

PRIMEOCR™

Access Guide **Version 5.10**

High Accuracy OCR Engine

PRIME RECOGNITION

PHONE: (425) 895-0550

EMAIL: support@primerecognition.com

WEB: www.primerecognition.com

FTP: [ftp.primerecognition.com](ftp://ftp.primerecognition.com)

Copyright © 1994-2010 Prime Recognition. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photo-copying, recording, or otherwise without the prior written permission of the publishers.

Printed in the United States of America.

Statement of Limited Warranty

Prime Recognition warrants to the original licensee of this program that it conforms to Prime Recognition's specifications. Should this program, in Prime Recognition's option, malfunction due to non-conformity with Prime Recognition's specifications, Prime Recognition will, at its option, repair, replace, or update the program at no charge, provided that the program has not been subjected to misuse, abuse, accident, disaster or non-Prime Recognition authorized alterations, modification, and/or repairs. In no event shall Prime Recognition be liable for incidental or consequential damages in connection with or arising out of the furnishing, performance, or use of any of these programs. Prime Recognition reserves the right to modify or revise all or part of this document without notice and shall not be responsible for any loss, cost, or damage, including consequential damage caused by reliance on these materials.

US Government Restricted Rights

The documentation and software are provided with restricted rights. Use, duplication, or disclosure by the government is subject to restrictions as set forth in subparagraph (C)(1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 52.227-7013.

Trademarks

PrimeOCR is a trademark of Prime Recognition.

Other trademarks appearing in this manual are trademarks of their respective companies.

Table of Contents

Preface

Overview

Product Architecture	1-1
Throughput.....	1-2
Inputs to PrimeOCR.....	1-2
File based Images	1-2
RAM based Images (Developers Only).....	1-3
Image Resolution.....	1-3
Image Size	1-3
Image Quality	1-4
Image Source.....	1-4
Font Types.....	1-4
Font sizes.....	1-4
Printer Type.....	1-5
Text Orientation	1-5
Language	1-5
Zones	1-6
Automatic Zoning	1-6
Zone limits.....	1-8
Engine Training	1-8
Engine Tuning	1-9
Outputs From PrimeOCR.....	1-9
File Based Output Formats.....	1-9
RAM Based Output Formats (Developers Only)	1-10
Developer Features	1-10
Memory Management	1-10
Progress Feedback (Callback)	1-10
Ease of Programming	1-10

System Requirements

Personal Computer.....	2-1
Software	2-2
OS Privileges	2-2

Installation

Software Installation	3-1
Hardware Key vs. Software Key	3-2
Hardware Key Installation	3-2
LPT1 based Key (no longer available to new customers, see USB key)	3-2
USB based Key	3-3
Software Key Installation	3-3
Software Key Hardware/OS Dependencies.....	3-4
Testing Installation Process	3-4
PRIMEOCR.INI Configuration	3-5
PRIMAGE.INI Configuration.....	3-10
Testing the Development Environment (Developers Only).....	3-18
PATH	3-19
Distributing the PrimeOCR Engine (Developers Only).....	3-19

PrimeOCR Job Server

Overview.....	4-1
Simple OCR Job Server Tutorial	4-2
Starting Job Server	4-3
Stopping Job Server	4-3
Error Reporting	4-3
Job Status	4-4
Configuration	4-4
Creating Jobs & Templates	4-4
Running as a Service.....	4-5
Job File Format	4-5
Template File Format.....	4-6
Image Zone Numbering.....	4-13
Customization Features.....	4-14

String Options

Overview.....	5-1
String Option: Change Characters in Output	5-2
String Option: Change Strings in Output.....	5-4
String Option: Various Statistical Measures	5-7
String Option: Variable Processing of Engines based on Image Quality.....	5-9
String Option: Change LOG File Settings	5-11
String Option: Phone Books	5-12
String Option: ASCII Defaults.....	5-13
String Option: HTML Defaults.....	5-14

String Option: RTF Defaults.....	5-15
String Option: PDF Defaults.....	5-17
String Option: PDF Retain.....	5-22
String Option: PDF Security.....	5-24
String Option: PDA Defaults.....	5-27
String Option: Word Data in PRO Output.....	5-28
String Option: Point Size Change.....	5-30
String Option: Output Point Size Change.....	5-31
String Option: UTF Output.....	5-32
String Option: File Output (second or third file output).....	5-33
String Option: Subscript/Superscript Recognition Enhancement.....	5-35
String Option: Subscript/Superscript Tagging.....	5-36
String Option: Strong Auto Rotate Accuracy Improvement.....	5-37
String Option: Bates Number Accuracy Improvement.....	5-38

PrimeOCR Output Formats

File Based Output Formats.....	6-1
PRO Output.....	6-4
Definitions.....	6-4
Coordinate Units.....	6-4
Line Baseline Coordinates.....	6-4
Recognized Characters.....	6-4
Confidence Levels.....	6-4
Character Format Data.....	6-5
Character Size.....	6-5
Character Bounding Box.....	6-5
PRO Output Hierarchy/Content.....	6-6
Document Level.....	6-6
Page Level.....	6-6
Zone Level.....	6-6
Line Level.....	6-6
Character Level.....	6-6
File Based Output.....	6-7
Example.....	6-8
OMR Zones.....	6-10
Image Zones.....	6-11
RAM based PRO Output (Developers Only).....	6-11
zonestructure Data Structure.....	6-12
linestruct Data Structure.....	6-12
bbox Data Structure.....	6-13
Example.....	6-14
Asian Language Output in PRO file.....	6-15

Other RAM based Output (Developers Only)	6-15
--	------

Troubleshooting

Troubleshooting Suggestions.....	7-1
PrimeOCR Job Server Error Reporting	7-2
Intermediate Error Reporting	7-2
Activity Log	7-3
Other Debug Facilities	7-3
Error Numbers/Messages.....	7-3
Trouble Shooting (Developers Only).....	7-3
API Error Reporting (Developers Only).....	7-4

Contacting Prime Recognition & Warranty/Support

Warranty/Support.....	8-1
-----------------------	-----

InputAccel Module

Other Documentation.....	9-1
System Requirements.....	9-1
Installation	9-1
Configuration	9-3
Local Setup.....	9-3
Starting Engine	9-4
Stopping Engine.....	9-4
Error Reporting	9-4
Features Not Supported.....	9-4

Programming to PrimeOCR API

Include Files	10-1
Libraries	10-1
Programming Languages.....	10-1
API Call Functionality Groups	10-2
API Call Scope.....	10-3
API Call Sequence	10-4
API Call Functionality	10-4
API Call Functionality	10-5
Configuration Memory.....	10-5
Error reporting.....	10-5
Yielding.....	10-5
Call Return Timing.....	10-5
Progress Reporting (Callback function)	10-5

API Calls.....	10-6
pr_open().....	10-8
pr_open().....	10-8
pr_configure_ini()	10-11
pr_report_error()	10-12
pr_close()	10-13
pr_read_image_RAM()	10-14
pr_read_image_file().....	10-15
pr_image_information()	10-17
pr_get_image_information().....	10-18
pr_blank_image_area().....	10-20
pr_deskew_image()	10-22
pr_preprocess_image()	10-24
pr_auto_rotate().....	10-25
pr_save_image().....	10-27
pr_save_image_mpage().....	10-28
NOTES ON IMAGES.....	10-29
FAX resolution files (204 X 196 or 204 X 98).....	10-29
Color/grayscale images	10-29
pr_set_new_document()	10-31
pr_set_language().....	10-32
pr_set_low_quality()	10-34
pr_set_printer_type().....	10-35
pr_autozone()	10-36
pr_get_autozone().....	10-39
pr_set_number_of_zones().....	10-40
pr_configure_zone().....	10-41
pr_configure_OMR_zone().....	10-45
pr_configure_image_zone()	10-47
pr_load_page_template().....	10-49
pr_recognize_image().....	10-50
pr_language_id()	10-51
pr_set_user_dictionary().....	10-53
pr_configure_lexical().....	10-54
pr_lexical_plus().....	10-55
NOTES ON OUTPUT	10-57
pr_output_OMR_percent().....	10-58
pr_output_zone_lines().....	10-59
pr_output_line_length().....	10-60
pr_output_line_data()	10-61
pr_output_char_box().....	10-63
pr_configure_output()	10-64
pr_convert_output_to_file()	10-65

pr_convert_output_to_file()	10-65
pr_del_results()	10-68
pr_string_options()	10-69
pr_pre_process_PDF().....	10-70

Example Applications

C Source Code Sample Application.....	11-1
Functionality	11-1
Files.....	11-2
Output.....	11-3
Example Source Code	11-3
Visual Basic Source Code Sample.....	11-4
Overview	11-4
Special Considerations of Interfacing to PrimeOCR via Visual Basic.....	11-4
Other Source Code Samples	11-5

OCR Job Server as a Service

Installation of Service	12-1
Removal of Service.....	12-1
Configuration of OCR parameters	12-1
Configuration of Service.....	12-1
Stop/Start.....	12-1
Auto Start	12-2
Auto Restart	12-2
Error Conditions	12-2

Appendix A - Recognized Characters

Index

Preface

Scope of Access Guide

The PrimeOCR Access Guide provides information on the installation of the PrimeOCR engine, and its capabilities, and the installation and use of the PrimeOCR Job Server. It is intended for users of the PrimeOCR Job Server and for developers who will write programs that interface to, and control, the PrimeOCR engine.

In preparing this Guide, we assume that you have some experience with:

- basic Windows user skills
- basic OCR terminology

Developers should have some experience with:

- basic Windows programming
- interfacing with a Windows style DLL

The PrimeOCR engine offers unparalleled levels of OCR accuracy at cost levels an order of magnitude below other OCR accuracy improvement technologies. However, this document does not discuss performance characteristics in detail, please refer to the PrimeOCR datasheet for information in this area.

Prime Recognition also offers other products, such as PrimeZone, PrimeView, PrimeVerify and PrimePost. These other products are Windows based tools which “front end” and “back end” the PrimeOCR engine. They are an optional compliment to PrimeOCR's functionality. PrimeOCR does not require these other products to perform the functionality described in this guide.

Contents of Access Guide

- **Chapter 1, Overview**
This chapter gives a brief overview of the PrimeOCR engine, describing its form, its features, and the facilities available to the user/developer.
- **Chapter 2, System Requirements**
This chapter provides detailed descriptions of the hardware and software required to support the PrimeOCR engine.
- **Chapter 3, Installation**
This chapter provides instructions and procedures for installing PrimeOCR including the Prime Recognition hardware key.

- **Chapter 4, PrimeOCR Job Server**
This chapter describes the functionality, configuration and operation issues of a prewritten application which allows end users to use the PrimeOCR engine without programming.
- **Chapter 5, String Options**
This chapter describes “string options”, special commands that can be used by the PrimeOCR Job Server, or by a developer to invoke new functionality in the PrimeOCR engine.
- **Chapter 6, PrimeOCR Output Formats**
This chapter provides descriptions of the output generated by PrimeOCR.
- **Chapter 7, Troubleshooting**
This chapter provides suggestions that might be helpful in troubleshooting PrimeOCR, and directions to a listing of all error messages.
- **Chapter 8, Contacting Prime Recognition**
This chapter describes how to contact Prime Recognition for sales or technical support.
- **Chapter 9, InputAccel Module**
Installation, configuration and other details of running PrimeOCR as a InputAccel Module.
- **Chapter 10, Programming to the PrimeOCR API Module**
Describes programming issues and all available API calls for PrimeOCR API.
- **Chapter 11, Example Applications**
Provides a brief overview of the C and VB source code samples provided with the Access Kit.
- **Appendix A**
A listing of all characters that can be recognized by PrimeOCR.
- **Index**
This is a subject index of the contents of this Guide.

Other Publications


Other publications that may be of interest include:

- *PrimeOCR data sheet*
- *PrimeView/PrimeVerify User's Guide*

- *PrimeView data sheet*
- *PrimeVerify data sheet*
- *PrimePost User's Guide*
- *PrimeZone User's Guide*
- *Prime Recognition High Accuracy OCR Cost Justification (White Paper)*
- *Prime Recognition High Accuracy OCR "Cleaner Data" Justification (White Paper)*

Documentation Conventions

The following conventions are used in this *Access Guide*:

Convention	Usage
Bold	Bold text indicates emphasis
<i>Italic</i>	Italics indicate variable information in command lines and in dialog box entries. Variable information is the appropriate data that you enter <i>in place</i> of the italic entry in the example or instructions. Italics in text also indicate titles of manuals, chapters, or headings.
Courier	Courier is used when presenting messages from the program to the user.
[]	Square brackets indicate optional entries. If an optional entry is desired, enter the information between the brackets. Brackets are also used to denote keystrokes in text, e.g., [F1].
	This icon points out important information.

Chapter 1

Overview

Prime Recognition's High Accuracy OCR Engine, "PrimeOCR", is a software program that runs on one Windows based personal computer. It can recognize text, "marks", or save images from bit map images created from scanning, FAX, or other electronic means. PrimeOCR reduces the number of character recognition, confidence level, attributes, and location errors that occur in conventional OCR technology.



Note

Prime Recognition licenses its products as a base configuration with many options. This allows users to pay for only the features they need. The functionality described in this manual includes all options.

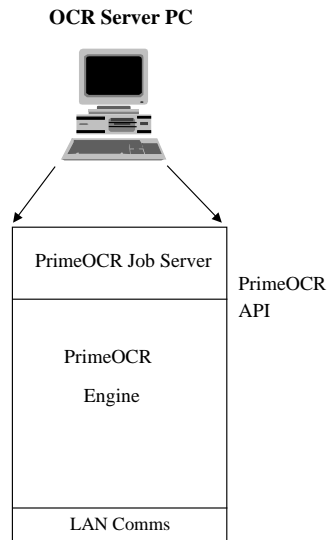
Product Architecture

PrimeOCR is designed as a DLL (Dynamic Link Library) that implements the PrimeOCR API. The PrimeOCR DLL will load and unload a series of DLLs, as required, that implement Prime Recognition's technology. All required technology is contained within the licensed PrimeOCR product and is managed by the PrimeOCR engine. No other products must be licensed, integrated, or managed by the developer and his/her code to enable OCR.

Users - However, the PrimeOCR engine is not a stand-alone application. A program must be written that interfaces to, and controls, PrimeOCR through API calls, which are defined in later in this guide. Prime Recognition has written several sophisticated applications, the PrimeOCR Job Server, PrimeView, and PrimeVerify, which allow you to use the PrimeOCR engine with no development. If you want to begin using the PrimeOCR engine right away, or you do not want to ever program your own PrimeOCR interface program then you should evaluate these programs (The Job Server is described in *Chapter 4*).

Developers - Two simple example programs, written by Prime Recognition, are included in the Toolkit in source code form and are described in *Chapter 11*.

If PrimeOCR is used as an OCR server in a LAN environment (a typical production imaging scenario), then the developer's application is responsible for all external communications to the imaging system, including all LAN traffic, except that PrimeOCR will read image files from a file server if the image path includes a DOS file mapping to the file server.



**Note**

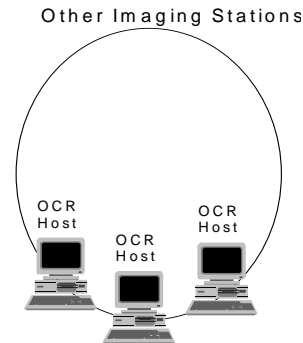
Other applications can be run on the same PC as PrimeOCR but, because OCR is very CPU and memory intensive, OCR throughput will be reduced. We highly recommend that the PC be dedicated to the PrimeOCR engine and supporting programs. Only one instance of the engine can be running at any time on one PC.

Throughput

Single PC Throughput. For high speed place PrimeOCR on the fastest available Windows based PC, dedicated to OCR, with at the minimum amount of free RAM.

PrimeOCR can use multiple CPUs in a multiple CPU system.

System Throughput. Very high throughput can be achieved by implementing multiple PrimeOCR engines in parallel as logically separate units (see figure).

**Inputs to PrimeOCR**

Images are accepted as image files or as raw bitmaps in RAM through the API.

File based Images

Image files must be supplied with a filename and path, long file names and UNC paths are supported. The PrimeOCR engine will automatically find and read the image file header, automatically identify the file type (e.g., TIFF vs. PCX), and read the image into RAM for processing.

Supported file types are:

- TIFF Version 5.0 and above
 - Uncompressed, Modified G3 (FAX), G3, G4, LZW, Multipage, Binary, Greyscale, Color
- PCX
- JPEG (Color, Greyscale) (including JPEG2000)
- PDF
 - Black & White, Color, Greyscale.
 - PDF Image only, PDF Image + Text.

PDF Normal is not officially supported since many PDF Normal files are non standard but the majority of files can be read. It would also be very unusual for PDF Normal files to be put through OCR since the text is already

electronic and 100% accurate. Some types of PDF files can not be read including: protected with security/encryption, non standard, damaged PDF files.

PDF attributes such as bookmarks, author, subject, page mode, and hyperlinks can be saved from input PDF via PDF_RETAIN string option.

RAM based Images (Developers Only)

RAM based images are passed to the engine via a pointer to the beginning of the image in RAM. The image must be in a raw binary (color/greyscale not supported in memory) bitmap format (not a Windows bitmap). A raw bitmap in RAM can be created by taking every row of pixels in an image and placing the rows sequentially into memory. For example, an image that looks like this on paper,

```

Row 0
Row 1
Row 2
Row 3

```

will be converted to this in memory:

```

Row 0Row 1Row 2Row 3

```

No special delimiters are placed in the bitmap. The pointer points to the beginning of the image (the "R" in Row 0).

Image Resolution

Image resolutions of 150, 200, 240, 300, 400, and 600 dpi (dots per inch) are supported. Both height and width resolution must be the same. (PrimeOCR can handle small variations in dpi from these standards.)



Note

PrimeOCR's accuracy is substantially better at 300 dpi vs. 200 dpi, so whenever possible generate images at 300 dpi. 400 dpi does generate better results than 300 dpi if a majority of the text in the image is smaller than 8 point (see *Font Sizes*). Other image types may or may not benefit from 400 dpi resolution. We recommend that you experiment with 400 dpi vs. 300 dpi on your target images. 400 dpi recognition is slower and 400 dpi images take substantially more RAM and disk space than 300 dpi images.

FAX Resolutions

Fine resolution FAX images (204 X 196 dpi) can be processed. Standard resolution FAX images (204 X 98) can also be processed. PrimeOCR automatically doubles the height to 204 X 196. FAX images are treated as 200 X 200 images, since PrimeOCR reports coordinates in a measure that is not resolution dependent this does not cause any location errors.

Image Size

The overall size of the image is limited by PrimeOCR to 26 inches in height and width (in other words, 5200 pixels at 200 dpi, 15600 pixels at 600 dpi), however, character recognition accuracy falls very rapidly for zones over 5600 pixels in size (height or width). If the full page is treated as one zone the practical limit for the size of the image is 5600 pixels in height and/or width. (For asian language images the limit is 22.5 inches in height and width, and 12,500 pixels).

Image Quality

PrimeOCR will process all types of image quality, from excellent to poor. PrimeOCR can handle significant amounts of page skew, speckling, broken characters, and other image problems. Poor quality images cause a significant decrease in throughput, and a significant increase in the level of errors generated, for conventional OCR as well as PrimeOCR. However, PrimeOCR's average error generation will always be significantly lower than conventional OCR. One benefit of PrimeOCR is that images that were too bad for conventional OCR for cost effective processing, such as microfiche, can now be cost effectively processed.



Note – Image Preprocessing

Prime Recognition highly recommends that images be "cleaned up". For example, deskew the image (but only if the image has more than 1% skew). PrimeOCR can perform deskew, despeckle, auto rotation and other image clean up operations within its engine.



Note – Color

Color images get converted to bitonal (black & white) in the engine before processing. If color shading exists behind the text we recommend you use the despeckle image preprocessing option to remove this shading in the black and white image before OCR.

Image Source

Images can be created by FAX, microfiche, microfilm, scanner input or any other type of device that outputs images electronically. Similarly the type of source document can be a magazine, insurance claim form, newspaper, technical report, business memo, or any other source. The only restriction is that the resulting electronic image passed to PrimeOCR must fit the criteria described elsewhere in this *Guide*.

Font Types

A wide variety of fonts can be recognized accurately. PrimeOCR includes "omnifont" technology that can recognize new fonts it has never seen before. Heavily stylized, or script type fonts are examples of fonts that are difficult to recognize accurately.

Fixed and variable pitch fonts (sometimes called monospaced and variably spaced fonts) can be recognized. This attribute is automatically sensed and reported by the engine.

Font sizes

Font or type sizes within the following ranges will be accurately recognized:

200 dpi	12-40 points
300 dpi	8-35 points
400 dpi	6-30 points
600 dpi	3-20 points

Sizes outside these ranges may be recognized but accuracy will fall off rapidly. This attribute is automatically sensed by the engine.

Printer Type

PrimeOCR will recognize characters formed by the following printer types:

- Machine print (includes typewriter, inkjet, laser printer, typeset, and any other printing technology that can form regular, fully formed characters).
- Dot matrix
- Bar Code (1D and selected 2D)
- MICR (custom version of PrimeOCR)
- OCRA/OCRB (custom version of PrimeOCR)

This attribute must be specified by the user via the PrimeView application (See *PrimeView User's Guide*), or the PrimeOCR Job Server template file (See *Chapter 4 – PrimeOCR Job Server*), or by the developer through the PrimeOCR API.

Text Orientation

Text must be in an upright orientation to be recognized correctly. The engine will not process text that is in a landscape orientation ("landscape" in this context only refers to the orientation of the text, not whether the page is wider than tall). PrimeOCR offers two "auto-rotation" features to automatically rotate the page until the text is orientated correctly.

Language

PrimeOCR will recognize text in the following languages:

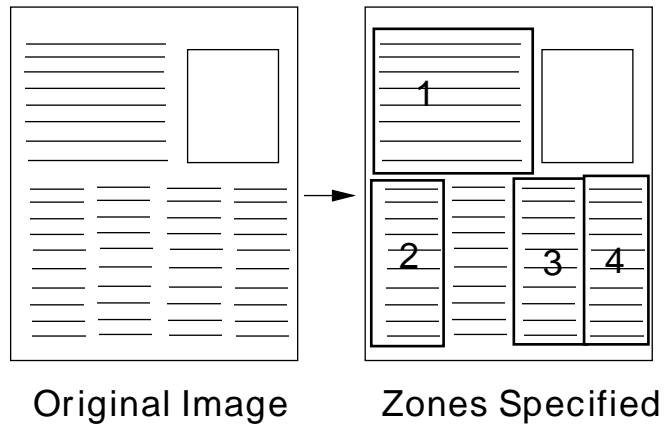
- Danish
- Dutch
- French
- German
- Italian
- Norwegian
- Portuguese
- Spanish
- Swedish
- U.K. English
- U.S. English
- Chinese (simple)
- Chinese (traditional)
- Japanese
- Korean

-Russian

PrimeOCR will recognize one dominant language per page. A secondary language, usually English, can also be recognized. Languages do not have to be identified in advance (although processing is significantly faster if you can specify the language), PrimeOCR has functionality that can identify the language for you.

Zones

Zones are rectangles of area on the original image which, ideally, do not overlap. These zones must be defined by the user through the PrimeView application (See *PrimeView User's Guide*) or the PrimeOCR Job Server template file (See *Chapter 4 – PrimeOCR Job Server*) or by the developer through the PrimeOCR API. Alternatively PrimeOCR can try to sense these zones through its automatic zoning feature.



At least one zone must be defined for each image (e.g., the whole page in a full text application). Within each zone text will be recognized as a single column of text. Therefore, if your image has multiple columns of text (e.g., a newspaper) then zones should be defined for each column of text (you will probably want to order the zones by the reading order of the text).

Zones may overlap, however, text that is in two or more zones will usually generate a higher error rate. Therefore, for example, if a skewed image is forcing zones to overlap, we recommend the following:

- solve the problem by deskewing the image before zoning and OCR
- if deskew is not practical, try to define zone overlap so that it only includes white space (no text).

Automatic Zoning

PrimeOCR can segment the image into zones automatically. This feature has two modes of operation. The first is segmentation only. In this mode the image is segmented into text (and optionally, image) zones and the location of these zones is reported back to the application for its review in order from top to bottom. The second mode zones the image for text (and optionally, image) zones and automatically orders the OCR output into the expected reading order, without any review, so that, for example, text in columns flows in the natural reading order.

Automatic zoning, in particular reading order, is not always accurate, however, PrimeOCR uses technology that has been noted by industry tests as one of the leading zoning technologies.



Notes

Please discuss your autozoning needs with Prime Recognition. We can custom configure our autozoning technology to fit your application and get substantially better autozoning results.

You can force autozoning to ignore the edges of a page when doing autozoning. See the file ...\BIN\“autozone_margins.ini.example” for more information.

Each zone will have the following unique properties:

- **Zone number**

- **Content restrictions**

- No restrictions. Characters recognized are listed in Appendix A.
- Alpha. Characters recognized are a-z, A-Z, comma, period and international alpha characters which are given strong preference (however, other characters may appear if they are extremely confident characters). See Appendix A for a full list.
- Numeric. Characters recognized are 0-9 and other characters typically found in numeric applications are given strong preference (however, other characters may appear if they are extremely confident characters). See Appendix A for a full list.
- Alpha and No restrictions can be further restricted to UPPERCASE and lowercase.
- OMR. Optical Marks, for example, "check boxes". PrimeOCR reports back percent of zone that is black (filled in.)
- Image. These zones are saved as independent image files.
- Barcode. Selected 1D and 2D barcode types (see prbarcode.ini for details).

- **Lexical check**

- Lexical check - Standard. A series of lexical checks are performed on the output. In "Alpha" and "Numeric" zones the checks will convert any characters outside the definition of the zone to a "?" with confidence equal to 1.
- Lexical check - Plus. A strong lexical check is performed on "No restrictions" zones (currently only works with English language).
- No Lexical check.

- **Accuracy level**

- Or in other words number of engines voting. More engines equal greater accuracy but slower speed. On average you will get the following results:

# Engines	Error Reduction	Speed
1	0%	1.0
2	15%	2.1
3	64%	3.8
4	74%	5.8
5	76%	6.1
6	80%	8.0

The **zone number** has no meaning to the PrimeOCR engine except that PrimeOCR's output is ordered by the zone number. This means that the developer can order the zones in an arbitrary order as required by his application.

Content restriction is useful, particularly in forms applications, to reduce the scope of the recognition task. This allows for faster speeds and higher accuracy.

Accuracy level configuration allows the developer to trade off throughput with accuracy. For example, very high accuracy may not be required on one or two fields on a form (e.g., a comments field), this field is configured as one zone and recognized with one engine accuracy (conventional OCR speed and accuracy), while the other fields are configured as zones and recognized with three engine accuracy, or the best accuracy available.

Lexical check slows down recognition speed considerably but has a very significant positive effect on accuracy of text if the text is lexical in nature. Data that is not lexically sensitive would include, for example, proper names, or alphanumeric part numbers.

Lexical check Plus

Compares low confidence words against a lexicon database. Can provide three basic functions:

- (1) Increases the confidence of words that it finds within its database (if, for example, you are manually verifying OCR output this option can significantly reduce the number of words to be verified). This function is relatively reliable and safe.
- (2) Can "fix" word and improve confidence. This is a very powerful function but be careful to configure this function carefully, settings that are too aggressive relative to your typical word patterns can cause changes that are not correct.
- (3) If can not "fix" word then it lowers confidence of characters further.

Configuration of this feature for the PrimeOCR Job Server is handled through `..\BIN\PRLEXICA.INI` file. Specific configuration documentation is also in this file.

