values of ‘yes’ and ‘no’ may be currently ‘off’ which means its value is ‘no’ though its export value (the ‘on’ value) is still ‘yes’.

ISEQUAL

a b onTrue <onFalse>

If the values of *a* and *b* are the same, the macro will copy the *onTrue* argument to output otherwise it will copy of value of the *onFalse* argument (if supplied).

This macro is general purpose but is mentioned in this section since it is particularly useful when dealing with interactive form elements.

Formatting Dates

Dates are formatted by using special place- holder characters, each corresponding to a different feature of the date or time. Arbitrary format sequences can be built using the % symbol to form a complete format specification. Characters not proceeded by %, will be output as is.

For example, assuming the date is 14th August, 1997:

```
%Y%m%d 19970814
%a %B %d, %Y Thu August 14, 1997
the %jth day of %Y the 227th day of 1997
```

It is essential to supply quotes around date formats because the whole format is represents a single argument to the macro.

For example,

```
[CREATED "%a %B %d, %Y"]
```

The following table lists all of the available date and time formatting characters.

Date Formatting Symbols

SYMBOL	DESCRIPTION
%a	Abbreviated weekday name
%A	Full weekday name
%b	Abbreviated month name
%B	Full month name
%c	Date and time representation appropriate for locale
%d	Day of month as decimal number (01 - 31)
%H	Hour in 24- hour format (00 - 23)
%I	Hour in 12- hour format (01 - 12)
%j	Day of year as decimal number (001 - 366)
%m	Month as decimal number (01 - 12)
%M	Minute as decimal number (00 - 59)
%p	Current locale's A. M./ P. M. indicator for 12- hour clock
%S	Second as decimal number (00 - 59)
%U	Week of year as decimal number, with Sunday as first day of week (00 - 51)
%w	Weekday as decimal number (0 - 6; Sunday is 0)
%W	Week of year as decimal number, with Monday as first day of week (00 - 51)
%x	Date representation for current locale
%X	Time representation for current locale
%y	Year without century, as decimal number (00 - 99)
%Y	Year with century, as decimal number
%z, %Z	Time- zone name or abbreviation; no characters if time zone is unknown
%%	Percent sign

Font Encodings

The table below lists the standard font encodings used in PDF document- PDFDoc, MacRoman and WinAnsii.

These tables may be useful when running Argus in PDDoc mode. They will be of less use when Argus is running in Unicode mode - see “[CHAR MAP]” on page 47 for more information.

When not running in Unicode mode, Argus represents all characters internally using PDFDoc encoding. This mean that the numbers on the left hand side of a CHAR MAP should refer to PDFDoc encoding character positions. Similarly, the text on the right hand side of a FONT MAP entry should be in PDFDoc encoding.

Some characters in PDFDoc cannot be typed on a normal keyboard. Use octal or 2-byte hexadecimal notation to enter these.

For example:

```
[CHAR MAP]
d30 =(ring);
d139=%thousand;/* Maps ‰ to Ascii
[-- END --]

[FONT MAP:Symbol]
d045 =\212;/* Maps char 45 to ‘minus’ */
[-- END --]
```

DECIMAL	OCTAL	PDFDOC	MACROMAN	WINANSII
24	30	breve	—	—
25	31	caron	—	—
26	32	circumflex	—	—
27	33	dotaccent	—	—
28	34	hungarumlaut	—	—
29	35	ogonek	—	—
30	36	ring	—	—
31	37	tilde	—	—
32	40	space	space	space
33	41	exclam	exclam	exclam
34	42	quotedbl	quotedbl	quotedbl
35	43	numbersign	numbersign	numbersign
36	44	dollar	dollar	dollar
37	45	percent	percent	percent
38	46	ampersand	ampersand	ampersand
39	47	quotesingle	quotesingle	quotesingle
40	50	parenleft	parenleft	parenleft
41	51	parenright	parenright	parenright
42	52	asterisk	asterisk	asterisk
43	53	plus	plus	plus
44	54	comma	comma	comma
45	55	hyphen	hyphen	hyphen
46	56	period	period	period
47	57	slash	slash	slash
48	60	zero	zero	zero

DECIMAL	OCTAL	PDFDOC	MACROMAN	WINANSII
49	61	one	one	one
50	62	two	two	two
51	63	three	three	three
52	64	four	four	four
53	65	five	five	five
54	66	six	six	six
55	67	seven	seven	seven
56	70	eight	eight	eight
57	71	nine	nine	nine
58	72	colon	colon	colon
59	73	semicolon	semicolon	semicolon
60	74	less	less	less
61	75	equal	equal	equal
62	76	greater	greater	greater
63	77	question	question	question
64	100	at	at	at
65	101	A	A	A
66	102	B	B	B
67	103	C	C	C
68	104	D	D	D
69	105	E	E	E
70	106	F	F	F
71	107	G	G	G
72	110	H	H	H
73	111	I	I	I
74	112	J	J	J
75	113	K	K	K
76	114	L	L	L
77	115	M	M	M
78	116	N	N	N
79	117	O	O	O
80	120	P	P	P
81	121	Q	Q	Q
82	122	R	R	R
83	123	S	S	S
84	124	T	T	T
85	125	U	U	U
86	126	V	V	V
87	127	W	W	W
88	130	X	X	X
89	131	Y	Y	Y
90	132	Z	Z	Z
91	133	bracketleft	bracketleft	bracketleft
92	134	backslash	backslash	backslash
93	135	bracketright	bracketright	bracketright
94	136	asciicircum	asciicircum	asciicircum
95	137	underscore	underscore	underscore

DECIMAL	OCTAL	PDFDOC	MACROMAN	WINANSII
96	140	grave	grave	grave
97	141	a	a	a
98	142	b	b	b
99	143	c	c	c
100	144	d	d	d
101	145	e	e	e
102	146	f	f	f
103	147	g	g	g
104	150	h	h	h
105	151	i	i	i
106	152	j	j	j
107	153	k	k	k
108	154	l	l	l
109	155	m	m	m
110	156	n	n	n
111	157	o	o	o
112	160	p	p	p
113	161	q	q	q
114	162	r	r	r
115	163	s	s	s
116	164	t	t	t
117	165	u	u	u
118	166	v	v	v
119	167	w	w	w
120	170	x	x	x
121	171	y	y	y
122	172	z	z	z
123	173	braceleft	braceleft	braceleft
124	174	bar	bar	bar
125	175	braceright	braceright	braceright
126	176	asciitilde	asciitilde	asciitilde
127	177	—	—	bullet
128	200	bullet	Adieresis	Euro
129	201	dagger	Aring	bullet
130	202	daggerdbl	Ccedilla	quotesinglbase
131	203	ellipsis	Eacute	florin
132	204	emdash	Ntilde	quotedblbase
133	205	endash	Odieresis	ellipsis
134	206	florin	Udieresis	dagger
135	207	fraction	aacute	daggerdbl
136	210	guilsinglleft	agrave	circumflex
137	211	guilsinglright	acircumflex	perthousand
138	212	minus	adieresis	Scaron
139	213	perthousand	atilde	guilsinglleft
140	214	quotedblbase	aring	OE
141	215	quotedblleft	ccedilla	bullet
142	216	quotedblright	eacute	Zcaron