User Dictionaries

The user may create his own user dictionary. This dictionary is used by Lexical check and Lexical Check Plus to improve character recognition accuracy and confidence reporting. The user dictionary file is merely a ASCII text file, with one word per line (a word must be 2-32 characters in length). If using the PrimeOCR Job Server then enter the path to the user dictionary in `..\BIN\PRLEXICA.INI` file. (The use of a user dictionary does not require the use of Lexical Check Plus).

Zone limits

2400 zones per image* (automatic zoning feature is limited to 240 zones per image)

500 lines per zone*

500 characters per line*

*These items can be increased by Prime Recognition on a custom basis. Please call for details.

Engine Training

The PrimeOCR engine offers several levels of training:

- The user/developer may notify the engine that an image is very similar to the last image processed (e.g., the next page in a multi page document). The PrimeOCR engine will attempt to use its experience with the last document to improve its accuracy and speed on this image.
- One of the internal engines supported by PrimeOCR can be "trained" on specific fonts and characters. This can have a small impact on accuracy and speed, however its utility

is limited to applications in which only one font used across a lot of pages. Please contact Prime Recognition for further information.

- Prime Recognition also offers its PRTRAIN engine (as an option). This engine is very font specific but it can recognize any arbitrary shape and is less susceptible to noise than conventional engines. The PRTRAIN engine must be trained by Prime Recognition and its application is best suited for custom projects.
- For large opportunities Prime Recognition can modify the engine for the specific application. The value of this effort depends greatly on the nature of the images being processed. Please contact Prime Recognition for further information.

Engine Tuning

The user/developer may also tune the PrimeOCR engine, on a zone by zone basis, by deciding which engines to run and how much weight to give to each engine. However, Prime Recognition has predefined accuracy settings of LEVEL_1, LEVEL_2, etc. These settings run engine 1 with full weight, engines 1 and 2 with full weight, etc. For most applications these settings are the best choice. However, if a zone is always of a specific type, e.g., always unconnected dot matrix, then it may be worth experimenting with engine weights for that zone.

Outputs From PrimeOCR

File Based Output Formats

PrimeOCR can generate output in the following formats:

- Prime Recognition format (PRO)
- ASCII text
- Formatted ASCII
- RRI3
- PDA
- ZyIndex
- RTF
- PDF – Image Only
- PDF – Normal
- PDF – Image plus hidden text (including PDF/A format)
- Comma delimited
- HTML
- XML
- Custom formats

For more details on each output format please read *Chapter 6 - Output Formats*.

The PRO format includes the following information:

- Page data: width, height, OCR configuration settings, average confidence
- Zone data: location and OCR configuration settings. If an OMR zone then the percentage of zone that is black. If an image zone then a path to the captured image zone as a TIF file.
- Line data: baseline based bounding box

-Character data: recognized characters, confidence level, bounding box, character attributes, point size.

Some of this data is also included in other output formats.

RAM Based Output Formats (Developers Only)

The OCR output is available as a pointer to a RAM based PrimeOCR data structure. The content of this data structure is highly analogous to the file based PRO output. This approach is best suited for sophisticated C/C++ based developers.

Alternatively the output can be retrieved “one feature at a time” through simple API calls. This alternative is best suited for Visual Basic developers or less sophisticated C/C++ developers.

Developer Features

Memory Management

Once the PrimeOCR output is no longer needed, one simple call will free up all RAM consumed by the PrimeOCR output.

Progress Feedback (Callback)

The PrimeOCR engine can generate messages periodically that indicate the ongoing progress of its processing. The developer must write a callback function (an example is provided in the sample application) and register the function through the PrimeOCR API. The PrimeOCR engine will call this function during processing and send it a message which indicates the current state of processing.

Ease of Programming

The PrimeOCR API and output format were designed to make it easy for developers to understand and program to the PrimeOCR engine. For example:

- The interface to the PrimeOCR engine is a relatively small number of simple, orthogonal, C style function calls. Other OCR engines utilize a much more complex event driven architecture or support a huge variety of function calls, at multiple architectural levels, with overlap in functionality between many calls.
- PrimeOCR output does not embed control, escape, and delimiter sequences into the recognized text output to identify attributes of the text. Instead each line of text is output as plain text with separate arrays of attributes for character attributes, such as confidence and bounding box coordinates. The output data is intuitively tied to each other via their position in the array and does not have to go through a sophisticated state machine parser for human or programmatic access. PrimeOCR output can be accessed randomly whereas other OCR engine output must be parsed in a sequential manner, then interpreted.
- PrimeOCR maintains configuration information in memory between images so variables that do not change between images, or change infrequently, such as resolution, can be set initially and then used for all following documents.

Chapter 2

System Requirements

The PrimeOCR Engine runs on Intel or compatible personal computers with a modern Windows operating system.

For PrimeOCR to work properly and efficiently, it needs the following hardware and software:

Personal Computer

- Intel PC or 100% compatible computer. See notes at end of this chapter on CPUS, and CPU utilization.

We strongly recommend dedicating the PC to OCR in high volume usage.

- A hard disk with 150 megabytes of available disk space for software installation.
- If a hardware key license is used then a USB port is required.
- RAM (free after loading all other software)
 - "Workstation" operating systems:
 - Minimum: 128 MB
 - Recommended: 256 MB
 - " Server" operating systems:
 - Minimum: 256 MB
 - Recommended: 512 MB

Items that require more memory:

- 400 dpi images: +16 MB
- 600 dpi images: +48 MB
- Color/grey scale: +32-128 MB
- PDF Image input (b/w 8.5x11) 32MB
- PDF Image input (color 8.5x11) 96MB
- Large dimension images (depends on size)

Software

Windows XP

Windows 2003 Server

Windows Vista

Windows 2008 Server

Windows 7

x64 versions are supported but may require manual installation configuration (see *ChangeNTErrrorReporting=1* in Chapter 3). Performance is 10% slower in x64 implementations.

Running under VMWare, Hyper-V, and Virtual PC is supported. Other virtual environments have unique licensing issues, please contact Prime Recognition.

OS Privileges

Administrator rights are required to install the software.

Read/write rights are required for the ...*(install dir)*\BIN directory.

Administrator rights are required if you are using a software license with a page limit (including the automatic evaluation license).

Administrator rights are generally required unless you make the changes to the Registry described in Chapter 3.



“Run as admin” (Vista, Windows 2008, Win7)

Depending on your security settings, it may be important for you to run the application that uses the PrimeOCR engine (for example, the OCR Job Server, or your programmed application) with Admin rights, even if you are not logged in with Admin rights. To do this, right click on `\bin\PRLICENSE.EXE` (for example) and select to run as Administrator. If you are not logged in as admin, you will be asked to do so (for this application only).



“Number of CPUs”

Up to four CPUs may be utilized depending on the number of CPUs licensed from Prime and by the number of “Levels” licensed from Prime (The evaluation license includes a license for four CPUs but the base configuration license for PrimeOCR, the least expensive license, only includes one CPU). (A “CPU” is Windows terminology, many users would call this a “core” in current CPU terminology). Hyperthread CPUs are automatically sensed and added to a Prime license at no extra cost. Hyperthread CPUs can result in up to a 15% speed improvement on PCs with a few number of CPUs (but less on PCs with a larger number of CPUs/cores).



CPU utilization

Although we may use up to four CPUs (cores), the average utilization of all CPUs can be considerably lower than 100%. The average utilization will depend on a number of factors, including accuracy levels configured, number of licensed and available CPUs, and how many “serial” processes are utilized (file reading, writing, image enhancement steps, autozoning, etc.)

To maximize the utilization of CPUs on a PC we recommend you license as many CPUs as your PC contains up to the number of accuracy levels, for example, if you select Level 3 accuracy and you are on a four CPU PC, we recommend you license three CPUs (we will also be utilizing a fourth CPU in this configuration at no extra cost to you). This will maximize the utilization of the four CPUs on your PC.

However, considerable CPU capacity will still exist on the PC. To further utilize the CPU capacity you can: (a) use VMWare or Hyper-V to create multiple OS instances on a single PC (each of which will require a separate PrimeOCR license), or (b) create multiple user accounts on the PC and run a copy of PrimeOCR in each user account. The advantage of (b) is that only one license for PrimeOCR is required. However, a disadvantage is that some operating system versions allow for communications between user accounts, which can cause PrimeOCR operation problems. (If you do decide to attempt the multiple user account approach, create copies of the \bin directory and run each account from a different \bin directory. If using the Job Server, delete the ocrjob.nam file in each account. It will get regenerated automatically with a unique ID.)

Chapter 3

Installation

The installation of the PrimeOCR engine is composed of two simple steps:

- Installation of the software
- Installation of license

Software Installation

Run PrimeOCR_5XXY.EXE that you downloaded (or is on the CD supplied to you).

The installation program will handle most installation issues automatically, including decompressing all files into a new or existing directory selected by the installer.

The directory will end up with the following subdirectories:

BIN	Executables required by PrimeOCR engine plus files for PrimeOCR Job Server and source code sample C program.
DOC	Manual, in PDF format (requires Acrobat [Reader])
JOB	Job file directory for the PrimeOCR Job Server application (used for sample jobs, user can use any directory)
Images	Image file directory for the PrimeOCR Job Server application (used for sample images, user can use any directory)

If you choose to install the Developer Software you will also get the following subdirectories:

LIB	Libraries required by PrimeOCR engine
SAMPLES	
-C	Files for PrimeOCR C example application
-VB	Files for PrimeOCR VB example application
-C++NET	Files for PrimeOCR C++ NET based example application
-VB NET	Files for PrimeOCR VB NET based example application
INCLUDE	*.h files for developer application development

Hardware Key vs. Software Key

PrimeOCR can utilize a hardware key or software key license.

The advantage of a hardware key is that it can be easily moved to a different machine, and with proper changes to the registry described in Chapter 3, allows PrimeOCR to run even if user does not have administrator rights. Its disadvantages include the need for a working USB (or LPT1 in some cases) port, and keeping track of the hardware key itself.

A software license cannot be moved to another machine, it is tied to the specific machine and that machine's OS/hardware configuration (see details below). It also requires that the user have administrator rights (if a page limit license is present). However, if the user does not have a working USB (or in some cases LPT1) port, or does not want to work with hardware keys, then this is another choice.

Hardware Key Installation

LPT1 based Key (no longer available to new customers, see USB key)

If the user requests a LPT1 based hardware key, it must be installed on the LPT1 printer port for the PrimeOCR engine to initialize and run. Please use the supplied thumbscrews to securely attach the key to the port. You may attach a printer cable to the hardware key as all normal LPT1 traffic is passed through, however, Prime Recognition does not recommend that you use the OCR engine PC for any other processes including printing.

If you have multiple hardware keys from different vendors you may have to experiment with the stacking order of the keys. A key usually performs best if it is the first key, i.e., closest to the PC.

The engine will periodically read the hardware key so the key must be present during all engine processing.

If you are receiving hardware key errors:

- Make sure the key is securely attached to the LPT1 port.
- Verify that LPT1 port is turned on in BIOS.
- Make sure that LPT1 traffic is not captured to a network printer.
- Remove the printer cable and any other hardware keys, if they are attached to the key.
- Note that only the hardware key supplied with the product will pass the engine's tests.

A hardware key software driver must be loaded into the operating system to allow PrimeOCR to communicate with the key. The installation program will automatically install the proper hardware key software driver, but the user must be sure to reboot after such installation.

USB based Key

After installing the software, install the key into an active USB port on the PC. You may get a message from the operating system that it needs to load a driver for the USB device. If so choose the option that allows you to load the driver of your choice, change the directory to the BIN\SENT\Sentinel System Driver directory and choose the .INF file that is in that directory. You should reboot at this point.

The engine will periodically read the hardware key so the key must be present during all engine processing.

If you are receiving hardware key errors:

- Make sure the key is securely attached to the USB port.
- Verify that USB port is active.
- Note that only the hardware key supplied with the product will pass the engine's tests.

Software Key Installation

If a software key has never been installed on the target machine, and the user tries to run PrimeOCR (or any other Prime Recognition product), a temporary software license will automatically be created.

A permanent software license can be created (or a temporary software license extended) by running ... \BIN\PRLICENSE.EXE. This will create the file ... \BIN\PRLICENSE.LOG. Send this log file to your sales contact by email (or call him/her by phone and read the relevant data).



Warning

The file name of the PRLICENSE.LOG file can be changed. If you are updating multiple software licenses at one time, make sure to name the log file something meaningful that will allow you to associate that file name to the specific PC it came from.

Your sales contact will return instructions by which to update the license via PRLICENSE.EXE.

In some cases you may receive a program. The name may be SoftwareKeyWrite.EXE or a name that corresponds to the name of the log file you sent (particularly if you sent in multiple log files). Place this program in your ... \BIN directory and run. You should get several dialog boxes indicating the progress and successful conclusion of the software license creation.



Warning (for Software License Only and for Vista, Windows 7, and Windows 2008)

If you plan on touching PrimeOCR from multiple user accounts (for example, install under Administrator, but run under User#1 account, then the software license needs to be setup in a specific way. While logged into the Administrator account, right click on `\bin\PRLICENSE.EXE` and select to run as Administrator. If you need to run this program again, remember to right click and run as Administrator. If you don't do this, then Vista (and later OSs built from this code base) creates a specific license for the Administrator account only (as opposed to a license that can be used by all user accounts including Administrator).

Software Key Hardware/OS Dependencies

The software license is specific to a particular machine. It cannot be moved. The license is also tied to three key items: the operating system installation, the hard drive and drive designator in which PrimeOCR is installed, and a combination of the CPU type and LAN adapter ID. This means that you can change anything else on the machine without affecting your license.



Warning (for Software License Only)

You can change either your CPU type or LAN adapter (but not both). You can not change or reinstall your OS installation (Service Packs and updates are OK), and you can not change the hard drive on which PrimeOCR was installed or its drive letter (e.g. "h:\").

Testing Installation Process

After the software and the hardware key have been installed, try running the application Prime OCR Job Server. If any errors occur then something is probably wrong with:

- The license. See *Hardware/Software Key Installation* sections.
- Your environment. Read *Chapter 2 – System Requirements* for the minimum system requirements of the PrimeOCR engine, and remove all other loaded programs.

PRIMEOCR.INI Configuration

A file called PRIMEOCR.INI is located in the \BIN directory. This file controls the configuration of selected items of the PrimeOCR engine. Most users will not need to view/modify the PRIMEOCR.INI file.

A copy of PRIMEOCR.INI file is not required for distribution. Items will default to those settings shown below if PRIMEOCR.INI is not present. Developers may also control these settings through the `pr_configure_ini()` call described in *Chapter 10 – Programming to the PrimeOCR API*.

To change the setting of an item merely edit the number after the "=".

Version=5.00

Indicates version of PRIMEOCR.INI file. Should match version of PROCR32.DLL (engine).

ShowProgressWindow=3

0=NO, 1=Independent Window, 2=Window within App window[old default],
3=Window within App window [default]

"0"- disable the display of the progress update window during PrimeOCR operation. This saves about 2% of CPU time.

"1"- places the progress window in a stand-alone window which can be moved by the user/developer.

"2"- places the progress window in a fixed position 330 units "Left" and 0 units "Top" relative to the parent application's window. (See description for "3" below).

"3"- places the progress window in a fixed position 394 units "Left" and 0 units "Top" relative to the parent application's window. If the parent window is not at least 394 units wide no part of the progress window will be visible. There is no way to change the position of the window except by Prime Recognition customization. However, you can install a callback function and receive the exact same messages, which you then can display in any way you choose.

ShowErrorsDialogBox=0

0=NO, 1=YES

Change this setting to 1 to enable more testing routines. If an error is found in these routines a dialog box will be displayed. (An error dialog box is helpful during testing but when the engine is put into production you don't want to require that someone be available to push the "OK" button of the dialog box before your application can proceed to handle the error automatically.)

ImageDebug=0

0=NO, 1=YES - PR use only

PrintIntermediateResults=0

0=NO, 1=YES - PR use only

PrintIntermediateResults2=0

0=NO, 1=YES - PR use only

CompareDebug=0

0=NO, 1=YES - PR use only

RunOneEngine=0

0=NO, 1-X=YES - PR use only

Debug1=0

0=NO, 1=YES - PR use only

LogFile=0

0=NO, 1=YES

Change this setting to 1 to enable the generation of a text file that logs all major PrimeOCR engine activity, including calls made to PrimeOCR engine. This file is often helpful to Prime Recognition for remote technical support. The LOG file's default path and filename is "\BIN\PROCR.LOG". The default filename, path and size may be modified by OCR Job Server Setup or a "string option" (see *Chapter 5*) may be used.

ConfidenceBbox=9

0-9, 0 means none - 9 means all

Change this setting to the confidence level (proper range is between 0-9) to control which characters report a non-zero bounding box in the PrimeOCR output. For example, if your error correction application does not look at characters with confidence levels above 6, then you will save a small amount of CPU time, and file based output will be smaller, if you change this setting to 6.

WhiteSpaceIs0InImage=1

0=NO, 1=YES [Most common]

In most image types "0" means white (and black is "1"). If instead white is "1" in your image types then change this item to 0.

MostSignificantBit=0

0=PC, 2=UNIX

"0" is the correct setting for PC generated files. If your images came from a UNIX system you may need to change this setting to "2".

Engine1OnPC=1**Engine2OnPC=1****Engine3OnPC=1****Engine4OnPC=1****Engine5OnPC=1****Engine6OnPC=1**

0=NO, 1=YES - PR use only

ChangeNTErrrorReporting=1

0=NO, 1=YES

"1" is very strongly recommended. If set to "0" a fatal error in one of the conventional engines will cause a dialog box to pop up detailing the error. Only if someone manually clears the dialog box will the memory of the failed engine be cleared. If not cleared 5 to 10 engine failures will consume all of the memory of the PC and cause the PrimeOCR engine to fail. A setting of "1" causes the error to be reported to the Event Viewer and the memory of failed engine is reclaimed automatically and immediately. Many internal engine failures can occur without causing problems for PrimeOCR. **You must be logged in with Administrator privileges for this feature to work! (or see notes below)**

If the setting value is "1" then the PrimeOCR engine will read the "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Windows\ErrorMode" setting in the Registry. If it is not "2" it will save the original setting and modify the registry setting to "2". On exit, the PrimeOCR engine will reset the registry entry to its original value.

This setting will also read the value for your default debugger program (often set to Dr. Watson) in "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\AeDebug\Debugger" and reset it to blank for the duration of PrimeOCR processing. On exit PrimeOCR will reset this value.

If the OS is XP/Windows2003 Server then this setting will also read the values in "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\PCHealth\ErrorReporting\DoReport" and "...ShowUI" and reset them to 0 for the duration of PrimeOCR processing. On exit PrimeOCR will reset these values.

If the OS is Vista then this setting will also read the values in "HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\ Windows Error Reporting\Disabled" and "DontShowUI" and reset them to 1 for the duration of PrimeOCR processing. On exit PrimeOCR will reset these values.

If the OS is Windows 2008 Server then this setting will also read the values in "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\ Windows Error Reporting\Disabled" and "DontShowUI" and reset it to 1 for the duration of PrimeOCR processing. On exit PrimeOCR will reset this value.

If the OS is Win7, Vista, Windows 2008 Server, XP SP2, Windows 2003 Server SP1 or later then this setting will also create/edit values in "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\AppCompatFlags\Layers". A value name of "C:\PRDEV\BIN\ICRSRV32.EXE" (except with actual install path) with Data Type REG_SZ, and value of DisableNXShowUI is created. A second value is created except with an EXE name of ... \XOCR32B.EXE. A third value is also created with the name of \PROCR8.EXE. If you need to make these last changes manually (see below), then we recommend you use an application in the Control Panel to make the same changes. The sequence to get to the appropriate screen is:

Control Panel\System\Advanced\Performance\Data Execution Prevention

If “Turn on DEP for essential Windows programs and services only” is selected no additional changes need to be made. However, if “Turn on DEP for all programs and services except for those I select:” is selected, then use the Add... button to select ICRSRV32.EXE, PROCR8.EXE, and XOCR32B.EXE (they will be in the install path\BIN, for example, C:\PRDEV\BIN).



Warning

Depending on your security settings, x64 versions of Vista, Win7, and Windows 200X Server may require that changes noted above for DEP be performed manually by user.

IF YOU CAN NOT GIVE THE USER ADMINISTRATOR PRIVILEGES

Login as the administrator and make the changes to the Registry manually that have been described in this section. When the user, who does not have administrative rights, logs in and uses PrimeOCR, the proper Registry settings will already be in place. Warnings may appear in the LOG file that PrimeOCR was unable to change the Registry, but these warnings can be ignored since the Registry changes were already made elsewhere.



Warning

This means that if the original setting was "0" or "1", programs that happen to run during the same time as PrimeOCR will now fail in a slightly different way than before. Instead of an error message being sent to the Event Viewer and an error dialog box being generated, now, only the error message is sent to Event Viewer. A symptom of this behavior is an application that just "disappears" without the usual dialog box that indicates the cause of the fatal error.



Warning

If the engine is abnormally terminated during processing, for example, via GPF, or using the task manager to terminate PrimeOCR, then the registry settings are not reset.

PrimeOCR must have read/write permissions to the \BIN directory. If your user does not have admin rights then you may need to manually give the user read/write permissions to this directory.

TimeOut=2

of seconds to add to internal formula which determines when an engine has ceased to operate correctly. The internal formula takes into account image size and complexity. Most users should leave at 2. If your images are particularly poor quality, your machine is slow, and/or you are running with a high number of CPUs (e.g. 4), you may experiment with a larger number, if you see a lot of “Engine bypassed” messages in the log.

PRIMAGE.INI Configuration

The PrimeOCR engine can perform many different image-preprocessing steps. A file called PRIMAGE.INI is located in the /BIN directory. This file controls the configuration of the image conversion/enhancement options of PrimeOCR.

A copy of PRIMAGE.INI file is not required for distribution. Items will default to the settings present in the PRIMAGE.INI file noted below.

To change the setting of an item merely edit the number after the "=".

[Color/Gray Image Binarization]

The following settings apply only to images that are sensed to be color, grey scale, or a non-standard format. These settings will be applied to the image that is converted to black and white for internal OCR purposes (any part of the image that will be output from OCR is covered in the next section).

Brightness=80

0-255 (If blank or "0" then value of 128 is used). This setting is used to threshold image in its conversion to a binary image for OCR purposes. 80 is a good value for OCR documents. (100-128 recommended if you use FixedThreshold=0 below).

FixedThreshold=1

0-1. If set to 0 then image will be converted to black and white using dynamic thresholding (more sophisticated and often more accurate way of distinguishing text from background). If set to 1 (default as it gives more control to user), then a fixed threshold will be used to distinguish text from background (a combination of Brightness and Contrast settings). This setting is only relevant in PrimeOCR, it is not present or implemented in other products).

DesampleDPI=0

0-600, dpi to convert image to. If blank or "0" then leave dpi of image equal to original.

Contrast=230

0-255 (If blank or "0" then value of 128 is used). This setting is used to threshold image in its conversion to a binary image for OCR purposes. 230 is a good value for OCR documents. ONLY USED IF FixedThreshold=1.

[Color/Gray Image Output]

The following settings apply only to images that are sensed to be color, grey scale, or a non-standard format. These settings will be applied to the image that is placed in the output file (e.g., RTF, PDF, etc.), or the .FIX output file, for example.

Binarize=0

0-1. If blank or "0" then output image (including any .FIX generated by the Job Server) will stay in its original color/grayscale format. If set to 1, then the output image will be in a black and white format (the same format used by the engine internally for OCR).



Warning

If left at 0 some image enhancement changes will not flow through to the output color image. For example:

Deskew, registration, and sub image actions will not be reflected in the original color image, leading to possible coordinate errors in PDF, RTF, HTML, or PRO output files.

Autorotation (strong and weak) will be reflected in the output color image (and coordinates in output files will be correct).

Despeckle, line removal, etc. do not change coordinates but these changes will not be reflected in the output image.

DesampleDPI=0

0-600, dpi of images that will be saved. If blank or "0" then leave dpi of image equal to original. This setting is used to control dpi of any sub images created by OCR engine, for example, image zones. Desampling the image can result in a much smaller output file (at the expense of image quality). (Only relevant if Binarize=0)

Compression=0

-100 to 100 (only valid for JPEG compression). If blank or "0" then average compression is used on image.

-100=Low compression. 100=High compression. Higher compression value decreases quality of image. This setting is used to control quality of any sub images created by OCR engine, for example, image zones. (Only relevant if Binarize=0)

Contrast=230

0-255 (If blank or "0" then value of 128 is used). Generally leave this setting at 0 unless some specific changes need to be made to image to make it more aesthetically pleasing to a viewer. (Only relevant if Binarize=0)

Brightness=80

0-255 (If blank or "0" then value of 128 is used). Generally leave this setting at 0 unless some specific changes need to be made to image to make it more aesthetically pleasing to a viewer. (Only relevant if Binarize=0)

[PDF2TIFF]

The following settings apply only to images that are input to the engine as PDF files.

Version=1

0,1. "1" is the default. "1" is faster in all black and white image scenarios. If the "1" version fails for any reason (for example, color or grayscale input image) then "0" version is automatically used.

ReadDPI=300

150, 200, 240, 300, 400, 600. The PDF file will be read in and converted to an image with this resolution. (No method exists to determine the original or native “resolution” of the PDF file, so this value must be set by the user).

AllowColor=1

0,1. “1” is the default. If you have no color or grayscale images then “0” is a faster setting.

AllowMultipage=1

0,1. “1” is the default. If you have no multiple page images then “0” is a faster setting.

Check4Text=0

0,1. “0” is the default. If set to “1”, and you are using the Job Server, and your input images are PDF, and we sense that the incoming PDF file already has text output in it, then this file is copied to the output directory without any further processing.

Check4Normal=0

0,1. “0” is the default. If set to “1”, and you are using the Job Server, and your input images are PDF, and we sense that the incoming PDF file has no images in it, then this file is copied to the output directory without any further processing. (Some files that are considered PDF Normal may have images embedded, so note that these files will not qualify for this treatment).

Pdf2Text=0

0,1. “0” is the default. If set to “1”, and you are using the Job Server or PrimeZone, then text will be extracted from incoming PDF files and output in a text file (with output path, file name of incoming PDF file, and .txt extension). This setting and (Check4Text or Check4Normal) are mutually exclusive. If both are set to 1 then Pdf2Text is automatically set to 0.

In many cases the following preprocessing steps can significantly increase OCR accuracy. This portion of the file controls the configuration of the image enhancement options of PrimeOCR. The settings for these features are exactly those of the ScanFix (TM) image-processing engine, which is embedded within the PrimeOCR engine.

The PRIMAGE.INI file controls the configuration of each image preprocessing step **BUT IT DOES NOT TURN ON THE STEP**. To turn on the step:

- (A) If you are using the OCRJOB Server, you must set the proper parameters in the template file (using Wizard, PrimeView, or editing the template file directly).
- (B) If you are programming to the API, you must make a call to `pr_preprocess_image()`.

---PR_DESKEW

Automatically straightens skewed (crooked) images on the page. Seeks out either text or horizontal lines to establish the amount of skew and then corrects by rotating image.

mindetectlength=600

Minimum length in pixels of a run of text or a line used to detect skew. Longer is typically better.

maxacceptableskew=250

A value of 150 means 150 horizontal pixels for each vertical pixel of skew. 100 corresponds to 1% skew, 200 to 0.5% skew. Set higher for more precision. NOTE: Set to no more than 50% of the minimum detect length parameter. 150-250 works well with OCR.

deskewprotect=1

“0” turns off. If “1”, this feature will attempt to adjust rotational distortion into white space between letters. Deskew takes longer with this option but we recommend you use this setting for increased OCR accuracy.