DECIMAL	OCTAL	PDFDOC	MACROMAN	WINANSII
143	217	quoteleft	egrave	bullet
144	220	quoteright	ecircumflex	bullet
145	221	quotesinglbase	edieresis	quoteleft
146	222	trademark	iacute	quoteright
147	223	fi	igrave	quotedblleft
148	224	fl	icircumflex	quotedblright
149	225	Lslash	idieresis	bullet
150	226	OE	ntilde	endash
151	227	Scaron	oacute	emdash
152	230	Ydieresis	ograve	tilde
153	231	Zcaron	ocircumflex	trademark
154	232	dotlessi	odieresis	scaron
155	233	lslash	otilde	guilsinglright
156	234	oe	uacute	oe
157	235	scaron	ugrave	bullet
158	236	zcaron	ucircumflex	zcaron
159	237	—	udieresis	Ydieresis
160	240	Euro	dagger	space
161	241	exclamdown	degree	exclamdown
162	242	cent	cent	cent
163	243	sterling	sterling	sterling
164	244	currency	section	currency
165	245	yen	bullet	yen
166	246	brokenbar	paragraph	brokenbar
167	247	section	germandbls	section
168	250	dieresis	registered	dieresis
169	251	copyright	copyright	copyright
170	252	ordfeminine	trademark	ordfeminine
171	253	guillemotleft	acute	guillemotleft
172	254	logicalnot	dieresis	logicalnot
173	255	—	—	hyphen
174	256	registered	AE	registered
175	257	macron	Oslash	macron
176	260	degree		degree
177	261	plusminus	plusminus	plusminus
178	262	twosuperior	—	twosuperior
179	263	threesuperior	—	threesuperior
180	264	acute	yen	acute
181	265	mu	mu	mu
182	266	paragraph	—	paragraph
183	267	periodcentered	—	periodcentered
184	270	cedilla	—	cedilla
185	271	onesuperior	—	onesuperior
186	272	ordmasculine	—	ordmasculine
187	273	guillemotright	ordfeminine	guillemotright
188	274	onequarter	ordmasculine	onequarter
189	275	onehalf	—	onehalf

DECIMAL	OCTAL	PDFDOC	MACROMAN	WINANSII
190	276	threequarters	ae	threequarters
191	277	questiondown	oslash	questiondown
192	300	Agrave	questiondown	Agrave
193	301	Aacute	exclamdown	Aacute
194	302	Acircumflex	logicalnot	Acircumflex
195	303	Atilde	—	Atilde
196	304	Adieresis	florin	Adieresis
197	305	Aring	—	Aring
198	306	AE	—	AE
199	307	Ccedilla	guillemotleft	Ccedilla
200	310	Egrave	guillemotright	Egrave
201	311	Eacute	ellipsis	Eacute
202	312	Ecircumflex	space	Ecircumflex
203	313	Edieresis	Agrave	Edieresis
204	314	Igrave	Atilde	Igrave
205	315	Iacute	Otilde	Iacute
206	316	Icircumflex	OE	Icircumflex
207	317	Idieresis	oe	Idieresis
208	320	Eth	endash	Eth
209	321	Ntilde	emdash	Ntilde
210	322	Ograve	quotedblleft	Ograve
211	323	Oacute	quotedblright	Oacute
212	324	Ocircumflex	quoteleft	Ocircumflex
213	325	Otilde	quoteright	Otilde
214	326	Odieresis	divide	Odieresis
215	327	multiply	—	multiply
216	330	Oslash	ydieresis	Oslash
217	331	Ugrave	Ydieresis	Ugrave
218	332	Uacute	fraction	Uacute
219	333	Ucircumflex	currency	Ucircumflex
220	334	Udieresis	guilsinglleft	Udieresis
221	335	Yacute	guilsinglright	Yacute
222	336	Thorn	fi	Thorn
223	337	germandbls	fl	germandbls
224	340	agrave	daggerdbl	agrave
225	341	aacute	periodcentered	aacute
226	342	acircumflex	quotesinglbase	acircumflex
227	343	atilde	quotedblbase	atilde
228	344	adieresis	perthousand	adieresis
229	345	aring	Acircumflex	aring
230	346	ae	Ecircumflex	ae
231	347	ccedilla	Aacute	ccedilla
232	350	egrave	Edieresis	egrave
233	351	eacute	Egrave	eacute
234	352	ecircumflex	Iacute	ecircumflex
235	353	edieresis	Icircumflex	edieresis
236	354	igrave	Idieresis	igrave

DECIMAL	OCTAL	PDFDOC	MACROMAN	WINANSII
237	355	iacute	Igrave	iacute
238	356	icircumflex	Oacute	icircumflex
239	357	idieresis	Ocircumflex	idieresis
240	360	eth		eth
241	361	ntilde	Ograve	ntilde
242	362	ograve	Uacute	ograve
243	363	oacute	Ucircumflex	oacute
244	364	ocircumflex	Ugrave	ocircumflex
245	365	otilde	dotlessi	otilde
246	366	odieresis	circumflex	odieresis
247	367	divide	tilde	divide
248	370	oslash	macron	oslash
249	371	ugrave	breve	ugrave
250	372	uacute	dotaccent	uacute
251	373	ucircumflex	ring	ucircumflex
252	374	udieresis	cedilla	udieresis
253	375	yacute	hungarumlaut	yacute
254	376	thorn	ogonek	thorn
255	377	ydieresis	caron	ydieresis