---PR_HORIZONTAL_REGISTER

Aligns the image with a consistent left margin. Accurate to within 3 pixels on average (1/100"). Zoning and forms OCR require registration because page scan locations may vary up to 1/2".

resultantleftmargin=10

The amount of left margin, in pixels, remaining in document after registration. 150 pixels corresponds to 1/2 inch of margin at 300 dpi.

horzcentralfocus=0

“0” turns off. If “1”, do central focus analysis. This will ignore letterheads, footnotes, etc. in margin analysis.

horzaddonly=0

“0” turns off. If “1”, restrict margin adjustment to add margin only. This will assure that no portion of your image will ever be inadvertently lost.

horzignoreholes=0

“0” turns off. If “1”, ignore any punch holes on the left edge of the document.

horzlineregister=0

Set to the minimum length of a vertical line which is expected near the left of the image. Set to 0 for register detection when there is no line at the left of the image. This option is particularly useful with forms.

---PR_VERTICAL_REGISTER

Aligns the image with a consistent top margin. Accurate to within 3 pixels on average (1/100"). Zoning and forms OCR require registration because page scan locations may vary up to 1/2".

resultantuppermargin=10

The amount of upper margin, in pixels, remaining in document after registration. 300 pixels corresponds to 1 inch of margin at 300 dpi.

vertcentralfocus=0

“0” turns off. If “1”, do central focus analysis. This will ignore left or right justified logos or page numbers, etc. in margin analysis.

vertaddonly=0

“0” turns off. If “1”, restrict margin adjustment to add margin only. This will insure that no portion of your image will ever be inadvertently lost.

vertlineregister=0

Set to the minimum length of a horizontal line which is expected near the top of the image. Set to 0 for register detection when there is no line at the top of the image. This option is particularly useful with forms.

---PR_HORIZONTAL_LINE_REMOVAL

Removes horizontal lines from an image and can report their location. Greatly reduces OCR errors and false translations. Repairs intersected characters.

horzminlinelength=300

Minimum length in pixels of a line to locate and (optionally) remove. Set small enough to detect the shortest lines in the image, but not so short as to remove large text as well. See Maximum Line Thickness parameter.

horzmaxlinewidth=10

Maximum thickness of a line to be detected. Set smaller (thinner) to differentiate between large text and thinner lines of the same length.

horzmaxlinegap=0

Maximum allowed gap in any line in pixels. Insures complete removal of degraded lines. Set higher to remove lines in poor-quality images.

horzcleanwidth=2

Amount of cleaning of stray pixels to be done near a removed line. Set higher if line remnants remain in poor-quality images. Reduce if characters near lines are affected.

horzreconstructwidth=20

The maximum pixel width of characters to be repaired after line removal. Set higher if text is not being fully repaired. Set lower if you find erroneous reconstruction. Set to 0 to disable character repair. In that case, set Edge cleaning factor lower than the default.

---PR_VERTICAL_LINE_REMOVAL

Removes vertical lines from an image. Greatly reduces OCR errors and false translations such as I's, L's, and 1's. Repairs intersected characters.

vertminlinelength=300

Minimum length in pixels of a line to locate and optionally remove. Set small enough to detect the shortest lines in the image, but not so short as to remove large text as well. See Maximum Line Thickness parameter.

vertmaxlinewidth=10

Maximum thickness of a line to be detected. Set smaller (thinner) to differentiate between large text and thinner lines of the same length.

vertmaxlinegap=0

Maximum allowed gap in any line in pixels. Assures complete removal of degraded lines. Set higher to remove lines in poor-quality images.

vertcleanwidth=2

Amount of cleaning of stray pixels to be done near a removed line. Set higher if line remnants remain in poor-quality images. Reduce if characters near lines are affected.

vertreconstructwidth=20

The maximum pixel width of characters to be repaired after line removal. Set higher if text is not being fully repaired. Set lower if you find erroneous reconstruction. Set to 0 to disable character repair. In that case, set Edge Cleaning factor lower than the default.

---PR_INVERSE_TEXT

Inverse Text Correction automatically detects areas of inverse (white-on-black) text and converts it into normal (black-on-white) text. It will handle multiple zones of inverse text on the page and inverse zones of various shapes, such as rectangles, circles, ovals, etc.

invheight=50

Smallest vertical thickness of any inverse text area. Set higher to reduce false detects, lower to catch thinner zones.

invwidth=300

Smallest horizontal width of any inverse text area. Set higher to reduce false detect, lower to catch thinner zones.

invedge=10

The smallest horizontal distance between the edges of an inverse text zone and the inverse text itself. Decrease if inverse zones are missed. Increase if normal text is being treated as inverse.

---PR_DESPECK

The settings before “despeckhorizontal” control a special version of despeckle, i.e., Dot Shading Removal, which removes dot shading from text. THIS IS THE SAME DOT SHADING THAT IS IN THE PRIMEOCR JOB SERVER WIZARD.

despeckheight=300

Smallest expected vertical thickness of any dot shaded area. Set higher to reduce false detects. Set lower to catch thinner zones of shading.

despeckwidth=50

Smallest expected horizontal width of any dot shaded area. Set higher to reduce false detects. Set lower to catch thinner zones of shading.

maxspecksize=5

Maximum thickness in pixels of any dots in a shaded area. Increase to find larger dots. Decrease to prevent removal of small text.

horzadjust=0

Tweaks the horizontal component of the automated dot sizing function. Set higher if some dots survive dot shading removal. For example, a value of 2 will remove dots 2 pixels wider than usual. Set to -1 or -2 if text is being degraded.

vertadjust=0

Tweaks the vertical component of the automated dot sizing function. Set higher if some dots survive dot shading removal. For example, a value of 1 will remove dots 1 pixel larger than usual. Set to -1 or -2 if text is being degraded.

protect=3

“0” turns off. If “1”, protects characters from removal damage in horizontal direction, “2” protects in the vertical direction, and “3” protects in both directions. We always recommend protection set to maximum level.

The next two settings control one of two different implementations of despeckle. Use `despeckhorizontal` and `despeckvertical`, or `despeckisolated`. Do not use both at the same time.

despeckhorizontal=0

Removes noise horizontally. Set to the width of larger specks. A negative number tries to protect characters from damage.

despeckvertical=0

Removes noise vertically. Only use if Horizontal De-Speck does not work. Set to the height of larger specks. A negative number tries to protect characters from damage.

This is the recommended implementation of despeckle (as opposed to `despeckhorizontal` & `despeckvertical`). Do not use both implementations at the same time.

despeckisolated=-2

Removes isolated noise. This is powerful and can be used in conjunction with normal dot shading removal. A negative number tries to protect characters from damage (a negative number is recommended). If this value is too large it can remove the "i" dots and/or periods.

---PR_SUBIMAGE

Use these settings to perform manual crop. Manual crop can be used to remove pixels from the left, right, top, or bottom of an image by the number of pixels you specify.

subimagetopedge=0

Number of pixels to crop from top of image. Set to zero (0) to leave edge as is.

subimagebottomedge=0

Number of pixels to crop from bottom of image. Set to zero (0) to leave edge as is.

subimageleftedge=0

Number of pixels to crop from left of image. Set to zero (0) to leave edge as is.

subimagerightedge=0

Number of pixels to crop from right of image. Set to zero (0) to leave edge as is.

subimagepad=0

Number of pixels to add to each edge of image. Set to zero (0) to leave edge as is.

----PR_ROTATE

1=90 degrees, 2=180 degrees, 3=270 degrees.

turnbefore=3

Rotate image before other processing.

turnafter=0

Rotate image after other processing.

----PR_PERIOD_REMOVAL - (Obsolete in PrimeOCR – Only supported in PrimeZone)

Removes “periods” (small black circles) that appear in a sequence. (Some times called dot leaders.)

maxperiodheight=10

Maximum height (in pixels) of “periods” (small black circles) to remove.

expectedfrequency=20

Average number of pixels between the centers of periods.

----PR_SMOOTH**smooth=1**

Does 'Sand and Fill' smoothing of your image. The value indicates the length in pixels of nicks or bumps that will be smoothed. Too large a value will produce a squaring effect on characters. We recommend a value of 1 for most images. 0 disables this feature.

----PR_AUTOROTATE

Use these settings to perform automatic image rotation. PrimeOCR will attempt to recognize the orientation of the image and adjust to the proper orientation.

autoportrait=1

If “1”, this feature will automatically detect if print is portrait oriented. If it is not, the image will be rotated 90° so that it becomes portrait.

autoupright=1

If “1”, this feature will detect if the image is upside down. If it is it will Rotate the image 180°. It works well in conjunction with AutoRotate - Portrait.

autorevert=0

If “1”, any automatic rotation will be undone after all image enhancement is complete.

---PR_CROP

IntelligentCrop can be used to automatically remove thick black or white borders from an image.

cropblack=1

If “1”, this feature will remove black noise from the edge of the image no matter how large the black noise is.

cropwhite=0

This feature removes excess white from around your entire image. The value you supply is the number of pixels of white edge left around the entire image. Note that this happens BEFORE registration. Set to 0 to disable.

---PR_GROWTH_EROSION (or PR_DILATION_EROSION)

This feature allow you to adjust blocks of characters (pixels, actually). Using these functions, you can bring out faint text, de-emphasize bold text, join dot matrix text into something better readable by PrimeOCR, and smooth out the edges of jagged text. Settings that are too large will connect characters, or character features incorrectly, causing OCR errors.

Some users prefer the terminology of “growth” to “dilation”.

dilation=1

Grow image by this amount in the vertical and horizontal directions. Try 4 to join dots in dot-matrix print.

erosion=1

Erode the image in the vertical and horizontal directions (that order). If you are using a value of 4 in dilation, try an erosion of 2 or 3. This will remove excess thickness.

options=

Allows the use of ScanFix string options. NOTE: Not all string options are supported, please contact Prime Recognition for more information.

Testing the Development Environment (Developers Only)

Prime Recognition has supplied a simple application for use in development, called PREXC32.EXE. It is available as source code and in a compiled version. Run the PREXC32.EXE from the \BIN directory. Once the supplied example application runs correctly, try to compile it from the supplied source code and make files. Some compilation issues are covered in *Chapter 10*. If you are not using Visual C++, you will need to consult the documentation for your programming tools. Errors at this point will most likely be a mismatch between the supplied PrimeOCR source code and your development environment.

Again, we very strongly recommend that you do not proceed any further until you are able to get the PrimeOCR example code working in your environment. Once a working example of the code is available then problems with your application can be much more easily isolated to the differences between your code and the example application.

We highly recommend that you use our code as a starting point for your development, if possible, and incrementally modify the code, testing at each significant modification. So long as the PrimeOCR example code is used in conjunction with our engine, you can modify it as much or as little as you like with no royalties or any other legal hindrance.

PATH

If you locate your application in the \BIN directory no PATH modifications are required. If you wish to move your application to another directory then:

- The \BIN directory must be added to the PATH statement. Call Prime Recognition if you need help in changing the PATH operating system settings.



Warning

During the use of a development tool such as Visual C++, set the output file name (for the application) to the \BIN directory. Otherwise, if you let the output executable default to \WINDEBUG, for example, then the engine will fail to initialize since the executable is not in the \BIN directory.

Distributing the PrimeOCR Engine (Developers Only)

Once you have finished developing the application that interfaces to and controls the PrimeOCR engine you will want to distribute the PrimeOCR engine and your program to your production environment. A license is required for each PC in your production environment.

Please contact Prime Recognition to receive a package of installation files (for example in a CD or download). An installation of a run time PrimeOCR engine requires the following setup:

- A \BIN directory with all files that are in the development \BIN directory (except for PRIMAGE.INI, PRIMEOCR.INI, PRBACKUP.*, PRCONFIG.*, OCRJOB*.*, PREXC32.EXE)
- All files from ...OCR\OCRSYS32 CD directory put into ... \WINDOWS\SYSTEM32 directory.
- If you will be processing color/gray scale input files :
 - o A ... \WINDOWS\PIXTRAN directory with all files that are in the CD ...OCR\PIX\PIXTRAN directory.
 - o All files from ...OCR\PIX\SYSDIR CD directory put into ... \WINDOWS\SYSTEM32 directory.

- All files from ...OCR\PIX\SYSTEMNT CD directory put into ... \WINDOWS\SYSTEM32 directory.

-If application is not in the \BIN directory then the PATH must point to \BIN directory.

-Installation of hardware key software drivers (unless you are using Prime's software license system. Note that the software license system requires that Prime generate the license file):

- Run the hardware key software drivers setup program. The program, SETUP.EXE, is located in the "SENT" directory on the installation CD.

- The setup program will auto-detect the operating system and install the correct hardware key software drivers.

- Re-boot the computer after the drivers have been installed.

You may use the Prime Recognition installation program to install a run time license. It may generate subdirectories and files that are not required for your run time installation. You may erase these extra files and directories at your option.

Chapter 4

PrimeOCR Job Server

Overview

The PrimeOCR Job Server is an application, written by Prime Recognition, that controls the PrimeOCR engine. The PrimeOCR Job Server is designed to read "jobs" from a user specified directory, configure and start the PrimeOCR engine processing, and then place the PrimeOCR results in a user specified output directory. If it encounters a recoverable error it writes the error to an

error file and continues processing.

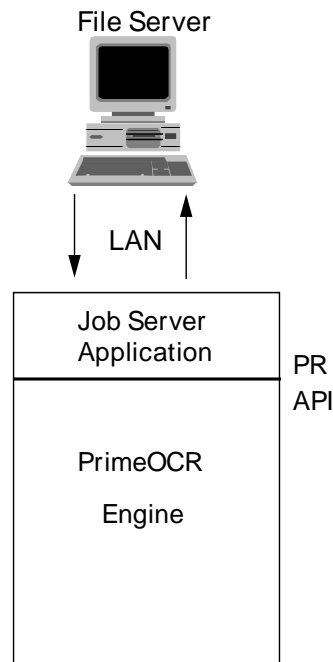
A "job" is a simple ASCII text file that is created automatically by the PrimeView tool, by a user application or manually. At the end of this section is a detailed description of the "job" file format.

Because "jobs" are files, and the output from the PrimeOCR Job Server are files, the PrimeOCR Job Server can be located on the same PC as the application generating the job files, or anywhere on LAN/WAN network where it has file access. Since the job server's architecture is file based it is not dependent on any specific network protocols, or even the operating system of the computer generating the job files. For example, a UNIX machine could be generating the "job" files and submitting them to a PrimeOCR Job Server across a WAN in another state.

The Job Server queue can contain multiple job files, waiting for processing, and new jobs can be added at any time. This means that many PrimeView stations, or user applications, can be sending jobs to the Job Server. The Job Server's simple architecture also means that you can set up multiple Job Servers, so that OCR output can be increased simply by adding more Job Servers.

A typical, simple, test installation will have one PC on the network creating jobs with the PrimeView tool. The jobs will be automatically saved to a user specified directory on a file server. The PrimeOCR Job Server, on a separate dedicated PC on the network, will be polling this directory for jobs. Once it senses a new job the Job Server reads it in and processes it. The results from the PrimeOCR engine are saved to another user-specified directory on the file server.

All of these same steps may take place on a single PC, as might be common in very small applications, or the initial evaluation and testing of the PrimeOCR engine.





Note

OCR is very CPU and memory intensive. If you will be processing a significant amount of documents we recommend that a PC be dedicated to the PrimeOCR engine.

The PrimeOCR Job Server reads the job file to identify which images to process and how to process those images. The "how to" information is described in a "template" file, a simple ASCII based text file. The format and settings of the template file are described later in this chapter.

Key Features

The flexibility of the job and templates files means that a wide variety of processing methods are available including some very powerful techniques, for example:

- Process all images in a directory by specifying "*.tif" in the path
- Process all images in a directory AND subdirectories by specifying "*.tif+"
- Process some images in a directory AND subdirectories by specifying "text*.tif+"
- Use one template file for all images or specify a unique template file for each image
- Multiple Job Servers can work on one job at the same time.

Jobs can be prioritized. Two job input directories exist, one of them is "low priority", any job submitted to this directory only is performed after jobs in the regular directory have been completed.

A report on the average confidence level of each image is created called "CONFID.TXT". This report allows you to, for example, only verify images that fall below a confidence threshold. The image path, page number, average OCR confidence level (100-900), number of characters, and the time required to perform OCR (seconds), is reported on each line in the file.

Multipage TIFF files can be processed. Output for each image is automatically appended into one file so that one multipage TIFF file generates one output file.

Fault tolerance is built in at several levels. The PrimeOCR engine has several fault tolerant features plus the Job Server has several additional features, including automatically handling of "exceptions" (normally would cause a "GPF", Job Server captures exception and keeps on going). For those rare occasions where something gets past all the other safeguards the Job Server has an "Automatic Restart" feature that will relaunch the Job Server automatically.

Email notifications can be sent based on certain events (or lack of events) in the Job Server. These events include:

- Job is completed
- A completed Job has errors
- Predicted Job completion date/time (updated after 1000 pages have been processed)
- "Heartbeat", i.e. hourly updates as to how many pages have been processed
- No Jobs are available to process
- Job Server is not running

Configuration of the email notifications is in the ...\\BIN\\PREOUT.INI file, along with detailed instructions at the bottom of this file.

Simple OCR Job Server Tutorial

1. Double click on the PrimeOCR Job Server icon.
2. Click on "Setup" menu item/button.
3. Configure the Job Server by clicking on each "Setup" menu choice and making choices as noted in the "help" window, in particular point the "Job Directory" to the directory to which you will be placing your job files. (We suggest you leave the default "\\JOB" in place, there is an example job file, "EXAMPLE.JOB" already in that directory.)
4. Click on the "Wizard" button and follow directions to create a new template and/or job. Be sure to save job to Job Directory.
6. Click on the "Start" button.

Starting Job Server

Clicking on the PrimeOCR Job Server icon brings up the main screen. Select the "Start" button to begin processing. To start the engine programmatically (no manual intervention), execute the OCRJOB32.EXE with "START" in the command line. For example,

```
WinExec("ocrjob32.exe START", SW_SHOW);
```

Stopping Job Server

Press "Stop" button during processing. The server will not stop until it has completed the current page. If the server is processing a multipage TIFF, and not all pages have been completed, then a message box will warn the user that if the Stop action is confirmed, the partial output file will be deleted. In other words, all pages processed to date in this file will need to be reprocessed when the job is started up again.

Error Reporting

If any errors occur during processing an "Errors: X" entry will show up in the user interface. The X indicates the number of errors that occurred but were captured, handled, and logged as processing continued.

Errors are logged to a file called "PROCR.LOG". If you are trying to understand any unusual behavior or problems with the Job Server we strongly recommend you review this file.

This file holds the date, time, image file, page number, and explanatory message(s) for a single error on each line in ASCII format. The file is limited to 1MB in size. Once the file reaches this size it is saved to "PROCR.001" and another PROCR.LOG file is started.

The PROCR.LOG, PRIMEOCR.INI (configuration file for PrimeOCR), and CONFID.TXT file are available from the user interface menus.



Note

An error during the processing of a multipage image file will cause the erasure of the output file. This is done to preserve the integrity of the multiple page output file. This means that the OCR results of all pages processed up to that point within that multiple image file are lost.



Note

During processing the Job Server will generate a “XXX.BLK” file in the output directory, where XXX is the file name of the target image. This file is used for several purposes. If the image is successfully processed this file is removed. However, an error in Job Server processing will cause the BLK file to remain. If the Job Server is restarted on this “JOB”, it will look at any existing BLK files, and try to reprocess the corresponding image files. If it fails again on a file the BLK file is modified so that the Job Server will bypass the image file on any further processing. This allows the Job Server to continue onto other jobs, leaving behind flags for the operator that signal which files could not be successfully processed.

Job Status

The Job Server displays the job status of the job being processed and displays a list of jobs that have completed processing. Once a job is done processing the Job Server displays the total of image files processed, if there were any errors associated with the job, and if the Job Server processed all the files described in the job. If the Job Server processed all of the files within a job the job is considered complete.

For users that require job completion information across a network the same tabulated information can be found in the “JOB_RESULTS.LOG” file found in the /BIN directory. This file includes the date and time the job finished processing.

Configuration

Select "Setup" menu item/button from the main screen. Help text is available within the Setup screens.

Creating Jobs & Templates

There are several ways to create a template and/or job files:

1. Click on "Wizard" button and follow directions. This method is fast but does not cover all options nor does it allow you to setup individual zones.

2. Use PrimeView tool. This tool covers almost all options and allows you to draw individual zones with the mouse.
3. Create your own job and template files directly, perhaps manually using a text editor, or automatically through your own program. The following sections will describe the job and template file formats. The job and template file formats are ASCII based and are open formats. You may use these formats without restriction so long as they are used with a Prime Recognition product.

Running as a Service

Some users prefer to run OCR as a Windows Service for security reasons. This allows the product to operate without logging into the PC. See Chapter 12 for more information on running OCR Job Server as a Service.

Job File Format

A job file is, in essence, a five line text file that can be created by the user using a text editor. Prime Recognition's PrimeView tool automatically creates job files, as does the OCR Job Server Wizard, or the developer may create these files. An example job file is included in the \JOB directory as "EXAMPLE.JOB".

The first non header line is the number of image/template pairs in the job file that will be processed. The second line is a path to the first image(s) that must be OCR'd. The third line is a path to the "template" file (zoning information and OCR engine configuration information.) The image path may point to one image, a whole directory, e.g., "C:\IMAGES*.TIF", or a directory and its subdirectories, e.g., "C:\IMAGES*.TIF+". The image file name can also be "*Text*" which will only process images that fit this wildcard criteria. Here are some valid examples: *001.tif, *001*.tif, 001*.tif, etc.

Job files must have a ".JOB" extension.

Example Job File:

Prime Recognition Job File	(Header 1)
Version 3.90	(Header 2)
2	(Number of image/templates pairs to follow)
c:\images\4931.tif	(full path to first image)
c:\templates\test3.ptm	(full path to first template)
c:\images\nov2.tif	(full path to second image)
c:\templates\fullpage.ptm	(full path to second template)

The key concept is an image/template pair that is repeated for as many images that need to be processed. You can process a number of images in one job or put each image into a separate job, there is no appreciable difference from an OCR standpoint.

Special Cases

1. If you specify "C:\IMAGES*.XXX" the OCR engine will process all images in the directory with the ".XXX" extension ("XXX" may be "*", as in "*.*"). This job must be limited to one image/template pair. All images will be OCR'd with the one supplied template. If you append a "+" at the end of the image path then all subdirectories will also be processed (subdirectories found will be processed in alphabetical order, with directory names ordered with case sensitivity). You can specify the template path with a "*.PTM". In this case, template files with the same name as the image file (but with a ".PTM" extension) and in the same directory as the image, will be used for processing.

2. If you leave the template entry blank (there must be a blank line to indicate this) then the OCR engine will OCR the image with the last known template supplied in this job. For example, you can submit a list of images, and just one template, and all images will be processed with this template.

3. Special commands may be used within the job file. Examples include:

-If the symbol "|" is added after the image path, any string placed after this symbol will be reported as a user tag in the first page of FASCIITAG format output.

- If the string "|APPEND" is added after a "*.image_extension" image path (including "*.tif+" for example), all output in a directory will be saved to one collated file in that directory. The name of the output file will be the name of the first file in the directory. Here is an example of the image path line:

```
c:\prdev\images\*.tif |APPEND
```

Special case: If you have selected the output to go to a fixed directory, and have not chosen to preserve the directory structure, then the file name in each input directory must be different, otherwise output files with the same name will overwrite each other.

Do not use this feature when pages might be skipped or processed out of order, for example, High and Low priority job directories are being used, or Skip files with existing output is selected.

If instead of APPEND you use APPEND_DIRNAME, the output file name will be the name of the directory in which the files are located (instead of the first file name).

-If the string "| OUTPUTPATH c:\output" is after the image path, all output will be placed in this directory (preservation of directory structure does not occur). (c:\output is just an example, you can specify the output directory).

-You can use both APPEND (or APPEND_DIRNAME) and OUTPUTPATH in the same job line, for example:

```
c:\prdev\images\*.tif |APPEND |OUTPUTPATH c:\output
```

Template File Format

A template file is a multi line text file that can be created by the user. Prime Recognition's PrimeView tool automatically creates template files as does the OCR Job Server Wizard. An example template file "FULLPAGE.PTM" is included in the \JOB directory.

The first non header line of text sets the OCR configuration items that are relevant across the whole document, the second line describes configuration data for the first image/page within a document, the third line describes the number of zones within the page, the fourth line (and other lines if any) describe the relevant OCR configuration settings for each zone.

If multiple pages exist within a document then the page configuration data line is repeated, followed by the number of zones on this page, zone configuration lines, etc.

Template files can have any extension but the standard extension is ".PTM". (.ACC is a special template format for use in creating PDF files that conform to Section 508 accessibility requirements. See Chapter 5, PDF_DEFAULTS for more information).

Example Template File:

Prime Recognition Document Template	(Header 1)
	(Header 2 - not currently used)
Version 3.90	(Header 3)
3,2	(Document Configuration Settings)
1,0,0,0,10,1,12,0,0,0	(Page 1 Configuration Settings)
2	(Number of zones on Page)
0,0,1,090099,100,200,6700,5000, R95T	(zone 0 description)
1,0,1,090099,100,5000,6700,9000	(zone 1 description)
2,0,0,0,10,0,3,0,0,0	(Page 2 Configuration Settings)
1	(Number of zones on Page)
0,0,1,999999,100,200,500,5000	(zone 0 description)

Document Configuration Settings

Based on 2 numbers separated by commas.

<u>Label</u>	<u>Example</u>	<u>Description</u>
Output Format	"3"	"0" ASCII
		"1" Formatted ASCII (add left edge space and blank lines to mimic original image).
		"2" PDA
		"3" Prime OCR format (".pro") format
		"4" ASCII but no lines separate zones
		"5" RRI3 (RRI Version 3)
		"6" FASCII with user tag at top of page
		"7" ZyIndex Version 3.0 format
		"8" PRSTAR format
		"9" RTF format
		"10" ASCIITAG format
		"11" PDF - Normal
		"12" PDF - Image Only
		"13" PDF - Image plus hidden text
		"15" Comma Delimited
		"16" HTML
		"17" FASCII with average confidence of document on first line of file
		"18" FASCII without white space improvement logic
		"19" Tab delimited – tabs inserted between output of each zone
		"20" XML/Word – Words reported with associated coordinates (in pixels), UTF-8 characters
		"21" XML/PRO – PRO format except in XML style, UTF-8 characters
Select PrimeOCR format if you want confidence levels, character data, and other data included in the output. See <i>Chapter 6 – PrimeOCR Output Formats</i> for more information.		
# Pages	"1"	Enter number of pages that will be OCR'd. Note that not all pages in a multipage TIFF file must be OCR'd. "-1" Means process all pages within document with following settings. Relevant if you want to process all pages in a multipage TIFF file with same settings.

Page Configuration Settings

Based on 10 numbers separated by commas.

<u>Label</u>	<u>Example</u>	<u>Description</u>
Page Number	"1"	Page number of image that appears in original TIFF file. If a multipage TIFF file then this number may be between 1 and # of images in multipage TIFF file. It does not have to be consecutive.