Symbol Font Encoding

Although the font “Symbol” is a standard, built-in PDF font, it does not have the standard encoding used by most of the other fonts. Instead it has a unique encoding as does ZapfDingbats. However, unlike Zapf, Symbol font includes meaningful names for all the glyphs it contains.

Use the following table to construct new font maps for “Symbol” (besides the one provided with Argus). Alternatively, use the table to lookup glyph names directly when using the “Glyph Names” feature of the FONT MAP section.

DECIMAL	OCTAL	GLYPH	DECIMAL	OCTAL	GLYPH
032	040	space	161	241	Upsilon1
033	041	exclam	162	242	minute
034	042	universal	163	243	lessequal
035	043	numbersign	164	244	fraction
036	044	existential	165	245	infinity
037	045	percent	166	246	florin
038	046	ampersand	167	247	club
039	047	suchthat	168	250	diamond
040	050	parenleft	169	251	heart
041	051	parenright	170	252	spade
042	052	asteriskmath	171	253	arrowboth
043	053	plus	172	254	arrowleft
044	054	comma	173	255	arrowup

DECIMAL	OCTAL	GLYPH	DECIMAL	OCTAL	GLYPH
045	055	minus	174	256	arrowright
046	056	period	175	257	arrowdown
047	057	slash	176	260	degree
048	060	zero	177	261	plusminus
049	061	one	178	262	second
050	062	two	179	263	greaterequal
051	063	three	180	264	multiply
052	064	four	181	265	proportional
053	065	five	182	266	partialdiff
054	066	six	183	267	bullet
055	067	seven	184	270	divide
056	070	eight	185	271	notequal
057	071	nine	186	272	equivalence
058	072	colon	187	273	approxequal
059	073	semicolon	188	274	ellipsis
060	074	less	189	275	arrowvertex
061	075	equal	190	276	arrowhorizex
062	076	greater	191	277	carriagereturn
063	077	question	192	300	aleph
064	100	congruent	193	301	Ifraktur
065	101	Alpha	194	302	Rfraktur
066	102	Beta	195	303	weierstrass
067	103	Chi	196	304	circlemultiply
068	104	Delta	197	305	circleplus
069	105	Epsilon	198	306	emptyset
070	106	Phi	199	307	intersection
071	107	Gamma	200	310	union
072	110	Eta	201	311	propersuperset
073	111	Iota	202	312	reflexsuperset
074	112	theta1	203	313	notsubset
075	113	Kappa	204	314	propersubset
076	114	Lambda	205	315	reflexsubset
077	115	Mu	206	316	element
078	116	Nu	207	317	notelement
079	117	Omicron	208	320	angle
080	120	Pi	209	321	gradient
081	121	Theta	210	322	registerserif
082	122	Rho	211	323	copyrightserif
083	123	Sigma	212	324	trademarkserif
084	124	Tau	213	325	product
085	125	Upsilon	214	326	radical
086	126	sigma1	215	327	dotmath
087	127	Omega	216	330	logicalnot
088	130	Xi	217	331	logicaland
089	131	Psi	218	332	logicalor

DECIMAL	OCTAL	GLYPH	DECIMAL	OCTAL	GLYPH
090	132	Zeta	219	333	arrowdblboth
091	133	bracketleft	220	334	arrowdblleft
092	134	therefore	221	335	arrowdblup
093	135	bracketright	222	336	arrowdblright
094	136	perpendicular	223	337	arrowdbldown
095	137	underscore	224	340	lozenge
096	140	radicalex	225	341	angleleft
097	141	alpha	226	342	registersans
098	142	beta	227	343	copyrightsans
099	143	chi	228	344	trademarksans
100	144	delta	229	345	summation
101	145	epsilon	230	346	parenlefttp
102	146	phi	231	347	parenleftex
103	147	gamma	232	350	parenleftbt
104	150	eta	233	351	bracketlefttp
105	151	iota	234	352	bracketleftex
106	152	phi1	235	353	bracketleftbt
107	153	kappa	236	354	bracelefttp
108	154	lambda	237	355	braceleftmid
109	155	mu	238	356	braceleftbt
110	156	nu	239	357	braceex
111	157	omicron	241	361	angleright
112	160	pi	242	362	integral
113	161	theta	243	363	integraltp
114	162	rho	244	364	integralex
115	163	sigma	245	365	integralbt
116	164	tau	246	366	parenrighttp
117	165	upsilon	247	367	parenrightex
118	166	omega1	248	370	parenrightbt
119	167	omega	249	371	bracketrighttp
120	170	xi	250	372	bracketrightex
121	171	psi	251	373	bracketrightbt
122	172	zeta	252	374	bracerighttp
123	173	braceleft	253	375	bracerightmid
124	174	bar	254	376	bracerightbt
125	175	braceright			
126	176	similar			

Error Return Values

On exit Argus returns a status code describing the reason for exit. An exit code of zero means processing was successful. All other codes indicate an error.

Though there is no detailed explanation of all the possible error conditions, the following is a list of mnemonics used internaly 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		

Encoding Names

Below is a list of all encoding names recognised by the encoding translation module built-into Argus. Each numbered name may be followed by a number of synonyms. Either the name or its synonym may be used to specify the output encoding. See "[CHAR MAP]" on page 47 for more details.

000. UTF8
utf-8
ibm-1208
cp1208

001. UTF16_BigEndian
utf-16be
x-utf-16be

002. UTF16_LittleEndian
utf-16le
x-utf-16le

003. UTF16_PlatformEndian
ISO-10646-UCS-2
csUnicode
utf-16
ibm-17584
ibm-13488
ibm-1200
cp1200
ucs-2

004. UTF16_OppositeEndian

005. UTF32_BigEndian
utf-32be

006. UTF32_LittleEndian
utf-32le

007. UTF32_PlatformEndian
ISO-10646-UCS-4
csUCS4
utf-32
ucs-4

008. UTF32_OppositeEndian

009. UTF-7

010. SCSU

011. LATIN_1
iso-8859-1
ibm-819
cp819
latin1
8859-1
csisolatin1
iso-ir-100
cp367
ISO_8859-1:1987
l1
ANSI_X3.110-1983
819

012. US-ASCII
ascii
ascii-7
ANSI_X3.4-1968
ANSI_X3.4-1986
ISO_646.irv:1991
iso646-us
us
csASCII
646
iso-ir-6

013. ISO_2022
ISO-2022
2022
cp2022

014. ISO_2022,locale=ja,version=0
ISO-2022-JP
csISO2022JP

015. ISO_2022,locale=ja,version=1
ISO-2022-JP-1
JIS
JIS_Encoding

016. ISO_2022,locale=ja,version=2
ISO-2022-JP-2
csISO2022JP2

017. ISO_2022,locale=ja,version=3
JIS7

018. ISO_2022,locale=ja,version=4
JIS8

019. ISO_2022,locale=ko,version=0
ISO-2022-KR
csISO2022KR

020. ISO_2022,locale=ko,version=1
ibm-25546
ibm-25546_P100
25546

021. ISO_2022,locale=zh,version=0
ISO-2022-CN
csISO2022CN

022. ISO_2022,locale=zh,version=1
ISO-2022-CN-EXT

023. HZ
HZ-GB-2312

024. LMBCS-1
lmbcs

025. LMBCS-2

026. LMBCS-3

027. LMBCS-4

028. LMBCS-5

029. LMBCS-6

030. LMBCS-8

031. LMBCS-11

032. LMBCS-16

033. LMBCS-17

034. LMBCS-18

035. LMBCS-19

036. ibm-367

037. ebcdic-xml-us

038. ibm-912
iso-8859-2
cp912
latin2
8859-2
csisolatin2
iso-ir-101
ISO_8859-2:1987
l2
912