Document Source	"0"	"0"	This page is not from the same document as the last page processed by the engine.
		"1"	This image is from the same document, it has exactly the same fonts as the previous image processed by the engine.
			Most users will set to "0".
Image Quality	"0"	"0"	Default quality setting.
		"1"	Poor quality setting.
			For the vast majority of documents "0" is the correct choice.
Print Type	"0"	"0"	Machine Print
		"1"	Dot Matrix Print
		"2"	Machine Print Fixed Pitch
			Do not select dot matrix unless the text is a series of unconnected dots, i.e. 9 pin Dot matrix, or badly broken characters.
Language	"10"	"0"	Danish
		"1"	Dutch
		"2"	French
		"3"	German
		"4"	Italian
		"5"	Norwegian
		"6"	Portuguese
		"7"	Spanish
		"8"	Swedish
		"9"	U.K. English
		"10"	U.S. English
		"12"	Chinese (Simple)
		"13"	Chinese (Traditional)
		"14"	Japanese
		"15"	Korean
		"16"	Russian
			Special Cases:
		"-1"	Automatically identify and set the main language based on each page in a document. If Asian Languages are installed it will check all languages (rather slow). This value is created by the Wizard when Automatic ID is selected.
		"-2"	Same as -1 above except no Asian languages are checked, even if they are installed (significantly faster than -1).
		"-16"	If this number is added to above numbers, Russian Language will be included in the ID (again decreases speed further). For example, -18 (-2 + -16) will check European languages and Russian.
		"-100"	If this number is added to the above numbers, only the first page in a multiple

document will be checked. All later pages will use the language identified on page 1. For example, -117 (-100 + -1 + -16) will check all languages, including Asian and Russian on the first page only. All later pages will be recognized with the language identified on page 1.

Deskew	"1"	"0"	Do not perform Deskew
		"1"	Perform Deskew if skew is > 1%.
		"2"	Perform Deskew but do not adjust any zone coordinates supplied in the template for the amount of skew. This setting is appropriate if you are sending a "static" template to the engine that is good on deskewed images.
		"3"	Perform Deskew but do not adjust any output coordinates supplied in the PRO file for the amount of skew. This means that the data in the PRO file will match the deskewed image, not the original image supplied to engine.
		"4"	A combination of "2" and "3".

Image PreProcess	"12"	Add together the values of each desired image preprocessing steps noted below.	
---------------------	------	--	--

"1"	PR_HORIZONTAL_REGISTER
"2"	PR_VERTICAL_REGISTER
"4"	PR_HORIZONTAL_LINE_REMOVAL
"8"	PR_VERTICAL_LINE_REMOVAL
"16"	PR_INVERSE_TEXT
"32"	PR_DESHADE
"64"	PR_DESPECK
"128"	PR_SUBIMAGE
"256"	PR_ROTATE
"1024"	PR_SMOOTH
"2048"	PR_AUTOROTATE
"4096"	PR_CROP
"8192"	PR_GROWTH_EROSION

For example, "12" is PR_HORIZONTAL_LINE_REMOVAL(4) + PR_VERTICAL_LINE_REMOVAL(8).

This setting will turn on the feature(s). To configure a feature you must edit the \BIN\PRIMAGE.INI file. The settings of the PRIMAGE.INI file are exactly those of the ScanFix product and are documented in *Chapter 3*.



Warning!

Be conservative in selecting image enhancement steps. Only select steps that you know are beneficial. Many image enhancement steps can harm image if target "defect" is not actually present. A common misconception is that image enhancement steps are automatic and "safe". This is not the case.

AutoZone	"0"	"0"	No Autozone
		"1"	PR_SEGMENT

- "2" PR_ZONE Find only text zones and report in order of top to bottom.
- "3" PR_CGS Find only text zones and report in "reading order".
Custom zoning version. Call Prime for more information.
- "4" PR_SEGMENT_IMAGE_TEXT Find text and image zones. Report in order from top to bottom. (See *Image Zone Numbering* section later in this Chapter).
- "5" PR_ZONE_IMAGE_TEXT Find text and image zones. Report in reading order. Image zones are saved in same directory as original image with the same name as the image plus an extension that denotes the number of the image in this document (See *Image Zone Numbering* section later in this Chapter).

Auto Rotation "0" "0" Do not Auto Rotate
"100-900"
OCR confidence value below which auto rotation sensing is performed. "700" is a good setting.
This auto rotate feature is much more accurate but a lot slower than PR_AUTOROTATE feature described in ImagePreProcess above. If only a small percentage of images are rotated then use this feature, as Auto rotation will only be attempted on a small number of images and therefore speed will not be a problem. If you have a lot of images that are rotated use both PR_AUTOROTATE (for speed) and Auto Rotation (to catch and correct PR_AUTOROTATE mistakes).

Save Image "0" "0" Do not save image
"1" Save image that is in memory after image processing and after recognition. This will save the image to the output path (note that this may be or may not be same path as original image) with the same file name as the original image except with a ".FIX" extension.
"2" Saves image with same name as original file including extension. Note that if output is going back to directory in which original image file was present, this setting will cause the original image to be overwritten by new file.

Number of Zones on Page

A single number.

"2" The number of fields or zones on this page of image. The configuration data of these zones is in "Zone Description" below.

Zone Description

Based on 8 numbers separated by commas. An optional ninth string can be added to this description.

<u>Label</u>	<u>Example</u>	<u>Description</u>
Zone Number	"0"	Zone numbers must be consecutive from 0 to number of zones -1.

Zone	"0"	"0"	No restrictions
Restrictions		"1"	Alpha only
		"2"	Numeric only
		"3"	No restrictions - upper case
		"4"	No restrictions - lower case
		"5"	Alpha only - upper case
		"6"	Alpha only - lower case
		"7"	OMR
		"8"	Image

See *Appendix A* for a listing of characters that fall within each restriction.

Other Settings

- "9" Anchor Region – Used by PrimeZone
- "10" Anchor Region – Used by PrimeZone
- "11" Barcode. See prbarcode.ini for configuration information.

Lexical Check	"1"	"0"	No lexical check, however, extra weight is given to characters that fall within each restriction.
		"1"	Perform lexical check on output. Any character that is not within restriction group is changed to a "?" and given a confidence value of "1". Lexical check is not valid for OMR, or Image zones.
		"1X"	Perform Lexical Check PLUS on output where X can range from 1-9 and represents the confidence level of a character in a word that will trigger a Lexical Check PLUS of the word. Lexical check PLUS configuration is controlled by the PRLEXICA.INI file in the INSTALLATION\BIN directory. This file also documents what functions the Lexical Check Plus performs. (See <i>Lexical Check</i> section in Chapter 1). Finally, special features of Lexical Check PLUS (or Lexical Plus) are implemented in LEXPLUSPRO.INI in the INSTALLATION\BIN directory.

Accuracy Level "990099" Range from "1" (i.e. "000001") to "999999"

Determines which engines run and what weight they are given in processing. For example, "699077" would mean:

- run sixth engine, give it a 6 weight
- run fifth engine, give it a 9 weight
- run fourth engine, give it a 9 weight
- do not run third engine ("0")
- run the second engine, give it a 7 weight
- run the first engine, give it a 7 weight

HOWEVER, we strongly recommend most users should give each engine licensed a value of "9" and stay with standard configurations as follows:

Level 1	9
Level 2	99
Level 3	90099

		Level 4	99099
		Level 5	99999
		Level 6	999999
Left Dimension	"0"	"0"	Automatically sense the left edge of the image (useful when you just want to OCR the whole document as one zone. You cannot enter "0" if there is more than one zone.)
			Otherwise you should enter the left edge of the zone on the image in units of 1/1200 of an inch. For example, if the zone starts one inch in from the left edge the value would be "1200". A negative number means move in from the left edge of the page by this amount.
			Proper values must be >-32000 and <32000.
Top Dimension	"0"		See left dimension description.
Right Dimension	"0"		See left dimension description.
Bottom Dimension	"0"		See left dimension description.
Zone Text	"R95T"		Optional string. This string can be up to 100 characters in length. If this string is part of an ACC style PTM file (see PDF_DEFAULTS in Chapter 5), then this alternative text for a zone in a PDF file. Otherwise this text is ignored by PrimeOCR.

We can easily extend the job, template, or zone definitions to include other configuration issues. Please contact Prime Recognition to discuss your specific needs.

Image Zone Numbering

Image zones are saved as G4 TIFF files (if the original image was color/grey scale/or non TIFF format, then image zone will be a JPEG file) in the output directory with the same file name as the image plus an eight character extension that denotes the page number and image zone on that page. The first four characters of the extension refer to the page number. The last four characters refer to the zone number on that page. (Zone numbers start at 0). For example, an image zone on the 12th page of the document called "TEST.TIF", that has a zone number of 2, will be named "TEST.00120002". The "0012" refers to the 12th page in the document, and "0002" refers to the 3rd zone on that page. (The first zone is 0, the second zone is 1, the third zone is 2, etc.) An image zone on page 37 with a zone number of 15 would have an extension of ".00370015".



WARNING!

Several advanced features of the Job Server can be confusing and cause apparently mysterious failures:

1. Erasing of files. Job, image, and template files can be erased by the Job Server (if so configured by user).
 2. You must coordinate directories carefully. For example, the Job Server will only look for jobs in the Job Directory, and will place output only in the selected OCR Output directory(s).
-

Customization Features

If you are interested in the following features please contact Prime Recognition. These capabilities exist within the product but are only available on a custom basis.

- System vendors may change labeling of Job Server to their own product names, company names, and contact information.
- Custom scripts of image preprocessing steps can be created which allow any arbitrary sequence of image preprocessing steps (including multiple calls to the same process) to be performed. (For example, in some applications the order of preprocessing steps is important and the best order may be different than the native order.)

Chapter 5

String Options

Overview

PrimeOCR includes a feature called “String Options”. This feature allows Prime Recognition to quickly and easily add functionality (often times custom functionality) to the PrimeOCR engine without changing the API or other important design parameters of the engine.

For users, string options are entered in the “Engine Mode” setup option. For developers, string options are entered via the `pr_string_options()` call.

The remainder of this chapter details the public string options that are available. Note that the functionality and timing of each call is highly variable.

If you need functionality not described in this Guide please contact Prime Recognition. We may be able to customize the engine to your application’s needs.



Note!!

Many key String Options are now configuration options in the PrimeOCR Job Server Setup program. We highly recommend you use the Setup to configure the required functionality instead of directly calling the string option.



Note

String Options are very sensitive to case and format. Review carefully the examples given for each string option and make sure to mimic these formats precisely.

String Option: Change Characters in Output

Function:

Changes the characters in PrimeOCR output before writing output to file (i.e. while it is still RAM based). Can be used to change one character, or repetitive single characters, in one zone or across the whole page, to a user/developer specified character. This function cannot convert a character to more than one character.

Timing:

Users: This is a “Post Recognition” string.

Developers: Can be made any time after a `pr_recognize_image()` call and before the corresponding `pr_del_results()` call. Most developers will want to call this before writing the output to file, or converting it to PDA format, etc.

This call is not “sticky”, as are many PrimeOCR configuration calls. In other words, you must explicitly call it for every page you want processed.

Input Variables:

The command string must include the following parameters separated by commas:

- Name of the command (must be “change_chars”)
- Target character to be replaced
- Number of consecutive target characters that must be found before qualifying for replacement
- Replacement character
- Zone number in which to perform replacement (-1 causes whole page to be processed)

Return Value:

No return values.

Example Usage:

Users:

```
change_chars, ., 2, , -1
```

Developers:

```
status=pr_string_options("change_chars, ., 2, , -1", NULL);
```

In this example two or more consecutive periods will be replaced with spaces in all zones in a document. If you wanted to replace all “R” characters with “B” only in zone 3 you would change the period, “.”, in the example string to “R”, change the “2” in the string to “1”, change the “ ” to “B”, and change the “-1” to “3”.

Notes:

Only the character is replaced, no changes are made to confidence level, character attributes, or any other output.

A "," entry for the Replacement character, that is, no entry, causes the Target Character sequence to be cut out.

String Option: Change Strings in Output

Function:

Changes any string in PrimeOCR output before writing output to file (i.e. while it is still RAM based). Can be used to change a single character string or a multicharacter string, in one zone or across the whole page, to a user/developer specified string.

Timing:

Users: This is a “Post Recognition” string.

Developers: Can be made any time after a `pr_recognize_image()` call and before the corresponding `pr_del_results()` call. Most developers will want to call this before writing the output to file, or converting it to PDA format, etc.

This call is not “sticky”, as are many PrimeOCR configuration calls. In other words, you must explicitly call it for every page you want processed.

Input Variables:

The command string must include the following parameters separated by commas:

- Name of the command (must be “change_string”)
- Target string to be replaced
- Replacement string
- Zone number in which to perform replacement (-1 causes whole page to be processed)

Return Value:

No return values.

Example Usage:

Users:

```
change_string,south,south west,-1
```

Developers:

```
status=pr_string_options("change_chars,south,south west,-1",  
NULL);
```

In this example the sequence of characters “south” (which happens to be a word but could instead be a phrase if needed), is replaced by “south west” .

Notes:

Only the character is replaced, no changes are made to confidence level, character attributes, or any other output.

A "," entry for the Replacement string, that is, no entry, causes the Target String sequence to be cut out.

String Option: Cut Repetitive Characters in Output

Function:

Removes repetitive characters in PrimeOCR RAM based output. Can be used to remove repetitive characters, in one zone or across the whole page. After processing, one character is left. This is most commonly used to remove extra spaces in output. (The "change_chars" string option can cut characters but it cuts all characters, it does not leave one character behind.)

Timing:

Users: This is a "Post Recognition" string.

Developers: Can be made any time after a `pr_recognize_image()` call and before the corresponding `pr_del_results()` call. Most developers will want to call this before writing the output to file or using results in RAM.

This call is not "sticky", as are many PrimeOCR configuration calls. In other words, you must explicitly call it for every page you want processed.

Input Variables:

The command string must include the following parameters separated by commas:

- Name of the command (must be "cut_repetitive_chars")
- Target character
- Number of consecutive target characters that must be found before qualifying for cut
- Zone number in which to perform replacement (-1 causes whole page to be processed)

Return Value:

No return values.

Example Usage:

Users:

```
cut_repetitive_chars,.,2,-1
```

Developers:

```
status=pr_string_options("cut_repetitive_chars,.,2,-1", NULL);
```

In this example two or more consecutive periods will be cut to one period.

Notes:

String Option: Various Statistical Measures

Function:

Separate calls that can calculate:

- average confidence level of all non space characters in a zone or page.
- total number of recognized characters in a zone or page.
- total number of characters in a zone or a page with confidence less than "x".
- total number of words in a zone or a page with one or more characters with confidence less than "x".

Most of these results are already reported to a user of the PrimeOCR Job Server via the CONFID.TXT file. In any case there is no means by which the user can get return data from this function so it is irrelevant to a user.

Timing:

Developer: Can be made any time after a `pr_recognize_image()` call and before the corresponding `pr_del_results()` call.

Input Variables:

The command string must include the following parameters separated by commas:

- Name of the command
- Zone number in which to perform calculation (-1 causes whole page to be processed)
- Confidence level (on some calls, see below)

Return Value:

The function returns the calculated value. Negative numbers indicate an error.

Example Usage:

AVERAGE CONFIDENCE

```
status=pr_string_options("average_confidence,-1", NULL);
```

(In this example the average confidence will be reported for the whole page.)

A return value of 0 means that no non-space characters were found in the output. The range of values is 100-900, with 900 indicating extremely high confidence.

NUMBER OF CHARACTERS RECOGNIZED

```
status=pr_string_options("number_of_characters,-1", NULL);
```

Includes space characters. (To measure the total number of non space characters use the "character_markers" option noted below and use a confidence of 9. All characters have a confidence of 9 or less.)

NUMBER OF CHARACTERS WITH CONFIDENCE EQUAL TO OR LESS THAN TARGET

```
status=pr_string_options("character_markers,-1,6", NULL);
```

This example measures the number of characters with confidence equal to or less than 6 in the whole page. The proper range of confidence levels is 1-9, with 9 meaning very high confidence. Space characters are not included in this measure.

NUMBER OF WORDS WITH 1 OR MORE CHARACTERS BELOW TARGET CONFIDENCE LEVEL

```
status=pr_string_options("word_markers,-1,6", NULL);
```

This example measures the number of words with one or more characters with confidence equal to or less than 6 in the whole page. The proper range of confidence levels is 1-9, with 9 meaning very high confidence.

String Option: Variable Processing of Engines based on Image Quality

Function:

Allows user/developer to control which engines run based on the automatically sensed quality of the image. This function has several applications, for example:

- Engines after the first engine do not run on very bad quality images. PrimeOCR does not waste a lot of time on an image that will be key entered or otherwise processed because of its poor quality.
- As image quality increases the number of engines running decreases until at extremely high quality images only one engine is running. This is good for "fuzzy search" applications that do not employ human OCR error verification. The user/developer is happy with the OCR quality of one engine on extremely high quality text (and wants the speed of one engine) but wants voting OCR to help out on lower quality images.

Timing:

User: This is a "Pre Recognition" string option.

Developer: The call must be made before the `pr_recognize_image()` call for each image. This call must be made for each relevant image, i.e., the setting is not "sticky" between images.

Input Variables:

The command string must include the following parameters separated by commas:

- Name of the command (must be "cut_process_short")
- Zone number in which to perform calculation (-1 causes whole page to be processed)
- For each of six engines:
 - Average lowest confidence that causes each engine to run expressed in units from 100-900.
-","
 - Average highest confidence that causes each engine to run expressed in units from 100-900.
-","

(There must be six of these number pairs, even if your PrimeOCR engine is only licensed for three or four engines. These numbers are sequenced to control engines in the following order: Engine 1, Engine 2, etc.)

Return Value:

No return values.

Example Usage:

User: cut_process_short, -1, 600, 900, 600, 800, 600, 750, 800, 900, 700, 900, 750, 900

Developer: status=pr_string_options("cut_process_short, -1, 600, 900, 600, 800, 600, 750, 800, 900, 700, 900, 750, 900", NULL);

In a five engine configuration this string causes the following actions: The first engine runs. (Its settings of 600,900 are ignored.) The average confidence for the whole page is calculated. If the average confidence for the whole page is 600 or above (equal to "6" on the character based scale of 1-9) and 800 or below then Engine 2 will run (settings 600,800). If it is equal to or above 600 and 750 or below then Engine 3 will run (settings 600,750), and so on.

Assume a three engine scenario: Engine 2, Engine 3, and Engine 5. In this scenario the settings for Engine 1 and Engine 4 are ignored (but they must be present.). Engine 2 is the first engine to run. So its settings are ignored (settings 600,800). (The first engine always runs regardless of its setting.) If Engine 2 senses the average confidence to be between 600-750 then Engine 3 runs, if the confidence is between 700-900 then Engine 5 will run.

Special Case Usage:

User: cut_process_short, -1,5

Developer: status=pr_string_options("cut_process_short, -1,5", NULL);

This string is equivalent to entering "600,900" for each engine. This special case is supported for ease of use. This version's basic effect is to prevent engines from running on poor quality images.

Notes:

- The settings for the first engine to actually run will be ignored. The first engine must always run in order to sense the quality.
- For most users an average confidence of 600 or below is a good description of a bad document that does not merit further OCR processing. For most users a setting of -1 to read the whole page, instead of a specific zone, is most appropriate.
- Remember that you have to enter six pairs of numbers, even if you have only licensed three engines. Numbers for the engines that are not licensed/used will be ignored.
- You cannot change the running order of engines in our standard product. If you wish to do this contact Prime Recognition for a custom version.
- A string that progressively cuts off processing on very high quality images might be:
"cut_process_short, -1, 600, 900, 600, 890, 600, 800, 600, 825, 600, 875, 600, 850"
- As of version 4.4, PrimeOCR, by default is set internally for:
"cut_process_short, -1, 500, 900, 500, 900, 500, 900, 500, 900, 500, 900, 500, 900"

String Option: Change LOG File Settings

Function:

Changes the path/file name of log file and maximum size. The log file writes all major activities of the engine to a text file (if it is turned on in the PRIMEOCR.INI file).

Timing:

Users: This is a “Pre Recognition” string option.

Developers: This call can be made any time after the `pr_open()`. Any output to the log file before this call, including the output made during the `pr_open()` call, will be made to the default log file name and location.

Input Variables:

The command string must include the following parameters separated by commas:

- Name of the command (must be “log_settings”)
- Path/file name of log file
- Maximum size of log file (in K bytes)

Return Value:

No return values.

Example Usage:

Users:

```
"log_settings,c:\newlog.txt,1300"
```

Developers:

```
status=pr_string_options("log_settings,c:\newlog.txt,1300", NULL);
```

In this example the new log file path/name is "c:\newlog.txt" and the maximum size is 1300K , i.e., 1.3MBytes.

Notes:

- Default log file name is "\BIN\PROCR.LOG".
- Default log size is 10MB (10,000K). Once maximum log size is reached the existing log file is renamed to "xxx.001" where "xxx" is the old log file name (without extension). Then a new log file is started with the appropriate name. THIS MEANS THAT RELEVANT LOG FILE ENTRIES MAY BE IN THE "XXX.001" FILE. For example, if the engine is stopped immediately after the log file was renamed to "XXX.001" then all relevant log file entries will be in the "XXX.001" file, not the new, and blank, log file.
- Turning on the log file slows down the operation of the engine. We recommend you only turn on the log file for diagnostic purposes.

String Option: Phone Books

Function:

Automatically corrects many errors that occur in the OCR of Phone Books.

Timing:

Users: This is a "Post Recognition" string.

Developers: This call can be made anytime after the `pr_recognize()` and before `pr_del_results()`.

Input Variables:

The command string must include the following parameters separated by commas:

- Name of the command (must be "phone_book")
- Zone number in which to perform calculation (-1 causes the entire page to be processed)

Return Value:

No return values.

Example Usage:

Users:

```
"phone_book, -1"
```

Developers:

```
status=pr_string_options("phone_book, -1", NULL);
```

In this example all defined zones will be processed.

Notes:

The dashed lines common in phone books between the address and phone number hurt OCR accuracy considerably. If possible, remove these lines with PrimeZone, or some other image utility before submitting image to PrimeOCR.

String Option: ASCII Defaults

Function:

Changes some of the settings used to report ASCII, FASCII and derivative file output types.

Timing:

Users: This can be placed in either “Pre Recognition” or “Post Recognition” string option areas.

Developers: This call can be made anytime before `pr_convert_output_to_file()`. This call is “sticky”, it only needs to be called once.

Input Variables:

The command string must include the following parameters separated by commas:

- Name of the command (must be “ASCII_DEFAULTS”)
- Wrapped paragraphs

Optional:

- Page end string to replace existing page end character (`\f` in C language). Can be up to 100 characters long or as short as one character. If you wish to hide the page break then we would recommend one space character.

Return Value:

No return values.

Example Usage:

User:

```
"ASCII_DEFAULTS,1,page break"
```

Developer:

```
status=pr_string_options("ASCII_DEFAULTS,1,page break", NULL);
```

In this example the Wrapped Paragraphs setting of 1 will result in soft carriage returns in text so that text will flow into paragraphs. Hyphens will be removed in words that are hyphenated between lines. At the end of each page "page break" will appear.

Notes:

String Option: HTML Defaults

Function:

Changes some of the settings used to report HTML file output.

Timing:

Users: This can be placed in either “Pre Recognition” or “Post Recognition” string option areas.

Developers: This call can be made anytime before `pr_convert_output_to_file()`. This call is “sticky”, it only needs to be called once.

Input Variables:

The command string must include the following parameters separated by commas:

- Name of the command (must be “HTML_DEFAULTS”)
- Wrapped paragraphs
- No Font reporting

Return Value:

No return values.

Example Usage:

User:

```
"HTML_DEFAULTS,1,1"
```

Developer:

```
status=pr_string_options("HTML_DEFAULTS,1,1", NULL);
```

In this example the Wrapped Paragraphs setting of 1 will result in soft carriage returns in text so that text will flow into paragraphs. Hyphens will be removed in words that are hyphenated between lines. The value of "1" in No Font reporting means that font information will NOT be included in HTML file (default font will be used to display text). A value of "0" does include font information (this is default setting).

Notes:

String Option: RTF Defaults

Function:

Changes some of the settings used to report RTF files output.

Timing:

Users: This can be placed in either “Pre Recognition” or “Post Recognition” string option areas.

Developers: This call can be made anytime before `pr_convert_output_to_file()`. This call is “sticky”, it only needs to be called once.

Input Variables:

The command string must include the following parameters separated by commas:

- Name of the command (must be “RTF_DEFAULTS”)
- Page Width
- Page Height
- Left Margin
- Top Margin
- Right Margin
- Bottom Margin
- Proportional Font (report proportional font output as this font)
- Proportional Font – sans serif (report sans serif proportional font output as this font)
- Fixed Pitch Font (report fixed pitch font output as this font)
- Wrapped paragraphs

Return Value:

No return values.

Example Usage:

User:

```
"RTF_DEFAULTS,9600,12000,600,660,8400,11400,Times New Roman,Arial,Courier New,1"
```

Developer:

```
status=pr_string_options("RTF_DEFAULTS,9600,12000,600,660,8400,11400,Times New Roman,Arial,Courier New,1", NULL);
```

In this example the page width will be reported as 8 inches (9600 BMUs), no matter what the real width of the page was. (If the real width of the page was greater than 8 inches then you may get potentially confusing word wraps and extension of page into 2 pages). The page height will be reported as 10 inches, the left margin as .5 inch, top margin as .55 inch, right margin as 7 inches (or one inch from right edge as sometimes expressed

by word processors), bottom margin as 9.5 inches. The proportional font used in the RTF file will be Arial, and the fixed pitch font will be Courier New.

The Wrapped Paragraphs setting of 1 will result in soft carriage returns in text so that text will flow into paragraphs. The default setting of 0 causes text to be reported only on the line as seen on the original document (this is accomplished through the use of hard carriage returns at the end of each line of text). Hyphens will be removed in words that are hyphenated between lines.

Notes:

- All coordinates are expressed in BMUs and are relative to the top left hand corner of the page.
- Any coordinate can be listed as "-1". In this case the real value is reported.
- Any coordinate can be "-99", in which case ALL coordinates will NOT be reported in the RTF output.
- If a coordinate is listed this fixed coordinate will be reported for all pages of OCR output in the RTF output.
- Make sure to use the exact spelling of the fonts used in your target application (such as a word processor).
- Most users will also want to use the "point_size_change" string option to reduce the font size reported in the RTF file.

String Option: PDF Defaults

Function:

Configures several settings used for outputting PDF files.

Timing:

Users: This can be placed in either “Pre Recognition” or “Post Recognition” string option areas.

Developers: This call can be made anytime before `pr_convert_output_to_file()`. This call is “sticky”, it only needs to be called once.

Input Variables:

The command string must include the following parameters separated by commas:

- Name of the command (must be “PDF_DEFAULTS”)
- Suspect Confidence Level (0-9)
- DeSample DPI (-1 if want to leave as is, otherwise 100-600 recommended)

The command string can (optional) include the following parameters separated by a comma:

- Optimize (0=no optimization, 1=Optimize)
- Thumbnails (0=do not include, 1=generate and include Thumbnails)
- Zip Compression (0=do not use Zip Compression, 1=Use Zip Compression)
- Accessibility (0=do not add Accessibility features, 1= Add/use Accessibility features).
- PDF/A (0=standard format, 1= PDF/A 1b format, 2=PDF/A 1a format)
- JBIG2 compression

Return Value:

No return values.

Example Usage:

User:

```
"PDF_DEFAULTS,6,200"  
"PDF_DEFAULTS,6,200,1,1,1,0,1,1"
```

Developer:

```
status=pr_string_options("PDF_DEFAULTS,6,200", NULL);  
status=pr_string_options("PDF_DEFAULTS,6,200,1,1,1,0,1", NULL);
```

In this example any word with one or more characters with a confidence level of 6 or less will generate a snippet of the word in the PDF Normal output. (Suspect Confidence Level is irrelevant in PDF Image Only or PDF Image plus hidden text). Any images that

are present in the PDF output will be reduced in resolution to 200 dpi. For example, if the original image was at 300 dpi and the user selects PDF Image Only then the resulting image in the PDF file will be at 200 dpi, significantly smaller in file size than the original image. The file will be optimized (“byte served” or “linearized”) for quicker viewing of file across the Internet. Thumbnails will be inserted into the file. Zip compression will be used to compress the image (if it is a color or gray scale image). The file will not be generated to conform to Accessibility guidelines, but it will conform to PDF/A (class 1b) standards. The file will be compressed with JBIG2 (assuming it is black and white).

NOTES:

OPTIMIZATION For optimization to work, Acrobat must be installed on the same machine as PrimeOCR.

THUMBNAILS: While viewing a PDF file, Acrobat Reader 5.0 and later will automatically generate thumbnails. Acrobat Reader 6.0 and later will automatically generate thumbnails when viewing a PDF file within a browser. In general, you do not need to turn on thumbnail support unless you want to embed thumbnails into the PDF file for faster thumbnail viewing. Thumbnails will only be embedded into the PDF file for image plus hidden text PDF output.

ZIP COMPRESSION Only use Zip compression if lossless compression is required for color and grayscale PDF output. Using Zip compression will increase PDF file size by 40-100% and processing time will also increase by the same percentage due to Zip compression taking longer than the default, lossy, faster JPEG compression.

JBIG2 COMPRESSION JBIG2 compression can produce file sizes that are 2-10 times smaller than standard compression PDF files. JBIG2 by definition only applies to black and white images/pages. This option significantly slows down processing, particularly on poor quality documents, the delay will be noticeable on the last page of a document and selected pages within the document (i.e. speed is “lumpy”). Note that jbig2 is a “lossy” compression method and hence will most likely not be acceptable for documents that have a legal status. Specifically jbig2 may make substitution errors (for example, a poor quality “e” may be interpreted and rendered as a “o”). PrimeOCR’s implementation of JBIG2 has a low error rate, very competitive with the best industry JBIG2 implementations, however, errors are possible. NOTE: DeSample DPI setting is ignored when JBIG2 is turned on. File sizes do not improve significantly (and can get larger) if DeSample is turned on (and processing time is increased and displayed image quality is lower, so there is no benefit to Desample when JBIG2 is used).

PDF/A: PDF/A is not supported for PDF Normal (PDF Text Only format). PDF Normal is almost never used in an OCR environment (PDF Image + Text is by far the most commonly used format) so this lack of support is should not be of great concern. If you would like us to support PDF Normal as well please contact us. PDF Image Only is supported.

ACCESSIBILITY: PrimeOCR PDF output offers a number of features that increase the accessibility of the contents of the PDF file for people with disabilities. These features also help government agencies and businesses comply with US government regulations as outlined in Section 508 of the Rehabilitation Act.

508 requirements:	PrimeOCR PDF output:
<p>A text equivalent for every non text element shall be provided</p>	<p>Alternate text provided for each significant graphic on the page</p> <p>Alternate text can be provided by the user for each zoned graphic on the scanned image (using PrimeView or other method - See "Manual" below).</p> <p>If alternate text is not provided by the operator from manual zoning then PrimeOCR automatically inserts a default string "This is a graphic from a scanned page". the alternate text can be modified later with a PDF tag editor.</p>
<p>Documents shall be organized so they are readable without requiring an associated style sheet</p>	<p>Reading order is identified and tagged in the PDF file along with paragraph markers.</p> <p>Reading order is determined from autozoning the scanned page (or set by the user manually using PrimeView or alternative editor).</p> <p>The language of the document is identified and tagged for each page. (The language for the page is from the language setting in the processing template or API call.)</p> <p>Text retains formatting so when the OCR text is exported to RTF from a PDF viewer the output text reflows within paragraphs and character formatting is retained.</p>
<p>Row and column headers shall be identified for data tables</p>	<p>Rows and columns are defined and tagged in the PDF file.</p> <p>Rows and columns must be manually defined (using PrimeView or other method - see "Manual" below) within the template or autozoning will segment the table as a single text zone or as several text zones.</p>

Another way to summarize the functionality is by the type of processing: automatic (autozoning), or manual.

Autozoning

If you use autozoning during OCR, PDF accessible output will include:

- a. Reading order of the text that is recognized during OCR. Text will be identified and tagged in the PDF output file.
- b. Paragraphs will be identified within the text for later export into another application. Formatting of the text is retained during export.
- c. Graphics within the scanned page will be identified and tagged with a generic alternative text tag - "This is a graphic from a scanned page". The alternative text for the graphic can be easily modified later with PDF editing software.
- d. Tables will not be recognized or tagged when using autozoning. Tables can only be identified by manually zoning the page prior to OCR taking place.

Manual

To provide alternate text for each graphic on a scanned page and to identify tables in the scanned document then an "accessible" template file must be provided to PrimeOCR for each page scanned. The alternate template file can be created by manually zoning the page with PrimeView then adding additional information to the zone line description or by manually creating the template file with a text editor. This file must be in the same directory as the image, and must have the file name of "imagefilename.acc" where imagefilename is the actual image file name, for example, "C:\TEST\00001.TIF" image would have an accessible template file name of "C:\TEST\00001.ACC".

Alternate text for image zones and Table rows and table columns designators are passed into PrimeOCR from the last field of each zone definition in the template file. Four accessible zone descriptions are available in that last field:

1. Text - identify the zone as text when only text is recognized in the zone.
2. Table Row - identify the zone as a table row when beginning a new row of a table. The zone will be the first column in the new row.
3. Table Column - identify the next zone as a table column for the next column in row that has been identified by a new row (Table Row).
4. Figure - identify the zone if the zone is an image zone and provide the alternate text for the scanned graphic in the image zone.

Here is an example:

Prime Recognition Document Template

Version=3.90

3,1

1,0,0,0,10,0,0,0,0,0,

11

0,0,1,999999,4708,1344,8268,1692,Text,

1,0,1,999999,1128,1888,1628,2248,Table Row,

2,0,1,999999,2420,1964,3980,2220,Table Column,

3,0,1,999999,4204,2016,5268,2208, Table Column,

4,8,1,999999,7200,1964,8584,2208,Figure,Graphic of Ace Inc. Company Logo

5,8,1,999999,9564,1988,11252,2208,Figure,Sales figures for 2004

6,0,1,999999,1144,2420,2032,2880, Table Row,

7,0,1,999999,2408,2456,4108,3640, Table Column,

8,0,1,999999,4176,2432,6900,6484, Table Column,
9,0,1,999999,7164,2368,9368,5460, Text,
10,0,1,999999,9552,2420,11632,3120, Text,

The first zone of a new row of a table should be designated with a Table Row. Every subsequent column of that row should be designated with a Table Column.

Table rows and columns can be identified if they have a single logic level (single column under row headings). Tables that have multiple logical levels of row or column headers may be identified as a single logic level or as a graphic with alternate text.

String Option: PDF Retain

Function:

Retains several fields and settings when reading PDF files and generating PDF output.

Timing:

Users: This can be placed in either “Pre Recognition” or “Post Recognition” string option areas.

Developers: This call can be made anytime before `pr_convert_output_to_file()`. This call is “sticky”, it only needs to be called once.

Input Variables:

The command string must include the following parameters separated by commas:

- Name of the command (must be “PDF_RETAIN”)
- Retain bookmarks (0 or 1)
 - 0=do not retain bookmarks from PDF input file
 - 1=retain bookmarks from PDF input file and copy the bookmarks to the PDF output file
- Retain document description fields (0 or 1)
 - 0=do not retain document description fields from PDF input file
 - 1=retain document description fields from PDF input file and copy them to the PDF output file
- Retain initial view (0 or 1)
 - 0=do not retain initial view mode from PDF input file
 - 1=retain initial view mode from PDF input file and copy it to the PDF output file
- Retain hyperlinks (0 or 1)
 - 0=do not retain hyperlinks from PDF input file
 - 1=retain hyperlinks from PDF input file and copy those same links to the PDF output file

Return Value:

No return values.

Example Usage:**User:**

```
"PDF_RETAIN,1,0,1,1"
```

Developer:

```
status=pr_string_options("PDF_RETAIN,1,0,1,1", NULL);
```

In this example bookmarks will be copied from the original PDF input file over to the newly created PDF output file. The PDF document descriptions (Title, Author, Subject and Keywords) will not be copied over to the new PDF output file. The PDF output file

will have the same initial view mode as the PDF input file. Any hyperlinks in the input file will be copied over to the output PDF file.

NOTES:

-Only Title, Author, Subject and Keywords are extracted and retained from the document descriptions at this time.

- Initial page view will be retained if the input PDF file initial page view is one of the following: 1) bookmarks panel and page, 2) pages panel (thumbnails view) and page, open in full screen mode.

- If the PDF/A setting in PDF_DEFAULTS is not zero , and any of the PDF retain settings are set to 1, the engine will automatically change the PDF retain settings to zero UNLESS optimize is also set to 1 in PDF_DEFAULTS.

-If the PDF file has security set for “copying/modifying” contents, then bookmarks/hyperlinks can not be copied over to the new file.

-We can not determine if hyperlinks are set to be not visible, we set all hyperlinks to be visible.

-We can not copy hyperlinks that are “named destinations”. This style of hyperlink is most often seen in PDF Normal files that have been created with authoring tools. These types of files are not normally sent for OCR so this should not be a large concern.

-If you require other attributes of the PDF input file to be retained during processing contact Prime Recognition with your requirements.

String Option: PDF Security

Function:

Allows the creation of passwords for PDF file access and/or a password to override/modify disabled features. Also allows disabling of features such as printing, editing, etc.

Timing:

Users: This can be placed in either “Pre Recognition” or “Post Recognition” string option areas.

Developers: This call can be made anytime before `pr_convert_output_to_file()`. This call is “sticky”, it only needs to be called once.

Input Variables:

The command string must include the following parameters separated by commas:

- Name of the command (must be “PDF_SECURITY”)
- open password (optional)
 - if a string is present, then this string/password must be used by user to open/view this PDF file
- permissions password (optional)
 - if a string is present, then this password can be used by the user to modify or bypass the restrictions (noted below) in the PDF file
- restrictions/features to be disabled (optional)(sum of the relevant numbers below)
 - 1 disables printing
 - 2 disables copying text and graphics from the document
 - 4 disables editing the document
 - 8 disables adding annotations
 - 16 disables filling form fields
 - 32 disables copying for use with accessibility features
 - 64 disables assembly of the document
 - 128 disable high resolution printing

Return Value:

No return values.

Example Usage:

User:

```
"PDF_SECURITY,openpw,permissionspw,5"
```

Developer:

```
status=pr_string_options("PDF_SECURITY,openpw,permissionspw,5", NULL);
```

In this example “openpw” is the password that will be required to open the PDF file. “permissionspw” is the password required to bypass or modify the restrictions placed on the file, and the restrictions on the file include (5=1+4):

- 1 disables printing
- 4 disables editing the document

Some other example strings are:

```
"PDF_SECURITY,openpw
```

In this example, only a password to open the PDF file is created.

```
"PDF_SECURITY,,permissionspw,5
```

In this example, only a password to bypass/modify the restrictions is created. Restrictions are placed on certain actions to the file.

```
"PDF_SECURITY,,,5
```

In this example, no passwords are created, but restrictions are created. (Note that the user will not be able to bypass or modify these restrictions because no password was supplied to do so. (Note the use of commas, with no intervening characters, to denote the lack of entry in these optional fields. Do not use any space characters in any of these entries as this will be confusing to the user.).

NOTES:

- Only Title, Author, Subject and Keywords are extracted and retained from the document descriptions at this time.
- Initial page view will be retained if the input PDF file initial page view is one of the following: 1) bookmarks panel and page, 2) pages panel (thumbnails view) and page, open in full screen mode.
- If you require other attributes of the PDF input file to be retained during processing contact Prime Recognition with your requirements.
- We can not retain information from (or even read) PDF files that are encrypted.
- We can not copy hyperlinks that are “named destination” style. Named destination style is often used in PDF Normal type files via automated tools. Since PDF Normal files are not usually put through OCR, this limitation should not be major problem.
- We can not detect whether a hyperlink border is invisible, so we always display hyperlinks as visible.

String Option: PDA Defaults

Function:

Changes some of the settings used to report PDA file output type.

Timing:

Users: This can be placed in either “Pre Recognition” or “Post Recognition” string option areas.

Developers: This call can be made anytime before `pr_convert_output_to_file()`.

This call is “sticky”, it only needs to be called once. (Thus once a setting is “turned” on it must be turned off with another call.)

Input Variables:

The command string must include the following parameters separated by commas:

- Name of the command (must be “PDA_DEFAULTS”)
- Presence of Word bounding boxes in output

Return Value:

No return values.

Example Usage:**User:**

```
"PDA_DEFAULTS,1"
```

Developer:

```
status=pr_string_options("PDA_DEFAULTS,1", NULL);
```

In this example bounding boxes for words will be included in output. (Bounding boxes for characters are always included).

Notes:

This string option allows PDA output to be sent to the InputAccel Indexing module, so that when the user selects a word in image display area, that word will be highlighted and the word text sent to the index field.

String Option: Word Data in PRO Output

Function:

Adds WORD based data to “in memory” and PRO file output.

Timing:

Users: This must be placed in “Post Recognition” string option areas.

Developers: This call can be made anytime after `pr_recognize_image()` and before `pr_convert_output_to_file()`. However, we recommend that developers call `pr_configure_output()` directly (see Chapter 10b).

This call is not “sticky”, it needs to be called for every page.

Input Variables:

The command string must include the following parameters separated by commas:

- Name of the command (must be “pr_configure_output”)
- Flag to cause addition of WORD data to output (“1”)

Return Value:

No return values.

Example Usage:

User:

```
"pr_configure_output,1"
```

Developer:

```
status=pr_string_options("pr_configure_output,1", NULL);
```

Notes:

This string option allows Word based data to be added to the in memory data structure and to PRO file based output. The “in memory” data structure changes are described in Chapter 10b. The impact on PRO file based output is as follows:

“Format” on line 1 of output will be “1” instead of “0”.

After the standard output for each line of text, 7 lines of data are added:

- # of words on line of text
- Character positions (zero based) that start each word (separated by commas)
(e.g. Word 2 starts at character position 13)
- Character positions (zero based) that end each word (separated by commas)
(e.g. Word 2 ends at character position 22)
- Left edge in BMUs of each word (separated by commas)
- Top edge in BMUs of each word (separated by commas)

- Right edge in BMUs of each word (separated by commas)
- Bottom edge in BMUs of each word (separated by commas)

String Option: Point Size Change

Function:

Changes the reported point size of the OCR output in memory (and can therefore impact file based outputs). Particularly useful when outputting to RTF format, where a 10-20% size reduction can sometimes help maintain the original look and feel of the document. (But also see Output Point Size Change).

Timing:

Users: This is a “Post Recognition” string option.

Developers: This call can be made anytime after `pr_recognize_image()` and before `pr_convert_output_to_file()`. This call is not “sticky”, it must be called for every image.

Input Variables:

The command string must include the following parameters separated by commas:

- Name of the command (must be “point_size_change”)
- Zone number in which to perform calculation (-1 causes whole page to be processed)
- Point size change

Return Value:

No return values.

Example Usage:

User:

```
"point_size_change,-1, -20"
```

Developer:

```
status=pr_string_options("point_size_change,-1, -20", NULL);
```

In this example the point sizes reported for the all text is reduced by 20%.

Notes:

The point size reduction value can be negative or positive.

String Option: Output Point Size Change

Function:

Changes the reported point size of selected OCR output file based formats (only RTF as of this writing). Particularly useful when outputting to RTF format, where a 10-20% size reduction can sometimes help maintain the original look and feel of the document. (But also see Point Size Change).

Timing:

Users: This is a “Pre Recognition” string option.

Developers: This call can be made anytime before `pr_convert_output_to_file()`. This call is not “sticky”, it must be called for each image.

Input Variables:

The command string must include the following parameters separated by commas:

- Name of the command (must be “output_point_size_change”)
- Point size change

Return Value:

No return values.

Example Usage:

User:

```
"output_point_size_change,-20"
```

Developer:

```
status=pr_string_options("output_point_size_change,-20", NULL);
```

In this example the point sizes reported for the all text is reduced by 20%.

Notes:

The point size reduction value can be negative or positive.

String Option: UTF Output

Function:

Changes the supported output file format to either UTF-8 or UTF-16 (or removes Unicode support).

Timing:

Users: This can be placed in either “Pre Recognition” or “Post Recognition” string option areas.

Developers: This call can be made anytime before `pr_convert_output_to_file()`. This call is “sticky”, it only needs to be called once.

Input Variables:

The command string must include the following parameters separated by commas:

- Name of the command (must be “UTF_OUTPUT”)
- Format of output: 0=turn off Unicode, 1=UTF-8, 2=UTF-16

Return Value:

No return values.

Example Usage:

User:

```
"UTF_OUTPUT,2"
```

Developer:

```
status=pr_string_options("UTF_OUTPUT,2", NULL);
```

In this example the format of any output that supports UTF will be UTF-16.

Notes:

The default is for Unicode to be set to 0.

This setting currently only affects ASCII, FASCII (and variants of each), and some custom formats.

Asian and Russian language recognition will automatically generate UTF-16 in RTF (and internally within PDF).

If you would like modifications to this functionality please contact us.

String Option: File Output (second or third file output)

Function:

Creates a second (and optionally third) file based output from `pr_convert_output_to_file()`.

Timing:

Users: This can be a “Pre Recognition” or “Post Recognition” string option.

Developers: This call can be made anytime before `pr_convert_output_to_file()`. This call is “sticky”, it can be called once and then is active until reversed.

Input Variables:

The command string must include the following parameters separated by commas:

- Name of the command (must be “file_output2” or file_output3)
- Value of output format, found in `prstrct.h`, including: 0=ASCII, 1= Formatted ASCII, 3=PRO, 9=RTF, 11=PDF Normal, 12=PDF Image Only, 13=PDF Image + Text, 16=HTML. Or -1 to turn off.

Return Value:

No return values.

Example Usage:

User:

```
"file_output2,13"
```

Developer:

```
status=pr_string_options("file_output2,13", NULL);
```

In this example the output will be PDF Image + Text.

Notes:

A second file will be output. This file will be placed in the same directory, with the same file name, as the standard file except it will have a different extension (extension is based on the output format, for example ASCII=.txt, or PDF=.pdf). Obviously the file output format should be different from the standard (or primary) file output.



Warning

If you have autozoning turned on with image zones, and PDF is one of your formats, make sure PDF is your primary output.

To get a third file output use two string option commands, one of them “file_output2,x” and the other “file_output3,y”.

To turn off this feature, enter -1 as the file format value.

String Option: Subscript/Superscript Recognition Enhancement

Function:

Improves recognition of Subscript and superscript characters.

Timing:

Users: This must be placed in “Pre Recognition” string option area.

Developers: This call can be made anytime before `pr_recognize_image()`. This call is “sticky”, it only needs to be called once.

Input Variables:

The command string must include the following parameters separated by commas:

- Name of the command (must be “sub_super_enhancement”)
- On (1) or Off (0) value

Return Value:

No return values.

Example Usage:

User:

```
"sub_super_enhancement,1"
```

Developer:

```
status=pr_string_options("sub_super_enhancement,1", NULL);
```

In this example the setting of 1 will result in somewhat improved recognition of subscript and superscript characters.

Notes:

This functionality is off by default since many documents do not include subscript and superscript characters, and therefore do not require the processing time (albeit fairly small) required by this functionality.

Recognition of these very small characters is difficult, this function will only improve that recognition by 30-40%. (This function only improves engines 1, 4, and 6. Engines 2, 3, 5 do report sub/superscript characters but are not improved by this option).

String Option: Subscript/Superscript Tagging

Function:

Inserts tag around Subscript and superscript characters.

Timing:

Users: This must be placed in “Post Recognition” string option area.

Developers: This call can be made anytime after `pr_recognize_image()` and `pr_del_results()`. This call is not “sticky”, it must be called for every image.

Input Variables:

The command string must include the following parameters separated by commas:

- Name of the command (must be “sub_super_tag”)
- String to place before subscript characters
- String to place after subscript characters
- String to place before superscript characters
- String to place after superscript characters
- Zone number in which to perform replacement (-1 causes whole page to be processed)

Return Value:

No return values.

Example Usage:

User:

```
"sub_super_tag,^s,^e,^S,^E,-1
```

Developer:

```
status=pr_string_options("sub_super_tag, ,^s,^e, ^S,^E,-1", NULL);
```

In this example “^s” will be placed before the first character in a subscript sequence, and “^e” will be placed after the sequence, “^S” will be placed before the first character in a superscript sequence, and “^E” will be placed after the sequence.

Notes:

String Option: Strong Auto Rotate Accuracy Improvement

Function:

Allows user to select which and how many engines are used in “strong” autorotate. Since default is Engine 1, any addition of engines will result in higher accuracy (and slower processing).

Timing:

Users: This must be placed in “Pre Recognition” string option area.

Developers: This call can be made anytime before `pr_recognize_image()`. This call is “sticky”, it only needs to be called once.

Input Variables:

The command string must include the following parameters separated by commas:

- Name of the command (must be “auto_rotate_level”)
- A valid engine string, e.g. “9” (Level 1), “99” (Level 2), “90099” (Level 3), “99099” (Level 4), “99999” (Level 5)

Return Value:

No return values.

Example Usage:

User:

```
"auto_rotate_level,99"
```

Developer:

```
status=pr_string_options("auto_rotate_level,99", NULL);
```

In this example the setting of 99 will result in somewhat improved recognition of rotated pages, as now two engines are used to sense the proper orientation.

Notes:

The default value is “9”, so there is no reason to use this function to set a value of “9”, unless you are trying to reset the value back to “9” after changing it to another value.

The vast majority of users should use “99” to improve recognition. Each additional engine added increases processing time linearly, but recognition rates improve only slowly with each new engine added.

String Option: Bates Number Accuracy Improvement

Function:

Will improve recognition accuracy of Bates numbers (numbers often added to legal document images).

Timing:

Users: This must be placed in “Post Recognition” string option area.

Developers: This call can be made anytime after `pr_recognize_image()` and before `pr_del_results()`. This call can be made multiple times for one page.

Input Variables:

The command string must include the following parameters separated by commas:

- Name of the command (must be “batesnumber”)
- Vertical location of Bates Number (1=top of page, 2=bottom of page)
- Horizontal location of Bates Number (1=50% of page on the left, 2=50% of the image width in the center of page, 3=50% of page width on the right)
- Format of number with A indicating Upper case alpha ASCII characters (A, B, C, etc.) and N indicating Numeric characters (0,1,through 9) Any character aside from A and N will be considered a fixed separator (e.g. space character, period character, etc.).

Return Value:

0=successful (Note that this does not mean that the Bates number was OCR'd 100% accurately, merely that it fits the defined format)

12250=could not find the Bates number

12251=Bates number did not match the defined format

Example Usage:**User:**

```
"batesnumber,2,1,AAAAA NNNNNNNN"
```

Developer:

```
status=pr_string_options("batesnumber,2,1,AAAAA NNNNNNNN ", NULL);
```

In this example the Bates number is expected to be on the bottom of the page and to the left. An example number would be “ABCDE 01234567”.

Notes:

Only a single zone is supported at this point. If the number is at the top of the page then the number must be in the OCR output of the first line. If the number is at the bottom of the page then the number must be in the last line. Other data may be on these lines but the Bates number must be separated by at least 7 spaces.

In some applications two or more Bates numbers may be legal for a specific page set. If you are a user we don't support this concept yet. If you are a developer, we strongly recommend you run the page against the longest legal format first. If it fails then try the next longest format, and so on.

We recommend that you do not use more than one space as a separator (more than one space is hard for OCR to accurately recognize). We also do not recommend using separators that are hard for OCR to recognize, for example, underscore | * ^ # @ ! \ < > and other unusual characters.

This string option will improve Bates Number recognition but it will not guarantee 100% accuracy.

Chapter 6

PrimeOCR Output Formats

The PrimeOCR engine can generate output in a variety of file based output formats. For developers the output can also be accessed in RAM. This chapter will first discuss which file based output formats are supported, it will then go into detail on the PRO format (PrimeOCR's native output format). The last portion of the chapter will discuss issues relevant to developers including access to the output in RAM.

File Based Output Formats

PrimeOCR can generate the following file based output formats:

ASCII

Just lines of characters recognized.

Conversion Rules:

- All lines are flush left, no blank lines are reported in output. Two blank lines are inserted between output of different zones.
- Word wrapping can be controlled via Setup\Output\AdvancedSettings\ChangeASCIIDefaults (see help text for more information) or ASCII_DEFAULTS string option (described in *Chapter 5*).

ASCII_NOZONEBUFFER,

ASCII but with no lines separating zones.

FASCII (Formatted ASCII)

Includes lines of characters recognized plus blank lines between lines of text and white space on left hand margin of text, as appropriate.

Conversion Rules:

- This file output tries to mirror the look of the zone of the page. If the whole page was defined as one zone then the output will look like the original page.
- If zones were defined on the page then each zone is reported linearly down the page, separated from the next zone by two inserted blank lines.
- There is no ability in ASCII files to define columns, or zones that appear next to each other in the output so the zones must be reported linearly down the page.
- Word wrapping can be controlled via Setup\Output\AdvancedSettings\ChangeASCIIDefaults (see help text for more information) or ASCII_DEFAULTS string option (described in *Chapter 5*).

FASCIITAG (Developer's Only)

FASCII but with "user tag" placed at top of document.

PDA

A file format that is defined in the WordScan 5.2 Development Kit.

PrimeOCR's PDA format is a Level 2 style. The output includes escape sequences compatible with the following WordScan engine settings:

```
c_s_attributes(CDP_A_XYLOCS|CDP_A_CBBS|CDP_A_POINTS)
c_s_marklevel2(CDP_MARK2,CDP_NONE)
```

PrimeOCR does not support image zones in PDA output.

This function should only be of interest to developers who have already interfaced to the WordScan product and want to use their existing output parsing code for PrimeOCR output.

PRO (Prime Recognition Output)

This output is described in detail later in this chapter.

PRSTAR

Custom version of PRO format. Contact Prime Recognition for more information.

RRI3

Format defined by Recognition Research, Version 3.

ZY3

Formatted ASCII plus tag at bottom of page which includes image file name, date, time as defined by ZyIndex, version 3.

RTF

Complies with version 1.1 format defined by Microsoft and used by many word processing programs. Supports multiple pages per file and linked image zones.

Conversion Rules:

- If zones have been defined each zone is created as a RTF frame. Text found within zone is placed within the frame.
- If image zones have been created, the path to the image file is inserted into the field. Note that the image file is not embedded into the RTF file, it is only a link to the image file created by PrimeOCR. Also note that image zones will only display correctly in Word 7.0+. Image zones are TIFF format if original image was bitonal, and JPEG if original image was color, grey scale, some other non TIFF bitonal format.
- Text is placed on the page (no zones defined) or within the frames with the margins and vertical spacing preserved.
- Three or more consecutive spaces in the output are converted to a tab.
- Text can be reported in two basic ways: (1) "As Is", no attempts are made to analyze and convert individual lines into paragraphs, or to remove hyphenation, or (2) "Wrap" text by removing carriage returns at end of lines (and hyphenation) to create paragraphs.
- A number of RTF parameters can be controlled via Setup\Output\AdvancedSettings\ChangeRTFDefaults (see help text

for more information) or RTF_DEFAULTS string option (described in *Chapter 5*).