039. ibm-913
iso-8859-3
latin3
cp913
8859-3
csisolatin3
iso-ir-109
ISO_8859-3:1988
l3
913

040. ibm-914
iso-8859-4
latin4
cp914
8859-4
csisolatin4
iso-ir-110
ISO_8859-4:1988
l4
914

041. ibm-915
iso-8859-5
cyrillic
cp915
8859-5
csisolatincyrillic
iso-ir-144
ISO_8859-5:1988
915

042. ibm-1089
iso-8859-6
arabic
cp1089
8859-6
csisolatinarabic
iso-ir-127
ISO_8859-6:1987
ecma-114
asmo-708
1089

043. ibm-4909
cp813
iso-8859-7
greek
greek8
elot_928
ecma-118
8859-7
csisolatingreek
iso-ir-126
ISO_8859-7:1987
813

044. ibm-813

045. ibm-916
iso-8859-8
hebrew
cp916
8859-8
csisolatinhebrew
iso-ir-138
ISO_8859-8:1988
916

046. ibm-920
iso-8859-9
ECMA-128
latin5
cp920
8859-9
csisolatin5

iso-ir-148
ISO_8859-9:1989
l5
920

047. ibm-923
iso-8859-15
latin9
cp923
8859-15
latin0
csisolatin0
iso8859_15_fdis
csisolatin9
923

048. ibm-1252
ibm-1004
cp1004

049. ibm-943_P130-2000
ibm-943_VASCII_VSUB_VPUA
ibm-943

050. ibm-943_P14A-2000
ibm-943_VSUB_VPUA
Shift_JIS
csWindows31J
sjis
cp943
cp932
ms_kanji
csshiftjis
windows-31j
x-sjis
943

051. ibm-949_P110-2000
ibm-949_VASCII_VSUB_VPUA
ibm-949

052. ibm-949_P11A-2000
ibm-949_VSUB_VPUA
KS_C_5601-1987
iso-ir-149
KS_C_5601-1989
csKSC56011987
KSC_5601
johab
ks_x_1001:1992
949
ksc5601_1992

053. ibm-1370
Big5
csBig5
x-big5
cp950
950

054. ibm-950

055. ibm-1386
gbk
cp936
zh_cn

056. ibm-33722
EUC-JP
ibm-eucJP
eucjis
cseucpkdfmtjapanese
X-EUC-JP
cp33722
33722

057. ibm-970
EUC-KR
ibm-eucKR
csEUCKR

058. ibm-964
EUC-TW
ibm-eucTW
cns11643

059. ibm-1383
EUC-CN
ibm-eucCN
GB_2312-80
chinese
gb
iso-ir-58
csISO58GB231280
GB2312
gb2312-1980
cp1383
1383

060. ibm-1162
tis-620
cp874
windows-874
ms874
cp9066
874

061. ibm-874
ibm-1161

062. ibm-437
cp437
csPC8CodePage437
437

063. ibm-850
IBM850
cp850
850
csPC850Multilingual

064. ibm-851

IBM851
cp851
851
csPC851

065. ibm-858
cp858
IBM00858

066. ibm-9044
852
csPCp852
cp852

067. ibm-852
IBM852

068. ibm-872
855
csIBM855
cp855
csPCp855

069. ibm-855
IBM855

070. ibm-856
cp856
856

071. ibm-9049
857
csIBM857
cp857

072. ibm-857
IBM857

073. ibm-859
cp859

074. ibm-860
IBM860
cp860
860
csIBM860

075. ibm-861
IBM861
cp861
861
cp-is
csIBM861

076. ibm-867
cp867
862
cp862
cspc862latinhebrew

077. ibm-862
IBM862

078. ibm-863
IBM863
cp863
863
csIBM863

079. ibm-17248
cp864
csIBM864

080. ibm-864
IBM864

081. ibm-865
IBM865
cp865
865
csIBM865

082. ibm-808
cp866
866
csIBM866

083. ibm-866

084. ibm-868
IBM868
cp868
cp-ar
csIBM868
868

085. ibm-9061
cp869
869
cp-gr
csIBM869

086. ibm-869
IBM869

087. ibm-878
KOI8-R
cp878
koi8
cskoi8r

088. ibm-901
cp921
921

089. ibm-921

090. ibm-902
cp922
922

091. ibm-922

092. ibm-942_P120-2000
ibm-942_VASCII_VSUB_VPUA
ibm-942
ibm-932
ibm-932_VASCII_VSUB_VPUA