PDFTO

PDF – Text (“PDF Normal”). Includes text with image zones, if any defined. As an option, images of low confidence words may be included.

PDFIO

PDF – Image Only. Includes only a copy of image. This format will probably not be used with the PrimeOCR engine, as the OCR output has no place in PDFIO format (use PrimeZone to generate PDF – Image Only format).

PDFIT

PDF – Image plus Text. Includes a copy of image with OCR text behind it. Text can be searched with common third party utilities.

COMMA

Comma delimited output. The output of each zone is separated by a comma. This format is useful for zoned images when the output must be imported into a database, or spreadsheet. Please contact us if you require a different delimiter.

HTML

Generate pages to display output on the web.

Conversion Rules:

- If the whole page was defined as one zone then the output will look like the original page.
- If zones were defined on the page then each zone is reported linearly down the page, separated from the next zone by two inserted blank lines. With this version of HTML output, there is no ability to define columns, or zones that appear next to each other in the output so the zones must be reported linearly down the page.
- Image zones are TIFF format if original image was bitonal, and JPEG if original image was color, grey scale, some other non TIFF bitonal format.
- A number of HTML parameters can be controlled via HTML_DEFAULTS string option (described in *Chapter 5*).

XML_WORDCOORD

Fairly simple XML output which only includes OCR’d words and their location coordinates. Units are in pixels. The character encoding is UTF-8.

XML_PRO

XML output which includes the same data that is the PRO format. The character encoding is UTF-8.

(Prime Recognition has developed many custom XML formats for its customers, if you need a different type of XML format or with different contents please contact us.)

PDF Output

PrimeOCR allows the control of many PDF output features.

Feature

Control/Documentation

Optimize (byte serve)

Desample image

Generate Thumbnails

Accessibility (508 compliance)

PDF/A (1a or 1b)

JBIG2 compression

Zip compression

Snippets in PDF Normal

Job Server: Setup/Output/Output

Settings/PDF/Details

String Option: PDF_DEFAULTS

Retain Input PDF information

-Bookmarks

-Document descriptions

-Initial view

-Hyperlinks

Job Server: Setup/Output/Output

Settings/PDF/Details

String Option: PDF_RETAIN

Bookmarks added manually or automatically

...\\(install directory)\\bin\\bookmark.ini (see file
bookmark.ini.example for documentation)

Add document descriptions (including Creator,
Producer, Title, Subject, Author, Keywords)

...\\(install directory)\\bin\\docprop_pdf.ini (see
file docprop_pdf.ini.example for
documentation)

Security/restrictions/passwords

String Option: PDF_SECURITY

PRO Output

PrimeOCR's native format is called "PRO". It reports all output that is available from the PrimeOCR in an "easy to read" format. We will first start the description of this format by defining the key concepts and variables used in the PRO format.

Definitions

Coordinate Units

PRO output coordinate measurements are always in units of 1/1200th of an inch, measured from point 0,0 which is the top left hand corner of the image.

Line Baseline Coordinates

A "baseline" is a line that extends through the bottom edge of all characters of a line (ignoring descenders on characters such as "p").



There are four coordinates to a baseline. The first two coordinates represent the x,y point of the left edge of the line on the original image. The second two coordinates represent the x,y point of the right edge of the line. (Note that y2 will often not match y1 because of skew in the image).

PrimeOCR extends the definition of a "baseline" to also include the height of the line via a fifth value.



Note

If a page is run in LEVEL_1 accuracy mode on all zones then $y1=y2=y$ coordinate of midpoint of line. Therefore, on documents with large skew y1 and y2 will be relatively inaccurate.

Recognized Characters

See Appendix A for a listing of characters that may be recognized by the PrimeOCR engine.

Confidence Levels

Confidence levels range from 1-9, with 1 being very low confidence. As confidence levels increase the percentage of recognition errors decrease.

Character Format Data

This single byte (i.e. char in C language) is a bit field for character attributes.

PR_CHAR_NORMAL	0x01 (proportional width, serif font)
PR_CHAR_BOLD	0x02
PR_CHAR_UNDERLINE	0x04
PR_CHAR_ITALIC	0x08
PR_CHAR_SUPERSCRIPT	0x10
PR_CHAR_SUBSCRIPT	0x20
PR_CHAR_FIXED	0x40 (Fixed width as opposed to proportional width)
PR_CHAR_SANS_SERIF	0x80 (sans serif as opposed to serif)

The NORMAL attribute is included in every character. (A NORMAL character by default is proportional width and a SERIF font.) Therefore a character that is bold and underline would be value 7. That is, $7 = 1 + 2 + 4$.

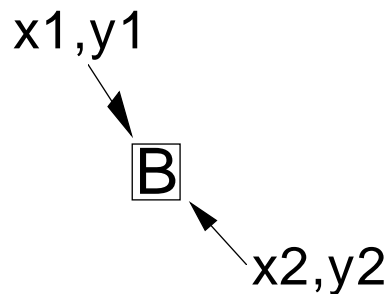
Note that character attributes are not very accurate as reported by conventional OCR engines. The most common mistake is to not report an attribute. PrimeOCR does improve on conventional OCR performance in this area, however, be warned that even PrimeOCR's output will not be 100% accurate.

Character Size

The size of the character is reported in points (1/72 of an inch). Again conventional OCR engines do not excel at reporting precise character sizes. It is best to think of these values as relative to each other vs. an absolute measure of the size of the character. The most common conventional OCR mistake is to report too large a size on characters larger than 20 points in size.

Character Bounding Box

A bounding box is a rectangle on the original image that encloses the character. The first two entries (0 and 1) in the bounding box array represent x_1, y_1 , the top left corner of the rectangle. The second two entries (2 and 3) represent x_2, y_2 , the bottom right hand corner of the rectangle.





Note

First, the good news. PrimeOCR's bounding boxes are more accurate than conventional OCR character bounding boxes, indeed many OCR engines do not give character bounding boxes because it is difficult to generate them accurately.

The bad news is that PrimeOCR's bounding boxes may not always totally enclose the character. For human error correction we recommend that you display an image that is centered on the PrimeOCR coordinates but is 1.5 times the average line height, and at least 2-3 times the reported width of the character. PrimeOCR does include bounding boxes for space characters although most applications have no need for this information. Bounding boxes for space characters in low quality documents will usually be in the correct general location but the width (difference between x1 and x2) is often incorrect.

PRO Output Hierarchy/Content

There is a five level hierarchy in PrimeOCR output. The first level is the document, the second level is the page, the third level is the zone, the fourth level is the line, and the fifth level is the character.

Document Level

If multiple pages exist in a document defined by the user/programmer then multiple pages will be appended together into one output document. Note that the number of pages in the document is not reported in the PRO output, you can only get this number by reading in, maintaining a count, of all the pages in the PRO output.

Page Level

The image height, width, resolution, color/binary type, and page based OCR configuration items are reported at this level including the number of zones defined.

Zone Level

The zone OCR configuration data is reported along with the number of lines in this zone.

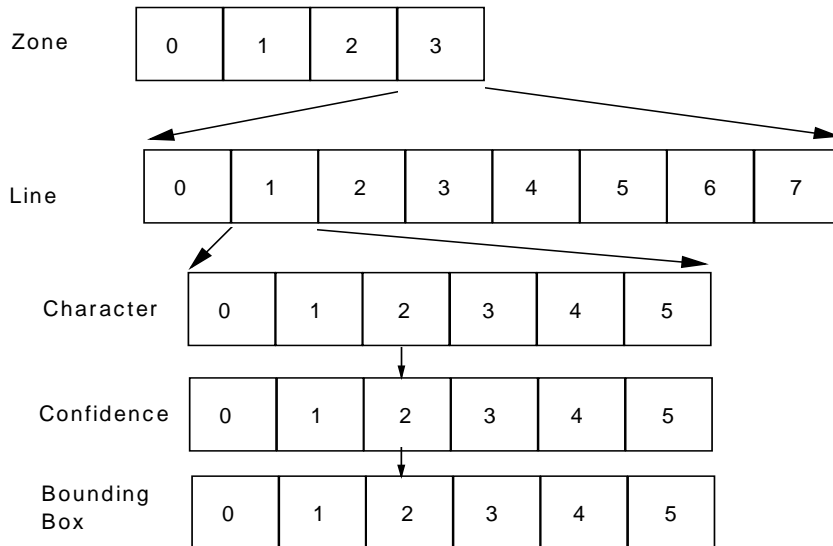
Line Level

Each line is described by a baseline plus average height for the line.

Character Level

Recognized text in this line. Each recognized character has a confidence level, character attribute data, point size, and a bounding box.

Text, Confidence, and Bounding Box of 3rd character of 2nd line of 4th Zone



File Based Output

Typically a PRO format output file will have a “.PRO” extension, especially if it has been output from the PrimeOCR Job Server, however, this is not a requirement.

The first 5 lines in a PRO file are used to describe the configuration settings for the first image or page within the document. The last line of this group describes the number of zones on the image.

The next line in the output indicates the beginning of zone descriptions. Then each zone's configuration data and output is fully described.

Within a zone description the first line is the configuration data. The next line is the number of lines recognized in this zone. Each recognized line is then described using 9 lines of text:

- The baseline coordinates of the line plus average height
- The recognized text
- Confidence levels for each character of recognized text
- Character attributes for each character of recognized text
- Point size for each character of recognized text (entries separated by a comma)
- x1 of the bounding box for each character (entries separated by a comma)
- y1 of the bounding box for each character (entries separated by a comma)
- x2 of the bounding box for each character (entries separated by a comma)
- y2 of the bounding box for each character (entries separated by a comma)

There are no delimiters to note a transition from one line of recognized text to the next or one zone to the next, however there is no need because the format described above is fixed, e.g., each line always is described by 9 lines.

The maximum memory requirements of each line are noted below.

30 characters (5 coordinates *(5 characters/coordinate + 1 character delimiter))
 500 characters (up to 500 characters recognized per line)
 500 characters
 500 characters
 2000 characters (up to 500 entries * (3 characters/coordinate + 1 character delimiter))
 3000 characters (up to 500 entries * (5 characters/coordinate + 1 character delimiter))
 3000 characters (up to 500 entries * (5 characters/coordinate + 1 character delimiter))
 3000 characters (up to 500 entries * (5 characters/coordinate + 1 character delimiter))
 3000 characters (up to 500 entries * (5 characters/coordinate + 1 character delimiter))

OMR and Image zone output differ from text zones and are described below via examples.

At the end of each page description there is a "\f" character (end of page).

Example

The first several lines in a PrimeOCR output file could look like this:

```
450,0,850,22908      Version, Format, Confidence
User Tag info goes on this line.
c:\images\4931.tif   2       0
0,1,1,10,0,200,0,0,0,9984,13428,0,0  Page settings
3      # of Zones
0,0,1,999909,3716,2564,5012,2760,    Zone Settings
18     # of Lines
6368,1127,8376,1115,124
Escrow #   7 0 3 -2
9999999199941215149
111111111113333333
12,12,12,12,12,12,12,12,12,12,10,10,10,10,10,10,10,10
0,0,0,0,0,0,0,7311,0,0,0,7811,7925,7925,8039,8039,8153,8153,0
0,0,0,0,0,0,0,1552,0,0,0,1665,1665,1596,1665,1665,1665,1644,0
0,0,0,0,0,0,0,7410,0,0,0,7925,8039,8039,8153,8153,8267,8284,0
0,0,0,0,0,0,0,1694,0,0,0,1803,1803,1803,1803,1803,1803,1803,0
```

Line by Line Explanation

"300,0,850,22908 Version, Format, Confidence"

Version of software, flag ("1") if custom word output is included, and average confidence of all characters on this page * 100 (therefore range is from 100-900). The last number in string is for Prime Recognition's internal use only.

"User Tag info goes on this line."

Data, if provided by user_tag string, in call to pr_convert_output_to_file() by a developer.

"c:\images\4931.tif 2 0"

String provided by "image_name" string in call to pr_convert_output_to_file(). Typically this the image name but it could be any arbitrary string. Prime Recognition's OCRJOB Server places the original image name, tab, the page number (e.g., "2" as in second page of a multipage tiff file), tab, indicator if image is black/white (0) or color/greyscale/or other non standard format (1).

```
"0,1,1,10,0,200,0,0,0,9984,13428,0,0 Page Settings
```

(New Document, Page Quality, Print Type, Language, Deskew, DPI, Image Process, Autozone, Training, ImageWidth, ImageHeight, Auto Rotation, Save Image)

Page Configuration data. The numbers represent the integer value of the enums defined in PRSTRCT.H where appropriate. For example, in this image the Document Source is SAME_DOCUMENT (0), Page Quality is LOW_QUALITY (1), Print Type is DOT_MATRIX(1), Language is US (10), Deskew is not performed (0), DPI is 200 dpi, no image processing, autozoning, or training took place (0,0,0), the image width was 9984 BMUs, the image height was 13428 BMUs, no auto rotation took place, nor was image processed image saved.

```
"3 # of Zones"
```

Page Configuration data. Number of zones on this page.

```
"0,0,1,999909,3716,2564,5012,2760, Zone Settings
```

(Zone #, Restrictions, Lexical Check, Acc Level, Zone bbox)

Zone Configuration data. First line in a repeating structure for each zone. The numbers represent the integer value of the enums defined in PRSTRCT.H where appropriate. For example, in zone 0 (0), the Restrictions is NO_RESTRICTIONS (0), Lexical check is NO_LEXICAL(0), Accuracy Level is 999909 (run all engines except TextBridge with weight "9"), and the zone bounding box coordinates in 1/1200ths of an inch are:

top right hand corner:

x - 3716

y - 2564

bottom left hand corner:

x- 5012

y - 2760

```
"18 # of Lines"
```

Zone Configuration data. Second line in a repeating structure for each zone.

```
"6368,1127,8376,1115,124"
```

Line Configuration data. First line in a repeating structure for each line.

Line baseline coordinates plus the average height of the line.

```
"Escrow # 7 0 3 -2"
```

RECOGNIZED TEXT OF LINE. Second line in a repeating structure for each line.

All whitespace has been converted to spaces. There are no tabs, or other special spacing characters in PrimeOCR ASCII text output.

```
"9999999199941215149"
```

Confidence level of each character recognized. Third line in a repeating structure for each line. The confidence level range is from 1-9. 1 is very low confidence, 9 is very high confidence. (Prime Recognition's testing has shown a sharp break in error rates at the confidence level 6, i.e., for most applications confidence levels 7-9 should be treated as accurate.)). The actual binary value in the PRO file is 48 higher than the value in RAM based PRO output. For example, "1" in RAM output is printed as "49" in file based output. This is because "49" is displayed as "1" when reading the file in NotePad, WordPad, or other editors.

```
"11111111111133333333"
```

Character attribute of each character recognized. Fourth line in a repeating structure for each line. This example shows normal characters (1), going to bold characters (3=1+2). The actual binary value in the PRO file is 48 higher than the value in RAM based PRO output. For example, "1" in RAM output is printed as "49" in file based output. This is because "49" is displayed as "1" when reading the file in NotePad, WordPad, or other editors. This can be a little confusing once the attribute value is greater than 9, so the key to remember is to take the binary value of the attribute printed in the file and subtract 48 to get the real value.

```
"12,12,12,12,12,12,12,12,12,12,10,10,10,10,10,10,10,10"
```

Size of each character recognized expressed in points (1/72 of an inch). Fifth line in a repeating structure for each line. This example shows point size 12 characters, going to point size 10 characters.

```
"0,0,0,0,0,0,0,7311,0,0,0,7811,7925,7925,8039,8039,8153,8153,0"
```

```
"0,0,0,0,0,0,0,1552,0,0,0,1665,1665,1596,1665,1665,1665,1644,0"
```

```
"0,0,0,0,0,0,0,7410,0,0,0,7925,8039,8039,8153,8153,8267,8284,0"
```

```
"0,0,0,0,0,0,0,1694,0,0,0,1803,1803,1803,1803,1803,1803,1803,0"
```

The bounding box of each character. 6th-9th lines in a repeating structure for each line.

Only "suspicious" (all characters is the initial default) characters generate a real bounding box, all other characters have a bounding box of 0,0,0,0. For example, the "#" character has a confidence level of 1, so the following bounding box is reported:

```
x1      7311
y1      1552
x2      7410
y2      1694
```

Bounding boxes are in units of 1/1200 of an inch, measured from the top, left corner (0,0) of the image supplied to PrimeOCR. x1,y1 defines the top left corner of the bounding box, and x2, y2 defines the bottom, right corner of the box. Each number in a row is separated from its neighbor by a comma.

The nine "line description" lines are repeated for the number of lines in this zone (18 in this example). Then the description of zone 1 begins, etc. until all zone descriptions are complete.

OMR Zones

An OMR zone reports a single line of zone configuration data and one line of zone output, for example:

```
"0,7,1,3,3716,2564,5012,2760,      Zone #, Restrictions, Lexical Check, Acc Level, Zone bbox"
```

(Note the restriction level "7". The lexical check and Accuracy Level data is irrelevant to an OMR zone and should be ignored.)

"75"

(Percentage of OMR zone that was black (number between 0-100).)

Image Zones

An image zone reports a single line of zone configuration data and one line that includes the image zone file name, for example:

```
"0,8,1,3,3716,2564,5012,2760,    Zone #, Restrictions, Lexical Check,  Acc Level, Zone bbox"  
"test1.00010003"
```

Note the restriction level "8". The lexical check and Accuracy Level data is irrelevant to an image zone and should be ignored.

No other data is reported for an image zone. However, a Group 4 TIFF file (JPEG if original images was color/greyscale/nonstandard format) is created in the output directory with the name "XXXXXXXX.YYYYZZZZ" where "XXXXXXXX" is the name of the original image file and "YYYY" is the page number and "ZZZZ" is the zone number on that page (Zone numbers start at 0). In base 36 each character can vary from 0-9 and then from A-Z (note that due to NT/WIN95's file system, letters are not case sensitive, therefore "a" is equivalent to "A"). For example, an image zone on the 12th page of the document called "TEST.TIF", that has a zone number of 2, will be named "TEST.00120002". The "0012" refers to the 12th page in the document, and "0002" refers to the 3rd zone on that page. (The first zone is 0, the second zone is 1, the third zone is 2, etc.) An image zone on page 37 with a zone number of 15 would have an extension of ".00370015".

Note that if you move the PRO file, you must move the image zones to the same new directory (unless you plan on ignoring the image zones).

(Developers: refer to the documentation for the `pr_configure_image_zone()` call and `pr_convert_output_to_file()` call for more details on the creation of the image files.)

RAM based PRO Output (Developers Only)

The `pr_recognize_image()` call returns a pointer to an array of a PrimeOCR defined datastructure, "zonestructure". The number of zonestructures in the array is equal to the number of zones defined by the developer's control program. This number must be remembered by the developer's control program, it is not reported by the PrimeOCR RAM output.

Note that the Document and Page level output data is not present in RAM based output. This data is already supplied to the developer or by the developer. RAM based output is based on one image, not a document or multiple page construct.

zonestructure Data Structure

The zonestructure data structure includes the following elements:

Short source	This is used by PrimeOCR internally. There is no significance to this attribute for the end user.
Short numlines	Number of lines in the zone.
Short confcount	% of zone that is black for OMR zones.
Linestruct * lines	A pointer to an array of pointers to line descriptions. There are numlines number of elements in this array.

linestruct Data Structure

The linestruct data structure includes the following elements:

Short source	This is used by PrimeOCR internally. There is no significance to this attribute for the end user.
Short linecoord[5]	x1,y1, x2,y2, height of line. This is not a bounding box. It is the coordinates of the baseline of the line. The last coordinate is the average height of the line.
Short num_chars	Number of characters in this line.
Char * character	Pointer to a string of characters that represent the reported line including conversion of any reported whitespace to spaces.
Char * confidence	<p>Pointer to a string of confidence values. Confidence values correlate precisely to character array by position, i.e. confidence[33] is the confidence level of character[33]. Confidence is reported on a 1-9 scale.</p> <p>1 - very low confidence 9 - very high confidence</p> <p>The distribution of characters by confidence level depends, of course, on the quality and other factors of any specific document.</p>

Char * char_data

Pointer to a string of character attribute information. These values correlate precisely to character array by position, i.e. char_data[33] is the attribute data of character[33]. The value is a sum of Normal =1, Bold=2, Underline=4, Italic=8, Superscript=16, Subscript=32, Fixed=64, Proportional Sans Serif=128. Normal is included in every character. (Normal is defined as a Proportional font with Serif style.)

For example, a normal character would be value 1. A bold, underline character would be value 7 (1+2+4).

Note that if Proportional Sans Serif attribute is turned on the resultant value is >128. This means that RAM based values of char_data will be interpreted as negative by a debugger, or other tools. This is not an error, the underlying bit pattern is accurate.

Char * point_size

Pointer to a string of character size information. These values correlate precisely to character array by position, i.e. point_size[33] is the size of character[33] in points (1/72 of an inch).

Bbox * bbox

Pointer to array of bbox datastructures which represent a bounding box for a character. Only characters that have a confidence level of equal to or below ConfidenceBbox level set in PRIMEOCR.INI will have a non zero bbox.

99% of the time at least a portion of the target character is in the x & y dimensions of the bbox. 99% of the time the character is fully enclosed by adding 50% of the bbox width and height to either side. For editing purposes the relevant image location should be centered on the bbox location, use the bbox y dimensions + 50% on top and bottom, but show at least 200% of the bbox width, and preferably > 1000% of the bbox width.

bbox Data Structure

The bbox data structure includes the following elements:

Short xy[4]	Typically used to describe a bounding box, a rectangle that encloses an area on the original image (e.g. zones or characters)
	Array elements defined as: 0,1 - the upper left hand corner 2,3 - the lower right hand corner

Example

A simple example will illustrate the ease of accessing PrimeOCR output in RAM.

After a series of initialization and configuration calls the developer will recognize the image with the call shown below:

```
short status;
zonestructure * pr_output;
pr_output=pr_recognize_image(&status);
```

The following variables will access all relevant PrimeOCR output:

pr_output[i]	The [i]th zone datastructure defined on this image. i can range from 0 to the number of zones-1 defined on this image. The number of zones defined by the developer for this image must be remembered by the developer's control program, it is not reported by PrimeOCR output.
pr_output[i].numlines	The number of lines in zone[i].
pr_output[i].lines[x1]	The [x1] line datastructure in zone[i]. x1 can range from 0 to pr_output[i].numlines-1.
pr_output[i].lines[x1].linecoord[x2]	The baseline coordinate[x2] of line[x1] in zone[i]. linecoord[4] is the average height of line. x2 can range from 0 to 4.
pr_output[i].lines[x1].character[x3]	The [x3] character of line[x1] in zone[i]. x3 can range from 0 to the number of recognized characters (see below) on this line-1.
pr_output[i].lines[x1].num_chars	The number of characters recognized in this line of text.
pr_output[i].lines[x1].confidence[x3]	The confidence value for the [x3] character.

<code>pr_output[i].lines[x1].char_data[x3]</code>	The character format bit field value for the [x3] character.
<code>pr_output[i].lines[x1].point_size[x3]</code>	The size in points for the [x3] character.
<code>pr_output[i].lines[x1].box[x3].xy[x4]</code>	The [x4] bounding box coordinate of the [x3] character of the [x1] line in zone[i]. x4 can range from 0 to 3.



Note

The elements in a C style array start numbering at 0 instead of 1. For example, if `pr_output[i].numlines` reports 18 lines, then `x1` in `pr_output[i].lines[x1]` can range from 0 (the first line) to 17 (the last line). It is a common, and potentially damaging, mistake to try to access the 18th line using `pr_output[i].lines[18]`.

Asian Language Output in PRO file

When the recognized language is an Asian Language, each recognized character is identified by 4 number/letters from the Unicode Standard 5.0, for example, “2E80” would represent 1 character. So if `num_chars` is 10, there will be a total of 40 number/letter characters in the character line. All other lines (`confidence`, `point_size`, etc.) will still have 10 corresponding entries.

Other RAM based Output (Developers Only)

Chapter 10 – Programming to the PrimeOCR API describes other calls (`pr_output_xxx()`) which can access individual elements of the RAM based output. These calls are highly recommend for developers using Visual Basic, or developers not comfortable with the multi-dimensional data structures noted above.

Chapter 7

Troubleshooting

This chapter describes strategies and tools to help you identify problems with PrimeOCR (including the PrimeOCR Job Server). Developer issues are covered at the end of the chapter.

Troubleshooting Suggestions

Check the PROCR.LOG file for any messages. This file is available under Logs\Job Server Log menu item, or Errors button, or look in the PRDEV\BIN directory.

A common problem is more than copy of software on the machine. Search your PC's full set of local hard drives for PROCR32.DLL. If multiple copies exist make sure you are using the correct version or delete all but one copy of software.

If you have installed imaging products from other OCR vendors, or other imaging vendors:
Temporarily remove the software from these other vendors to see if any problems persist.

If the program appears to run correctly but does not write an output file, or the output file is 0 bytes size:

Make sure that enough disk space exists to write PrimeOCR results (if file output is being used). A PrimeOCR output file can easily require 60-70KB of disk space. The PROCR.LOG file will warn you when disk space in the output drive or the log drive is below 5 MB in free space.

If the program runs much of the time but once in a while does not run correctly:

Restart the operating system and in some cases, reboot the system. The majority of these problems are caused by a lack of RAM. A dialog box may also appear describing the error. Please note both the error number and the text in the error dialog box.

If the program has problems on the very first image (depending on your configuration it may continue on to the next image), try the following:

Select "Stop Processing" in the \SETUP\OCR Engine\Error Handling area of the PrimeOCR Job Server.

Select the "Log PrimeOCR engine entries" setting in the \SETUP\OCR ENGINE\LOGGING area. Look at resulting PROCR.LOG file.

If these actions still do not solve the problem please call technical support at Prime Recognition. We may ask you to send us the files necessary to replicate and/or diagnose your problem. These files would include:

- Image file
- Template file (.ptm)
- Job file (.job)
- PRIMEOCR.INI, OCRJOB.INI, PRIMAGE.INI
- PROCR.LOG

You would send these files as email attachments to Prime Recognition's technical support (mail to: support@primerecognition.com).



“Run as admin” (Vista, Windows 2008, Win7)

Depending on your security settings, it may be important for you to run the application that uses the PrimeOCR engine (for example, the OCR Job Server, or your programmed application) with Admin rights, even if you are not logged in with Admin rights. To do this, right click on \bin\PRLICENSE.EXE and select to run as Administrator. If you are not logged in as admin, you will be asked to do so (for this application only).

PrimeOCR Job Server Error Reporting

Any serious errors will be reported to a file called PROCR.LOG in the \BIN directory. Warnings and log entries (if you turned on the Job Server log setting in the Setup\OCR Engine box) will also appear here.

Intermediate Error Reporting

PrimeOCR can automatically handle almost any error from the intermediate processes. Errors handled this way will be logged to a file called "PROCR.LOG". For example, if one intermediate engine fails on an image then PrimeOCR will terminate the engine for that image and continue processing. (The engine will be brought back up for the next image.). The error will be logged in the PROCR.LOG file as a warning to inform the user that this image did not receive processing by all engines. The warning listing includes the engine which reported the trouble, a message, the image which caused the problem, and the date and time. It can be a fairly common occurrence to have an internal engine fail so most users should not worry about these warnings (unless they occur on each image for a specific engine, or, at the other extreme, more often than every 20 images for any specific engine). However, more serious errors are also reported to this file so we encourage you to review its contents.

Activity Log

A text file log of all major activities of the engine can be generated for debugging purposes. Select the “Log PrimeOCR engine entries” setting in the \SETUP\OCR ENGINE\LOGGING area. Look at resulting PROCR.LOG file.