093. ibm-942_P12A-2000
ibm-942_VSUB_VPUA
shift_jis78
sjis78
pck
ibm-932_VSUB_VPUA

094. ibm-5346
windows-1250
cp1250

095. ibm-5347
windows-1251
cp1251

096. ibm-5348
windows-1252
cp1252

097. ibm-5349
windows-1253
cp1253

098. ibm-5350
windows-1254
cp1254

099. ibm-5351
windows-1255
cp1255

100. ibm-5352
windows-1256
cp1256

101. ibm-5353
windows-1257
cp1257

102. ibm-5354
windows-1258
cp1258

103. ibm-1250

104. ibm-1251

105. ibm-1253

106. ibm-1254

107. ibm-1255

108. ibm-1256

109. ibm-1257

110. ibm-1258

111. ibm-1275
macintosh
mac
csMacintosh

112. ibm-1276
Adobe-Standard-Encoding
csAdobeStandardEncoding

113. ibm-1277
Adobe-Latin1-Encoding

114. ibm-1280
macgr

115. ibm-1281
mactr

116. ibm-1282
macce

117. ibm-1283
maccy

118. ibm-1051
hp-roman8
roman8
r8
csHPRoman8

119. ibm-1388

120. ibm-849
cp1131

121. ibm-848
cp1125

122. ibm-5104
cp1008

123. ibm-9238
cp1046

124. ibm-1363_P110-2000
ibm-1363
ibm-1363_VASCII_VSUB_VPUA
ibm-1362

125. ibm-1363_P11B-2000
ibm-1363_VSUB_VPUA
windows-949
cp949
cp1363
ksc
korean

126. ibm-5210
cp1114

127. ibm-21427
cp947

128. ibm-37
IBM037
ibm-037
cpibm37
ebcdic-cp-us
ebcdic-cp-ca
ebcdic-cp-wt
ebcdic-cp-nl
csIBM037
cp37
cp037
037

129. ibm-273
IBM273
csIBM273
ebcdic-de
cp273
cpibm273
273

130. ibm-277
IBM277
EBCDIC-CP-DK
EBCDIC-CP-NO
csIBM277
ebcdic-dk
cp277
cpibm277
277

131. ibm-278
IBM278
ebcdic-cp-fi
ebcdic-cp-se
csIBM278
ebcdic-sv
cp278
cpibm278
278