The default name of the log file is PROCR.LOG, and the default size is 1000K bytes. You may change both of these settings with the \SETUP\OCR ENGINE\ADVANCED SETTINGS\LOGGING\Change PrimeOCR Log Attributes or with string option API call "log_settings" (see *Chapter 5 – String Options* for more information).

The log file includes a description of all API calls made, along with the supplied variable values. It also includes major internal activities. We recommend you call Prime Recognition for assistance in reading the log file.

Other Debug Facilities

Several debugging tools are embedded within the PrimeOCR engine and can be activated by making some changes in the PRIMEOCR.INI file. However, these tools are intended for the use of Prime Recognition for remote technical support and are not documented here.

Error Numbers/Messages

PrimeOCR returns an error number, always negative, to indicate what type of error has occurred. The latest list of errors and an explanation of those errors is kept in the PROCRERR.TXT file in the \BIN directory.

The Job Server will translate this error number into a text message that provides an English based explanation of the error and in some cases suggestions as how to fix the error. Developers might note that the Job Server uses the pr_report_error() call to perform this task.

Trouble Shooting (Developers Only)

If your program does not work at all:

Try to compile and run the C source code sample program as is. If the sample program does not compile or run correctly with the included example images then there is a problem with your environment.

If the program works but fails after a limited number of images:

Make sure you are calling pr_del_results() after you are done with the RAM based output, otherwise the operating system will eventually run out of memory which will cause PrimeOCR operating errors.

Are you getting dialog boxes that warn you of GPFs? If so you should change the “ChangeNTErrorsReporting” setting in the PRIMEOCR.INI file to the value of “1”. **(Remember you must be logged in with Administrator privileges).**

API Error Reporting (Developers Only)

The best way to determine the cause of errors is to always check the value of the "status" variable returned by each PrimeOCR API call. A value of 0 indicates success, any other value indicates an error except for the `pr_open()` call which returns the version number as a positive number..

All API calls perform as much input validation as possible, such as receiving a NULL pointer, or input data out of range. Bad inputs will cause immediate function termination (no intended processing occurs) and reporting of bad input.

An API call, `pr_report_error()` is available that will translate the error number into a text based message. This text comes from the `PROCRERR.TXT` file. If so desired a developer may modify the text in this file to better fit their needs (for example, an international developer may wish to translate text to another language.) If an error is reported that is not listed in `PROCRERR.TXT` please call Prime Recognition technical support.

Chapter 8

Contacting Prime Recognition & Warranty/Support

We encourage you to call Prime Recognition with any technical or sales questions or comments. We are very interested in feedback on our products. If you need any functionality which is not in the existing product please let us know.

If you have a significant problem with our product or support, please contact technical support directly at the phone number below. We will do our best to set it right.

Prime Recognition can be contacted by phone, FAX, Email, Web, or FTP as noted below.

Phone: 425-895-0550

FAX: 425-484-6415

Email Address: support@primerecognition.com

FTP Address: <ftp.primerecognition.com>

Web Address: www.primerecognition.com

Warranty/Support

PrimeOCR offers an annual warranty and support program. The program offers free unlimited technical support via phone, FAX, FTP, mail, or Email at the above numbers, plus free maintenance upgrades and 50% discount on major upgrades. A Premium maintenance program is also available that includes free major upgrades.

In order to be covered by our warranty program all licensed products, even those not the subject of the current technical support question or issue, must be under an active warranty program.

On-site Technical support is available for a daily charge plus travel expenses. Please call to verify the current daily rate and availability.

Chapter 9

InputAccel Module

PrimeOCR can be run as an InputAccel module. (InputAccel is a document capture system developed and marketed by EMC).

PrimeOCR and other Prime Recognition products have been certified by EMC to work reliably within InputAccel.

Other Documentation

We strongly recommend you read:

- The PrimeOCR Access Guide
- Documentation for InputAccel

This Chapter does not attempt to define InputAccel terms or methodology.

System Requirements

OS: XP through Windows 7

RAM: 128 MB of free RAM recommended. See *Chapter 2* for RAM requirements of high resolution/color or other extra memory consumption processing.

InputAccel Server: Version: 1.1, 1.2, 2.0, 2.2, 3.0, 4.0, 5.0, 5.2, 5.3, 6.0 or later.

A PrimeOCR license (hardware key or software key based).

Installation

Setup workstation to be able to communicate with IA server (see IA documentation, usually requires editing a HOST and SERVICES file for TCP/IP communications).

Install hardware key or software key license (see *Chapter 3* for more details).

Put Installation Directory\Bin path in PATH statement and reboot PC (otherwise IA Supervisor will not be able to load OCR Setup program). (This can be done by ControlPanel\System\Environment\SystemVariables\Path).

Install software.

InputAccel specific files are installed to PrimeOCR installation \BIN sub directory, including:

MDF Files -----

Each file is self documenting for values that are supported by module.

PROCRIA.MDF	InputAccel Module Definition File for PrimeOCR
PRVRFIA.MDF	InputAccel Module Definition File for PrimeVerify
PRPOSTIA.MDF	InputAccel Module Definition File for PrimePost
PRZONEIA.MDF	InputAccel Module Definition File for PrimeZone

IAP/PCF/IPP Files (not a complete list)-----

PROCR-scan+procr+textexport

Very simple example using PrimeOCR engine, includes SCAN, OCR (PrimeOCR), TextExport. The other modules are standard InputAccel modules. Use NOVERIFY.PTM (described below) as configuration file for PrimeOCR engine with this configuration.

PROCR-scan+procr+verify+textexport

Very simple example using PrimeOCR engine and PrimeVerify, includes SCAN, OCR, Verify, TextExport. Use VERIFY.PTM (described below) as configuration file for PrimeOCR engine with this configuration.

PROCR-scan+procr+pdfexport

Very simple example using PrimeOCR engine, includes SCAN, OCR, PDFExport. The other modules are standard InputAccel modules. Use NOVERPDF.PTM (described below) as configuration file for PrimeOCR engine with this configuration.

Template Files ("PTM")-----

NOVERIFY.PTM	Causes PrimeOCR engine to generate PDA2 output which in turn can be used by many InputAccel modules directly, including Text Export Module.
VERIFY.PTM	Causes PrimeOCR engine to generate PRO output. PRO output is sent to PrimeVerify IA version which in turn can output PDA2 or PDF format output which can be used by many InputAccel modules directly, including Text Export Module/PDF Export Module.
NOVERPDF.PTM	Causes PrimeOCR engine to generate PDF output (default is Image plus hidden text but can be modified to Image Only and Normal formats). PDF output can be used by standard InputAccel PDF Export module.

Configuration

Setup is performed in five areas:

1. IPP/PCF. See self documented MDF file, and example IPP/PCF files.
2. IA Supervisor Setup – (IA Supervisor must be installed/run on same machine to which PrimeOCR is installed.) This allows you to configure which template (a page oriented OCR configuration file, sometimes called a “PTM” file) will be used by PrimeOCR. Setup also allows you to define “string options” that will influence the OCR processing. String options are defined in *Chapter 5*. (If you want to use PrimeOCR with IA Index module see PDA_DEFAULTS string option in *Chapter 5*).
3. Template/PTM File. You can use the Wizard (found in IA Supervisor Setup as a button) to create a template file, or PrimeView (outside of IA), or manually create/edit the template ASCII file as defined in *Chapter 4 – PrimeOCR Job Server*. (You can also use one of the predefined PTM files.)
4. PRIMEOCR.INI/PRLEXICA.INI/PRIMAGE.INI - See *Chapter 3 - Installation*. (Most users can leave as is).
5. Local Setup. Run PROCRIA.EXE. Choose Setup button. See further details below.

Local Setup

CPUs Select smaller of number of CPUs licensed or number of CPUs available for processing. “1” is always a safe choice.

Verbose Logging Detailed entries of each step performed by PrimeOCR will be placed in the ...PROCR.LOG file. This setting is only recommended when trying to debug problems with PrimeOCR.

If an Error Occurs:

Stop processing Will stop processing on this PC only.

Continue processing Processing with new images/pages will continue. The error counter on the main screen will increment, and an error will be logged in the PROCR.LOG file.

Restart PrimeOCR PrimeOCR will be shut down and automatically restarted if an error occurs that is in the Error List below. We ship the product with a default list of errors, you can add/subtract errors from this list (make sure to include negative sign if relevant, and separate entries with a comma). IT IS CRITICAL that you edit PRBACKUP.INI to restart PrimeOCR and with the appropriate logon information (more information is included in PRBACKUP.INI).

Sometimes Windows RAM gets corrupted, especially after a long time of processing, restarting PrimeOCR can often fix this issue. However, in extreme cases you will need to reboot the PC.

Starting Engine

Run PROCRIA.EXE. Select one of the two “START” buttons depending on your batch processing requirements. Statistics will be captured on the screen.

Stopping Engine

Select “STOP” button. Engine will stop when current page is finished processing. You may push “START” button and engine will resume with next image.

Error Reporting

Errors will be reported to a file “PROCRIA.LOG” in the \BIN subdirectory of installation directory.

See PROCRES.PCF for a simple example of how to handle OCR errors at the PCF level. (Similar error logic should be applied to the PrimeVerify PCF logic if applicable.)

Features Not Supported

The current implementation does not support image zones (unless included in PDF Normal format directly output from PrimeOCR).

Chapter 10

Programming to PrimeOCR API

The PrimeOCR interface is a DLL (Dynamic Link Library) with an API. The PrimeOCR DLL, in turn, manages calls to other DLLs to execute its functionality. The DLLs were developed in Microsoft Visual C++ , however, the master DLL which implements the API to the developer was implemented with a "C" style interface which can be accessed by C, C++, Visual Basic, Delphi, and other tools/languages. Source code for four sample applications are included which use C and Visual Basic and CNET/VBNET as the development language. Please refer to these examples for an

illustration of how to implement the items noted below.

Include Files

Any call from a C/C++ program to the PrimeOCR API requires these include files:

```
#include <prstrct.h>
#include <prapi.h>
```

The PRSTRCT.H file details the PrimeOCR enumerated constants and data structures and the PRAPI.H file contains the API call formats.

Libraries

The PROCR32.LIB library must be linked into the controlling application. This library is in the \LIB subdirectory. The PROCR32.LIB file automatically links your program into PROCR32.DLL, the main DLL for PrimeOCR.

Programming Languages

Developers using other languages should refer to their language documentation for instructions on how to call a DLL with a C style function call interface. Visual Basic users are encouraged to read *Chapter 11 – Example Applications* for a discussion of the PRVBEX32.EXE example program which was written in Visual Basic.

API Call Functionality Groups

The API calls can be grouped by functionality. These groups are:

Open/Manage/Close Engine

- pr_open()
- pr_configure_ini()
- pr_report_error()
- pr_close()

Input/Process Image

- pr_read_image_RAM()
- pr_read_image_file()
- pr_image_information()
- pr_blank_image_area()
- pr_deskew_image()
- pr_preprocess_image()
- pr_auto_rotate()
- pr_save_image()
- pr_save_image_mpage()

Configure Page

- pr_set_new_document()
- pr_set_low_quality()
- pr_set_language()
- pr_set_printer_type()• pr_load_page_template()
- pr_set_user_dictionary()
- pr_configure_lexical_plus()
- pr_lexical_plus ()

Configure Zone

- pr_autozone()
- pr_get_autozone()
- pr_set_number_of_zones()
- pr_configure_zone()
- pr_configure_OMR_zone()
- pr_configure_image_zone()

Recognize Page

- pr_recognize_image()
- pr_language_id()

Get Output

- pr_output_xxx()

Convert RAM based Output to ...

- pr_convert_output_to_file()

Delete Output Memory

- pr_del_results()

Custom Functionality

- pr_string_options()
- pr_configure_output()
- pr_pre_process_PDF() (independent of the rest of the engine)

API Call Scope

API calls also differ in their scope. There are three levels of API call scope:

1. Engine - calls apply to the whole engine
2. Image or page - calls apply to one full image/page
3. Zone or multi-zone - calls apply to one or multiple zones

The API calls are organized below by their scope:

Engine Calls

- pr_open()
- pr_configure_ini()
- pr_report_error()
- pr_close()

Image Calls

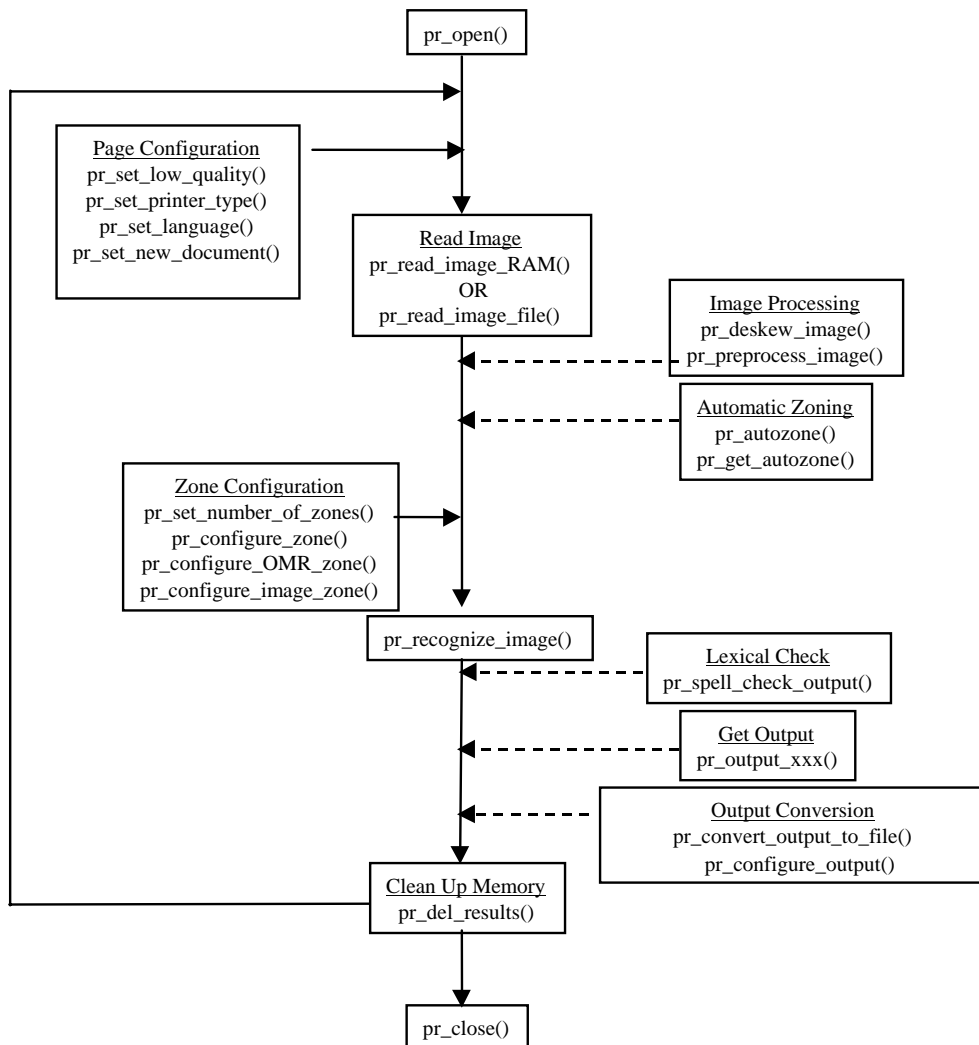
- pr_read_image_RAM()
- pr_read_image_file()
- pr_image_information()
- pr_blank_image_area()
- pr_deskew_image()
- pr_preprocess_image()
- pr_auto_rotate()
- pr_save_image()
- pr_save_image_mpage()
- pr_autozone()
- pr_set_new_document()
- pr_set_number_of_zones()
- pr_set_language()
- pr_set_low_quality()
- pr_set_printer_type()
- pr_recognize_image()
- pr_convert_output_to_file()
- pr_del_results()
- pr_load_page_template()
- pr_set_user_dictionary()
- pr_configure_lexical()
- pr_lexical_plus ()
- pr_configure_output ()
- pr_language_id ()

Zone Calls

- pr_get_auto_zone()
- pr_configure_zone()
- pr_configure_OMR_zone()
- pr_configure_image_zone()
- pr_output_xxx()

API Call Sequence

Each API call described later in this chapter includes a discussion of when it can be called in relation to other calls. A figure below gives a rough description of the flow for the more popular calls, however, please refer to the actual API descriptions for a more precise description. The line going down the middle of the diagram represents the minimum process for one page. Solid horizontal lines indicate mandatory calls, although the calls may only need to be made once at engine startup. Dotted horizontal lines indicate optional calls.



API Call Functionality

Configuration Memory

Many of the PrimeOCR API calls perform configuration. PrimeOCR retains configuration data between API calls. Any processing is based on all configuration data that is loaded at the time of process initiation. This means that most configuration need not be reentered for each page, e.g., the `pr_set_low_quality()` call can be made once, with all subsequent pages processed against this setting. Because PrimeOCR does not reset configuration data between pages any settings that need to change between pages must be set explicitly by the developer.

Changing one configuration value never causes other configuration settings to change, although it may make other settings incomplete or irrelevant. For example, assume that 5 zones were defined and processed on page one. On page two there are 7 zones so the number of zones is changed to 7. All the zone configuration data for zones 0-4 are still in memory and will be used for page two unless it is changed. Obviously zones 5 - 6 must be configured for page two. (Zone numbering starts with 0, for example, the fifth zone is zone number 4.)

If, instead, 7 zones were defined on page 1 and then 5 zones on page 2, zone data for zones 5 and 6 would remain in memory for page 2 but would be ignored since only zones numbered from 0 to number of zones-1 are processed.

Error reporting

Each function returns an integer or "status" variable indicating 0 for success and -X for errors.

Refer to *Chapter 7 - Troubleshooting* for more information on trouble shooting. The file `PROCRERR.TXT` includes error numbers and corresponding textual explanations.

Yielding

API calls which consume a significant amount of CPU time have been designed to periodically yield a small amount of time to other applications during processing. Developers may also design their OCR engine control applications to yield between PrimeOCR API calls and during PrimeOCR callbacks, if processing time is required by other applications. However, we do not recommend running CPU intensive, RAM intensive, or timing sensitive, applications concurrently with PrimeOCR.

If you implement the callback function then you may also add your own yield process in this call back function. This allows the developer to add more yields during recognition.

Call Return Timing

Each API call will return only when processing is complete. For all calls, except the `pr_recognize_image()`, processing will complete quickly. On large, bad quality images, with high accuracy settings, the `pr_recognize_image()` call can take up to several minutes to return.

Progress Reporting (Callback function)

A callback function may be registered through the `pr_open()` call. This function will receive messages from the PrimeOCR engine during its processing which indicates the progress of the engine.

API Calls

Programming to PrimeOCR API

Include Files	1
Libraries	1
Programming Languages	1
API Call Functionality Groups	2
API Call Scope	3
API Call Sequence.....	4
API Call Functionality.....	4
API Call Functionality.....	5
Configuration Memory.....	5
Error reporting.....	5
Yielding.....	5
Call Return Timing	5
Progress Reporting (Callback function).....	5
API Calls	6
pr_open().....	8
pr_configure_ini()	11
pr_report_error()	12
pr_close()	13
pr_read_image_RAM().....	14
pr_read_image_file().....	15
pr_image_information()	17
pr_get_image_information()	17
pr_blank_image_area()	20
pr_deskew_image().....	22
pr_preprocess_image().....	24
pr_auto_rotate().....	25
pr_save_image().....	27
pr_save_image_mpage()	28
NOTES ON IMAGES.....	29
FAX resolution files (204 X 196 or 204 X 98).....	29
Color/grayscale images.....	29
pr_set_new_document().....	31
pr_set_language().....	32
pr_set_low_quality()	34
pr_set_printer_type().....	35
pr_autozone()	36

pr_get_autozone()	39
pr_set_number_of_zones().....	40
pr_configure_zone().....	41
pr_configure_OMR_zone().....	45
pr_configure_image_zone()	47
pr_load_page_template().....	49
pr_recognize_image()	50
pr_language_id()	51
pr_set_user_dictionary()	53
pr_configure_lexical().....	54
pr_lexical_plus()	55
NOTES ON OUTPUT	57
pr_output_OMR_percent().....	58
pr_output_zone_lines()	59
pr_output_line_length().....	60
pr_output_line_data().....	61
pr_output_char_box().....	63
pr_configure_output()	64
pr_convert_output_to_file()	65
pr_convert_output_to_file()	65
pr_del_results()	68
pr_string_options().....	69
pr_pre_process_PDF()	70

pr_open()

short pr_open(HINSTANCE hInstance, HWND hWnd, short speed, FARPROC pr_callback);

Function:

Initializes PrimeOCR Engine.

Required?:

Required.

Timing:

Should be called only once and before any other PrimeOCR call.

Input Variables:

HINSTANCE hInstance

The instance of the user program. (This can be a positive dummy number if not available).

HWND hWnd

The handle to the user program window. Can be 0 if not available. If 0 then the update window will not be created (progress still available through callback).

short speed

The speed rating controls the number of processes that are run concurrently. Set the speed rating equal to the number of processors that are free for PrimeOCR processing. For example, if you have a dual processor NT machine, and only PrimeOCR is run on this machine, then you can set this to 2 (or 1 if you don't want to take advantage of the second CPU). Do not choose a number higher than the number of processors on your machine, higher than you are licensed for, or higher than the number of processors that are actually free for PrimeOCR processing. A setting of "0" loads each engine as required and unloads it immediately after processing. This method uses much less memory, but is quite slow, slowly consumes resources, and is only recommended for "demos" or other special situations.

A setting of -1 will allow PrimeOCR to sense how many processors exist on the machine, and to use all the processors (subject to any licensing restriction).



Memory Consumption

All engines required for an image are loaded and do not get removed until the PrimeOCR engine is terminated (unless "speed" is "0"). The memory requirements are: 5MB plus 10MB times the number of engines. For example, a three engine version would require 35MB of free RAM.

FARPROC pr_callback

The address of the callback function written by the developer. PrimeOCR will send a message to this function that indicates the current step in its processing. If a callback is not requested then enter NULL for the address. Typical messages are shown below:

<u>Message</u>	<u>PrimeOCR Step</u>
“Opening PrimeOCR Engine”	During pr_open()
“Reading d:\images\test.tif”	During pr_read_image_file() Includes name of image to be read by engine.
“Image size 990,890 RES=300”	At the end of pr_read_image_file() Includes pixel width, height, and resolution in dpi of image.
"Intermediate Results(% Complete)"	During pr_recognize_image() Indicates that the intermediate algorithms are being processed.
“XX”	During pr_recognize_image() Indicates percentage of processing that is complete. The percentage may momentarily decrease, or even start over at 0 again during processing. This is not a bug, it indicates that processing is looping back for a 2nd, 3rd, etc. pass.
“Processing Results”	During pr_recognize_image() Indicates that processing is almost complete.

Note: If you do not have significant experience with callback functions please use the callback function and code provided in the example application provided with this toolkit. Also review your compiler’s instructions regarding compile and make as they relate to callback functions.

The callback must use `_cdecl` calling convention. C# and VB, for example, require special coding in your application to enable this convention (C/C++ typically do not).

Typical Return Values:

A negative return value indicates an error. A positive return value indicates the version number of the software expressed as an integer. For example, "380" indicates release 3.80.

-100	Invalid hInstance
-101	Invalid hWnd
-111	Incompatible API version, license does not match software version. You may have updated the software without updating the license.
-112	Speed rating exceeded speed rating contained in license, in other words you tried to set a speed that was higher than speed rating licensed.
-113	Could not find a file called "PRIMEOCR.INI" in PRDEV\BIN directory.
-887	Could not find the hardware key, or hardware key contents were not read correctly, and no software license was found.
-4599	(1) Calling application was not in same directory as PROCR32.DLL, OR,

(2) FF20M32.DLL, FFWIN32.LOK, and PROCR32.DLL was not in PATH.



Application Location

If you locate your application in the \BIN directory no PATH modifications are required. If you wish to move your application to another directory then the \BIN directory must be added to the PATH statement. Call Prime Recognition if you need help in changing the PATH operating system settings.

pr_configure_ini()

short pr_configure_ini(char * string);

Function:

Allows configuration items that are typically found in the PRIMEOCR.INI file to be set via API calls. This means that configuration items can be configured without including the PRIMEOCR.INI file in the software distribution, and/or configuration items can be changed while the engine is running.

Required?:

Optional. If this call is not made, and if the PRIMEOCR.INI file does not exist then default settings are used. The default settings are those present in the PRIMEOCR.INI file when it was originally shipped by Prime Recognition.

Timing:

Can be made at any time, including BEFORE the pr_open() call. (This is the only call which can be made before the pr_open() call.)

Input Variables:

char * string

A string that exactly matches the line of text found in the PRIMEOCR.INI file. For example, "ConfidenceBbox=8", would set the ConfidenceBbox configuration value to 8.

Typical Return Values:

-1 or -2 if string was unrecognized.

Notes:

pr_report_error()

```
short pr_report_error(short error_number, char * string);
```

Function:

Returns a text message for an error number.

Required?:

Optional.

Timing:

Can be called at any time.

Input Variables:

short error_number

This error number is generated by the PrimeOCR engine on another call. Once sensed it is sent to this call to translate it into a text message.

Typical Return Values:

-1 PROCRERR.TXT file could not be opened.

Notes:

The call reads the PROCRERR.TXT file for the text message that corresponds to the error number. In some cases the user may wish to modify the PROCRERR.TXT file to return messages that better match their application or environment.

pr_close()

short pr_close();

Function:

Cleans up PrimeOCR engine resources right before exit.

Required?:

Required.

Timing:

Must be the last call made to PrimeOCR engine.

Input Variables:

None.

Typical Return Values:

None.

Notes:

Any calls made to the PrimeOCR engine after pr_close() are not valid.

pr_read_image_RAM()

```
short pr_read_image_RAM(short ImageWidth, short ImageHeight, short dpi,  
                        unsigned char *DataBlock);
```

Function:

"Sends" an image to PrimeOCR which developer has already placed in RAM memory, by telling PrimeOCR where to find the image (pointer to memory).

Required?:

This call OR `pr_read_image_file()` is required to give the PrimeOCR engine an image to process.

Timing:

Can be made at any time before the `pr_configure_zone()` call. Only needs to be called once per image. If multiple calls are made before `pr_recognize_image()` then only the last call is valid.

Input Variables:

short ImageWidth

Width of image in memory in pixels. Must be a multiple of 32.

short ImageHeight

Height of image in pixels. Must be a multiple of 32.

short dpi

Resolution of image in dots/pixels per inch. This setting must be 200, 240, 300, 400, or 600. 204 X 196 dpi images (equivalent of fine mode FAX) can be input as 200 dpi images. (200 X 98 FAX images are not supported via RAM input.) See NOTES ON IMAGES, after `pr_read_image_file()` description, for warning on FAX resolution image location accuracy.

unsigned char *DataBlock

Pointer to binary image in memory (color/gray scale not supported). The format in memory must be a raw bitmap, e.g., not a Windows bitmap. A brief description of this format is in *RAM Based Images*, Chapter 1.

Typical Return Values:

- 121 RAM input resolution was not 200, 240, 300, 400, or 600 square
- 130 ImageWidth not a multiple of 32
- 134 Image Width or Height too small (< 2 pixels)
- 136 Image Height not a multiple of 32

Notes:

See NOTES ON IMAGES after `pr_read_image_file()` description.

PrimeOCR does not free the image memory after use. The developer's application placed the image in memory, and is responsible for deleting the memory used by the image as well.

pr_read_image_file()

```
short pr_read_image_file(char * filename, short page_number, short * dpi,  
                        short *number_of_pages);
```

Function:

Reads an image into RAM for PrimeOCR engine usage from an image file, including multipage TIFF files. See Notes below for supported file formats. Image will be converted to a binary image in PrimeOCR's memory (even if it was color/grey scale in file format).