132. ibm-280
IBM280
ebcdic-cp-it
csIBM280
cp280
cpibm280
280

133. ibm-284
IBM284
ebcdic-cp-es
csIBM284
cp284
cpibm284
284

134. ibm-285
IBM285
ebcdic-cp-gb
csIBM285
ebcdic-gb
cp285
cpibm285
285

135. ibm-290
IBM290
EBCDIC-JP-kana
csIBM290
cp290

136. ibm-297
IBM297
ebcdic-cp-fr
csIBM297
cp297
cpibm297
297

137. ibm-420
IBM420
ebcdic-cp-ar1
csIBM420
cp420
420

138. ibm-424
IBM424
ebcdic-cp-he
csIBM424
cp424
424

139. ibm-500
IBM500
cpibm500
csIBM500
cp500
ebcdic-cp-be
ebcdic-cp-ch
500

140. ibm-803
cp803

141. ibm-834
cp834

142. ibm-835
cp835

143. ibm-871
IBM871
ebcdic-cp-is

csIBM871
cpibm871
cp871
871

144. ibm-930
cp930
cpibm930
930

145. ibm-933
cp933
cpibm933
933

146. ibm-935
cp935
cpibm935
935

147. ibm-937
cp937
cpibm937
937

148. ibm-939
cp939
939

149. ibm-1006_P100-2000
ibm-1006
ibm-1006_VPUA

150. ibm-1006_X100-2000
ibm-1006_STD

151. ibm-1025_P100-2000
ibm-1025
ibm-1025_STD

152. ibm-1026_P100-2000
ibm-1026
CP1026
IBM1026
csIBM1026
ibm-1026_STD

153. ibm-1097_P100-2000
ibm-1097
ibm-1097_VPUA

154. ibm-1097_X100-2000
ibm-1097_STD

155. ibm-1098_P100-2000
ibm-1098
ibm-1098_VSUB_VPUA

156. ibm-1098_X100-2000
ibm-1098_VSUB

157. ibm-1112_P100-2000
ibm-1112
cp1112
1112
ibm-1112_STD

158. ibm-1122_P100-2000
ibm-1122
cp1122
ibm-1122
1122
ibm-1122_STD

159. ibm-1124_P100-2000
ibm-1124
ibm-1124_STD

160. ibm-1125_P100-2000
ibm-1125
ibm-1125_VSUB

161. ibm-1129_P100-2000
ibm-1129
ibm-1129_STD

162. ibm-1130_P100-2000
ibm-1130
ibm-1130_STD

163. ibm-1131_P100-2000
ibm-1131
ibm-1131_VSUB

164. ibm-1132_P100-2000
ibm-1132
ibm-1132_STD

165. ibm-1133_P100-2000
ibm-1133
ibm-1133_STD

166. ibm-1137_P100-2000
ibm-1137
ibm-1137_STD

167. ibm-1381_P110-2000
ibm-1381
ibm-1381_VSUB_VPUA

168. ibm-806_P100-2000
ibm-806
ibm-806_VSUB

169. ibm-870_P100-2000
ibm-870
CP870
IBM870
ibm-870_STD
ebcdic-cp-roece
ebcdic-cp-yu
csIBM870

170. ibm-875_P100-2000
ibm-875
cp875
ibm-875
875
ibm-875_STD

171. ibm-9030_P100-2000
ibm-9030
ibm-9030_STD

172. ibm-9066_P100-2000
ibm-9066
ibm-9066_VSUB

173. ibm-918_P100-2000
ibm-918
CP918
IBM918
ibm-918_VPUA
ebcdic-cp-ar2
csIBM918

174. ibm-918_X100-2000
ibm-918_STD

175. ibm-1390
cpibm1390

176. ibm-1371
cpibm1371

177. ibm-1047
cpibm1047

178. ibm-1123
cpibm1123

179. ibm-1140
cpibm1140
IBM01140

180. ibm-1141
cpibm1141
IBM01141

181. ibm-1142
cpibm1142
IBM01142

182. ibm-1143
cpibm1143
IBM01143

183. ibm-1144
cpibm1144

184. ibm-1145
cpibm1145

185. ibm-1146
cpibm1146

186. ibm-1147
cpibm1147

187. ibm-1148
cpibm1148

188. ibm-1149
cpibm1149
ebcdic-is

189. ibm-1153
cpibm1153

190. ibm-1154
cp1025
cpibm1154

191. ibm-1155
cpibm1155

192. ibm-1156
cpibm1156

193. ibm-1157
cpibm1157

194. ibm-1158
cp1123
cpibm1158
1123

195. ibm-1159
cp28709

196. ibm-1160
cp9030
cpibm1160

197. ibm-1164
cp1130
cpibm1164

198. ibm-1399

199. ibm-1364_P110-2000
ibm-1364_VPUA
ibm-1364
cp1364

200. ibm-8482

201. ibm-4899
cpibm4899

202. ibm-4971
cpibm4971

203. ibm-9027

204. ibm-5123
cp1027

205. ibm-12712
cpibm12712
ebcdic-he

206. ibm-16684
cp300

207. ibm-16804
cpibm16804
ebcdic-ar

208. ibm-37-s390
ibm037-s390

209. ibm-1047-s390

210. ibm-1140-s390

211. ibm-1142-s390

212. ibm-1143-s390

213. ibm-1144-s390

214. ibm-1145-s390

215. ibm-1146-s390

216. ibm-1147-s390

217. ibm-1148-s390

218. ibm-1149-s390

219. ibm-1153-s390

220. ibm-12712-s390

221. ibm-16804-s390

222. gb18030

Notes