Required?:

This call OR pr_read_image_RAM() is required to give the PrimeOCR engine an image to process.

Timing:

Can be made at any time (i.e. sequence of calls) before the pr_configure_zone() call. Must be called only once per image or memory leaks will occur.

Input Variables:

char * filename

Pointer to a string which includes the full DOS path and filename for the image file, e.g. "D:\IMAGES\4931.TIF".

short page_number

Number of page within a multipage TIFF file (legal numbers are 1 < - > # of pages in file). If a single image file then this number should be 1.

Return Variables:

short * dpi

Resolution of image file in dots per inch.

short * number_of_pages

Returns total number of pages present in multipage TIFF file. **HOWEVER**, on some files it may just return a number that is +1 from "page_number". This indicates that the program can not sense the total number of pages but can sense that at least one more page exists in the file.

Typical Return Error Values:

- 121 PrimeOCR could not automatically convert incoming file to input resolution of 200, 240, 300, 400, or 600 square
- 130 ImageWidth not a multiple of 32
- 134 Image Width or Height too small (< 2 pixels)
- 135 ImageWidth results of file read were not a multiple of 32
- 140 No filepath/name given
- 141 Filepath/name exceeded 260 characters
- 142 PrimeOCR engine could not find filepath/name
- 144 File did not get read into memory

Notes:

Supported file formats include:

TIFF (binary, Version 5.0 and above)

-uncompressed

-Group 3 (fax)

-Group 3

-Group 4

-MultiPage

TIFF* (color\grey scale)

-uncompressed

-JPEG

JPEG* (color, grey scale)

PCX*

PDF **

*OCR-COLOR option required to read file.

** OCR-PDFIN option required to read file. PDF Image Only, PDF Image+Text, bw\grayscale\color, multiple page supported. PDF Normal is not supported but approximately 50% of PDF Normal files can be read. PDF files that are encrypted/secured cannot be read. Memory requirements increase by 30MB to 90MB (color) and more if images are bigger than 8.5x11. Processing time increases by 2-10 seconds per image.

File resolutions are read by the PrimeOCR engine and do not need be input by the developer. Supported file resolutions (dpi) are:

600 X 600

400 X 400

300 X 300

240 X 240

200 X 200

204 X 196 (Fine mode FAX)

204 X 98 (Standard mode FAX)

PrimeOCR can handle deviations from these desired values.

Standard mode FAX images will automatically be line doubled to 204 X 196 resolution by the PrimeOCR engine.

See *NOTES ON IMAGES* for warning on FAX resolution image location accuracy.

pr_image_information()

```
short pr_image_information(pr_image_info * image_data);
```

Function:

Returns width, height, dpi, of image currently in engine. If image was read from a file it will also indicate if this image was color/grayscale or some other non TIFF bitonal format.

THIS CALL WILL BE REMOVED IN THE NEXT RELEASE. USE PR_GET_IMAGE_INFORMATION() INSTEAD.

Required?:

Optional.

Timing:

Call can be made anytime after pr_read_image_xxx() and before pr_del_results().

Input Variables:

None

Output Variables:

pr_image_info * image_data

Pointer to structure that hold image data (defined in PRSTRUCT.H).

Width and height are measured in pixels. DPI is measured in dots or pixels per inch. "color_image" indicates whether image files read in by pr_read_image_file() call was a bitonal conventional TIFF file, or some other format such as color, grey scale, PCX, etc.

Typical Return Values:

-140 image_data pointer was null (i.e. not valid).
-118 image has not been loaded into memory yet.

Notes:

Among the uses of this function is to sense whether the image read was a color/greyscale, or some other non conventional image format. A flow of data from such a non standard format should be slightly different vs. a conventional bitonal TIFF file. See *NOTES ON IMAGES* later in this chapter.

pr_get_image_information()

short pr_get_image_information(short data_selector, short * data_return);

Function:

Can return width, height, dpi of image currently in engine. If image was read from a file it can indicate if this image was color/grayscale or some other non TIFF bitonal format. If deskew and/or "weak" autorotate was performed it will report sensed skew and rotation of the image.

Required?:

Optional.

Timing:

Call can be made anytime after pr_read_image_xxx() and before pr_del_results().

Input Variables:

short data_selector

- 0 = return width (in pixels) of image currently in memory
- 1 = return height (in pixels) of image currently in memory
- 2 = return resolution (in dpi) of image currently in memory
- 3 = return a flag that indicates whether image in memory was a bitonal TIFF file (value 0), or a color/grayscale, or unrecognized bitonal format (value 1). Only valid if image was read in from a file.

- 4 = returns amount of skew in file prior to deskew. (Only valid if deskew was performed on this image prior to this call). Values represent the number of horizontal pixels for every 1 pixel of vertical displacement, e.g. 50 means 50 pixels of horizontal displacement for every 1 pixel of vertical displacement (50 is noticeable skew). 500 would be very small skew. A negative number means that the skew is counter clockwise.

- 5 = returns the amount of rotation that "weak" autorotate (autorotate implemented through pr_enhance_image()) performed:
 - 0=no rotation (PrimeOCR thinks image is already upright)
 - 1=90 degrees clockwise (image originally was <----)
 - 2=180 degrees (image was upside down)
 - 3=270 degrees (image was --->)Note that weak autorotate is fast but only about 90% accurate so the value returned could be incorrect.

Output Variables:

short * data_return

Returns the data requested in "data_selector"

Typical Return Values:

- 140 image_data pointer was null (i.e. not valid).
- 118 image has not been loaded into memory yet.

Notes:

Among the uses of this function is to sense whether the image read was a color/greyscale, or some other non conventional image format. A flow of data from such a non standard format should be slightly different vs. a conventional bitonal TIFF file. See *NOTES ON IMAGES* later in this chapter.

pr_blank_image_area()

```
short pr_blank_image_area(short number_blank_zones, bbox * blank_zone_array, short
                          number_non_blank_zones, bbox * non_blank_zone_array);
```

Function:

Blanks rectangular areas of image in memory, except for those areas denoted by “non_blank_zone_array”.

Required?:

Optional.

Timing:

Can be made at any after pr_read_image_XXX() call and before pr_del_results() although most developers will want to call it before pr_recognize_image().

Input Variables:

short number_blank_zones

Number of zones defined in blank_zone_array. Limited to 100.

bbox * blank_zone_array

An array of coordinates on image that defines rectangular areas that should be blanked (any pixels removed). The units of the coordinates is BMUs (1/1200th of an inch) measured from top left corner of image (0,0). A bbox is a structure of four numbers that comprise x1,y1, x2, y2, where x1/y1 comprise the units of the top left corner of rectangle, and xy,y2 comprise the units of the bottom right corner of rectangle.

short number_non_blank_zones

Number of zones defined in non_blank_zone_array. Limited to 100.

bbox * non_blank_zone_array

An array of coordinates, similar to blank_zone_array, except these rectangles will not be blanked by this call.

Typical Return Error Values:

-118	No image is loaded into memory
-163	Array pointer was NULL
-180	x1 coordinate was <0 or >image width
-181	y1 coordinate was <0 or >image height
-182	x2 coordinate was <0 or >image width
-183	y2 coordinate was <0 or >image height

Notes:

There is no memory between calls to this function. In other words, for example, non blank zones defined in one call are not retained for future calls.

No correction is made to zone coordinates for deskew or other image enhancement operations. For example, if your coordinates are accurate on original image, they will not be accurate if the image was deskewed by PrimeOCR. If you desire this capability please call Prime Recognition.

Example C++ Usage:

```
bbox area, nonblankarea[2];
area.xy[0]=1200;
area.xy[2]=6000;
area.xy[1]=1200;
area.xy[3]=6000;
nonblankarea[0].xy[0]=2400;
nonblankarea[0].xy[1]=2400;
nonblankarea[0].xy[2]=3600;
nonblankarea[0].xy[3]=3600;

nonblankarea[1].xy[0]=4800;
nonblankarea[1].xy[1]=4800;
nonblankarea[1].xy[2]=5400;
nonblankarea[1].xy[3]=5400;

status_pi=pr_blank_image_area(1, &area, 2, nonblankarea);
```

This code will blank an area starting 1 inch in from left and top side, to 5 inches in from left and top side, with the exception of two rectangles within this area that will not be blanked. These rectangles are:

- 2 inches in from left/top, and 1 inch high and wide
- 4 inches in from left/top, and ½ inch high and wide

pr_deskew_image()

short pr_deskew_image(deskew flag);

Function:

Will automatically deskew the image in memory based on settings in PRIMAGE.INI file.

Required?:

Optional. (This is a purchased option).

Timing:

Call can be made anytime after pr_read_image_xxx() and before pr_recognize_image().

Input Variables:

deskew flag

deskew is a PrimeOCR defined enum. Allowed values for flag are:

"OFF",

Do not perform deskew (default).

"ON"

Perform deskew. Any zones defined are adjusted in location for any skew. On images with extremely high skew and very close zones this could lead to zones with a substantial amount of overlap.

"ON_NOZONEADJUST"

Perform deskew but does not adjust the zone definitions for any skew found. (Appropriate when you are supplying a fixed zone template and will be deskewing and registering image to fit into the fixed zone template layout.)

"ON_NOOUTPUTADJUST"

Perform deskew but does not adjust the output for any skew found. (Appropriate when you want the output coordinates to match the deskewed image, not the original image.)

"ON_NOZNOUTADJUST"

"ON_NOZONEADJUST" plus "ON_NOZNOUTADJUST".

Typical Return Values:

-1001 License does not include licensing for deskew.

Notes:

Prime Recognition uses ScanFix to perform its deskew. The settings in the PRIMAGE.INI file correspond exactly to ScanFix settings, and are documented in *Chapter 3*.

If the PRIMAGE.INI file does not exist then PrimeOCR uses default settings. The default settings are those settings present in the PRIMAGE.INI file as it shipped from Prime Recognition.

In general PRIMAGE.INI should be configured so that images with skew of less than .5% are not deskewed because deskew harms the characters resulting in lower accuracy OCR.

The following settings are recommended in PRIMAGE.INI:

mindetectlength=600
maxacceptableskew=250
deskewprotect=1

pr_preprocess_image()

short pr_preprocess_image(long flag);

Function:

Will preprocess the image in memory using the settings in the PRIMAGE.INI file.

Required?:

Optional. (This is a purchased option).

Timing:

Call can be made anytime after pr_read_image_xxx() and before pr_recognize_image().

Input Variables:

long flag

All operations desired are OR'd (added) together into this flag. The values for this flag are noted in PRSTRCT.H. For example a setting of "3" turns on PR_HORIZONTAL_REGISTER (1) and PR_VERTICAL_REGISTER (2).

Typical Return Values:

-1031 License does not include licensing for image preprocessing.

Notes:

Prime Recognition uses ScanFix to perform many (but not all) of its image preprocessing. The settings in the PRIMAGE.INI file correspond exactly to ScanFix settings, and are documented in *Chapter 3*.

If the PRIMAGE.INI file does not exist then PrimeOCR uses default settings. The default settings are those settings present in the PRIMAGE.INI file as it shipped from Prime Recognition.

The PR_AUTOROTATE setting is much faster than the pr_auto_rotate() call but significantly less accurate at finding the correct orientation. You can use both PR_AUTOROTATE and pr_auto_rotate(), indeed this is a good strategy in most cases. Call Prime Recognition to discuss our recommendations in this area. PR_AUTOROTATE does not work well with Asian Languages.

The PR_FIX_SIZE setting will change the size of the image (and stated resolution) so that IF it exceeds 5400 pixels in width or height, it will be changed to have a maximum size of 5400 pixels. (please contact Prime if you need this size modified).

pr_auto_rotate()

short pr_auto_rotate(short confidence_in, short * confidence_out, short * orientation);

Function:

Will automatically rotate the image until it finds an orientation that generates an OCR confidence of equal or greater to "confidence_in". If no orientations achieve this target then the orientation that generates the highest confidence is returned.

This function can be somewhat slow and is, in some ways, not very sophisticated. It runs one conventional engine on the full page to determine the OCR confidence. Images in which only a small minority of the text is captured may find this overhead too high, and/or if text on the page is rotated in several directions no clear answer may emerge from this function.

However, this function is very good at finding the correct orientation in the vast majority of cases. Another "auto_rotation" feature is available in the image preprocessing call. This call is significantly faster than pr_auto_rotate() however it is much less accurate.

For most users the pr_auto_rotate() function will work well, especially if performed as an exception process to the regular OCR processing, i.e., not performed on every image.

Required?:

Optional. (This is a purchased option: image enhancement).

Timing:

Call can be made anytime after pr_read_image_xxx() and before pr_recognize_image(), however it should be called right away after pr_read_image_xxx() so that any other calls are applied to the correctly oriented image.

Input Variables:

short confidence_in

This is a value from 100-900. The value of 700 is recommended for most applications. The function will rotate the image and calculate the OCR accuracy for this orientation. If the value exceeds "confidence_in" the function stops rotating the image. This rotated image is then used for all further processing within the engine. (For Asian languages this value is overwritten with 900).

Return Variables:

short * confidence_out

This is a value from 100-900. This value reports the OCR confidence of the orientation that was selected. You may enter a null pointer if you do not want this data.

short * orientation

This is a value from 0-3. This value reports the orientation that was selected.

0=original orientation

1=original orientation + 90 degrees

2=original orientation + 180 degrees

3=original orientation + 270 degrees

You may enter a null pointer if you do not want this data.

Typical Return Values:

- 810 confidence_in was <100 or >900
- 811 no image has been loaded to rotate
- 1031 License does not include licensing for image enhancement

Notes:

In most applications the number of pages that are not correctly oriented is a small percentage of the total. In this case we recommend you design your application so that it initially processes the image without calling the `pr_auto_rotate()` call. Use the "cut_process_short" string option to reduce processing on those pages that are rotated. Use the "average_confidence" string option to calculate the confidence of OCR. If it falls below a threshold, delete the OCR results, read the image file again, and this time call the `pr_auto_rotate()` function before processing image.

If no page exceeds the threshold, then the best orientation is selected, but only if the confidence of this page exceeds the confidence of the original orientation by a certain "hurdle" amount. By default this hurdle is 125. You can increase (or decrease) this hurdle amount via a string option entitled "rotate_hurdle". An example use of this string option would be "rotate_hurdle, 175". This would increase the hurdle to 175.

It is important for accuracy that the proper language is set prior to making this call. For example, if you set the language to English, but are submitting French documents, the accuracy of this call will be diminished significantly.

pr_save_image()

```
short pr_save_image(char * filename);
```

Function:

Will save the current image in memory to a TIFF group 4 file. This function is useful if you want to save the processed version of an image that has been deskewed or put through some other preprocessing step. It also has some use in debugging to see, for example, if a image loaded through pr_read_image_RAM() has been correctly loaded.

Required?:

Optional.

Timing:

Call can be made anytime after pr_read_image_xxx() and before pr_del_results().

Input Variables:

char * filename

Full path and file name to which you want to save image.

Typical Return Values:

-811 Image has not been loaded into engine

-814 filename is a null pointer

Notes:

THIS FUNCTION WILL BE REMOVED IN FUTURE RELEASES, USE pr_save_image_mpage() INSTEAD.

pr_save_image_mpage()

short pr_save_image_mpage(char * filename, short append);

Function:

Will save the current image in memory to a TIFF group 4 file (unless original image was in greyscale/color, see notes). This function is useful if you want to save the processed version of an image that has been deskewed or put through some other preprocessing step. It also has some use in debugging to see, for example, if a image loaded through pr_read_image_RAM() has been correctly loaded.

Required?:

Optional.

Timing:

Call can be made anytime after pr_read_image_xxx() and before pr_del_results().

Input Variables:

char * filename

Full path and file name to which you want to save image.

short append

PR_APPEND (0) will append this file to any existing file.

NO_APPEND (1) will cause a new file to be created (use for page 1 of a new file)

PR_LAST_PAGE (2) not currently used in this call.

Typical Return Values:

-811 Image has not been loaded into engine

-814 filename is a null pointer

Notes:

If the original image was in color or grayscale, then the saved image will be in color or grayscale. The only modification to the original image will be any autorotation that was performed to the image. (Deskew or other changes to the image in memory are not saved to this color/grayscale image). The Append functionality only works for multi-page file formats like TIFF (not, for example, JPEG).

NOTES ON IMAGES

Image size is limited to 22" x 22". However, OCR accuracy declines very rapidly for zones that exceed 5600 pixels in height or width. Therefore if the image is defined as one zone a practical limit of the image is 5600 pixels wide and high. (For asian language images the limit is 22.5 inches in height and width, and 12,500 pixels).

Text in images must be upright, PrimeOCR has two functions that can automatically orient the text correctly.

PrimeOCR will work on skewed images, however, skew in excess of .5% will harm OCR accuracy. We recommend you deskew images with skew above .5% (if possible, prior to being sent to PrimeOCR and prior to zone coordinate determination.) You may also wish to deskew images with less than .5% skew. The deskew process harms the image so OCR accuracy will actually decline slightly, however even small amounts of skew will hurt format retention so if this is important you may wish to trade off some OCR accuracy to achieve better format retention.

FAX resolution files (204 X 196 or 204 X 98)



Warning

The PrimeOCR engine must use a 200 X 200 resolution which leads to a 2% distortion of any image location data. If extreme image location accuracy is required then adjust all coordinates sent to the engine and received from the engine by plus or minus 2% as required.

PrimeOCR requires the following image attributes:

- The most significant bit is to the left of the least significant bit in each image byte
- A "0" bit is the equivalent of white on a image (and "1" is black)

Color/grayscale images

Color/grayscale images (or any type of image that is not a standard bitonal TIFF format file) is treated slightly different from a conventional file:

- You must have licensed the OCR-COLOR option to read these kinds of files.
- Internally PrimeOCR converts the image to a bitonal image for OCR purposes.
- You can not send these types of files to PrimeOCR via the `pr_read_image_RAM()` call (only the `pr_read_image_file()` call).
- Any image enhancement will be made on the binary version of the file, it will not affect the original version of file. Similarly, saving the file, for example often done after image enhancement, will save the binary version of the file, not a new color/grey scale version. This has some significant implications, for example, if you plan on using an output format that includes character location data, such as PRO, or PDF, you should only use the Deskew setting of "ON" (integer value of "1" in template file). This value of deskew means that the character location data (and word, line, and other location data) is always in terms of the original color/grayscale image.

-If you select an output option such as RTF or PDF Normal, and define image zones (either manually or through autozoning), the image zones will be in JPEG format.

pr_set_new_document()

short pr_set_new_document(document_source flag);

Function:

Notifies PrimeOCR Engine that this image is or is not from the same document (or a very similar document in terms of font and image quality) as the previous image.

Required?:

Optional. The default is that each page is from a new document/document type.

Timing:

Can be made at any time before the pr_recognize_image() call. Should be called only once per image. If this call is made multiple times before pr_recognize_image() then only the last call is actually acted upon.

Input Variables:

document_source flag

document_source is a PrimeOCR defined enum. Allowed values for flag are:

"NEW_DOCUMENT",

This image is from a different document or document type as the previous image.

"SAME_DOCUMENT"

Typical Return Values:

-150 flag was not a valid document_source value

Notes:

NEW_DOCUMENT is the default.

The SAME_DOCUMENT setting can increase the accuracy in some situations because PrimeOCR will "learn" between documents and OCR the later documents more accurately. However, if you fail to reset the flag to NEW_DOCUMENT when the document or type changes then accuracy on these later documents will be reduced. Do not expect visually noticeable accuracy improvements from this setting. The NEW_DOCUMENT setting is "safer" in that it assumes nothing about each image.

pr_set_language()

```
short pr_set_language(language flag);
```

Function:

Sets the language of the recognized text for a page. (This is a purchased option.)

Required?:

Optional. US is the default and is available in the base product.

Timing:

Can be made at any time before the pr_recognize_image() call. Should be called only once per image. If this call is made multiple times before recognize_image() then only the last call is acted upon.

Input Variables:

language flag

languages is a PrimeOCR enum. Allowed values for flag are:

```

"DANISH"
"DUTC" (Dutch)
"FRENCH"
"GERMAN"
"ITALIAN"
"NORWEGIAN"
"PORTUGUESE"
"SPANISH"
"SWEDISH"
"UK" (English)
"US" (English)
"ALL_LANGUAGES" (custom setting)
"CHINESE_SIMPLE"*
"CHINESE_TRADITIONAL"*
"JAPANESE"*
"KOREAN"*
"RUSSIAN"*

```

*(requires Asian Language option)

***(requires Russian Language option)

Typical Return Values:

-157 flag was not a valid language value

Notes:

If choosing a non-US language, make sure the other language files are loaded into PRDEV\BIN.

Asian Languages:

All will recognize English characters in the same document (accuracy on English characters is substantially below "US" setting).

These fonts are recognized:

Simplified Chinese: Hei, Song, Kai, FangSong

Traditional Chinese: Ming, Kai, Hei, Yuan

Korean: Batang, Myeongjo, Gothic

Japanese: MS Gothic, MS Mincho

Only PDF, RTF, ASCII/FASCII and a custom version of PRO file output are supported at this time for Asian Language output. If you are interested in other formats please let us know.

Russian Language:

The Russian characters within Unicode character set from 0401 – 044F are recognized.

Only PDF, RTF, ASCII/FASCII and a custom version of PRO file output are supported at this time for Russian Language output. If you are interested in other formats please let us know.

pr_set_low_quality()

short pr_set_low_quality(page_quality flag);

Function:

Indicates whether this image is low quality (e.g., poor quality standard mode FAX or dot matrix).

Required?:

Optional.

Timing:

Can be made at any time before the pr_recognize_image() call. Should be called only once per image. If this call is made multiple times before recognize_image() then only the last call is acted upon.

Input Variables:

page_quality flag

page_quality is a PrimeOCR enum. Allowed values for flag are:

"DEFAULT"

Default setting, no assumption made about quality of document.

"LOW_QUALITY"

This image is known to be quite poor quality.

Typical Return Values:

-151 flag was not a valid page_quality value

Notes:

Only set quality to "LOW_QUALITY" when the image quality is quite bad, e.g. poor quality standard mode fax. This setting will somewhat boost speed, with a small increase in accuracy.

pr_set_printer_type()

```
short pr_set_printer_type(print_type printer);
```

Function:

Indicates type of print (character formation) on this image.

Required?:

Optional. Default is machine print.

Timing:

Can be made at any time before the `pr_recognize_image()` call. Should be called only once per image. If this call is made multiple times before `pr_recognize_image()` then only the last call is acted upon.

Input Variables:

`print_type printer`

`print_type` is a PrimeOCR enum. Allowed values for `printer` are:

"MACHINE"

Machine print characters, i.e., fully formed characters from laser printer, typewriter, typeset, etc.

"MACHINE_FIXED"

MACHINE characters which are known to be fixed pitch (all characters have same width).

"DOT_MATRIX"

Characters were formed by a dot matrix printer with spaces between dots (if dots overlap then use the MACHINE setting.)

Typical Return Values:

-152 `printer` was not a valid `print_type` value.

Notes:

Dot matrix print has higher error rates, and takes longer to process than machine print characters.

pr_autozone()

```
config_zone * pr_autozone(auto_zone auto_order, short * num_zones, short * status);
```

Function:

Automatically determines zoning of a image.

Required?:

Optional. (Requires that the TextBridge engine be one of the licensed engines.)

Timing:

Must be made before pr_set_number_of_zones(), and pr_configure_zone(). Should be called only once per image. Although this call identifies zones the developer's application must still call pr_set_number_of_zones() and pr_configure_zone() for each zone.

Input Variables:

auto_zone auto_order

auto_zone is a PrimeOCR enum. Allowed values are:

"NO_AUTOZONE"

Used to turn off autozoning.

"PR_SEGMENT"

Segments the image into text zones (image zones are ignored).

Zones are returned in order of top to bottom of image.

Developer's application can review the zone information and modify anything including number of zones, zone location, etc. The OCR output is ordered as per the zone numbers.

Recommended for use with RTF output and most other applications.

"PR_ZONE"

Segments the image into text zones (image zones are ignored).

Zones are returned in order of top to bottom of image. (The

number of zones, zone order, and zone locations cannot be changed). Zones in OCR output are automatically re-ordered in the "reading order". Use PR_SEGMENT unless you really need the reading order of the text preserved.

"PR_SEGMENT_IMAGE_TEXT"

PR_SEGMENT except image zones are also captured. A image zone is a picture or some other relatively dense graphic. Warning: less dense graphics, such as charts, will probably not be sensed as an image zone. They will either be ignored or zoned as text.

"PR_ZONE_IMAGE_TEXT"

PR_ZONE except image zones are also captured. A image zone is a picture or some other relatively dense graphic.

Warning: less dense graphics, such as charts, will probably not be sensed as an image zone. They will either be ignored or zoned as text.

Return Variables:

short * num_zones

The number of zones defined by the image segmentation.

config_zone *

An array of config_zones (num_zones in size) that holds the coordinates of the segmented zones. Note that all other attributes of a config_zone are not filled in. PrimeOCR creates the memory for this array and frees the array memory during the pr_del_results() call.

Typical Error Values (status values):

-8001	Null num_zones address sent to function.
-8002	TextBridge engine was not defined for this version of PrimeOCR or is not available.
-8004	auto_order was not a valid auto_zone value

Notes:

Automatic zoning is important for text that is in columns. If this kind of text is processed without manual or automatic zoning then the reading order of text is destroyed. Automatic zoning will not work for fine levels of zoning, for example, distinguishing cells in a table or identifying separate fields on a insurance form.

The PR_SEGMENT mode is appropriate for developer's who want to review and potentially modify the text zones that are sensed by PrimeOCR before character recognition. The developer takes responsibility for the ordering of the zones.

The PR_ZONE mode is appropriate when the developer wants PrimeOCR to perform all zoning automatically, including the ordering of the output to follow the "reading order" of the text.

Automatic zoning is a difficult recognition task. Do not expect automatic zoning to be 100% accurate. On complex pages it is often virtually impossible to achieve a perfect reading order.

Although this call identifies zones the developer's application must still call pr_set_number_of_zones() and pr_configure_zone() for each zone. This allows the developer to set other attributes of each zone, such as LEXICAL check. It also allows the developer, in the PR_SEGMENT mode, to modify the number, location, and order of the zones.

Image zones, if recognized, will be reported first.

You can force autozoning to ignore the edges of a page when doing autozoning. See the file ...\\BIN\\"autozone_margins.ini.example" for more information.

The following code example illustrates an example usage of this call:

```
config_zone * zdata; short num_zones, status;
zdata=pr_autozone(PR_ZONE, &num_zones, &status);

status=pr_set_number_of_zones(num_zones);

//Generate configuration information for each zone
long zone_location[4];
for (i=0; i<num_zones; i++)
{
    zone_location[0]=zdata[i].zone_location.xy[0];
    zone_location[1]=zdata[i].zone_location.xy[1];
    zone_location[2]=zdata[i].zone_location.xy[2];
    zone_location[3]=zdata[i].zone_location.xy[3];

    // Send zone information to Prime Recognition engine
    status=pr_configure_zone(i,NO_RESTRICTIONS,LEXICAL,999,zone_location);
}
```

Autozoning is a difficult task for many applications. Please contact Prime Recognition to investigate whether we can custom modify our autozoning technology to work better in your application.

pr_get_autozone()

short pr_get_autozone(short zone_number, short *x1, short *y1, short *x2, short *y2);

Function:

Gets restriction and coordinates for a specific zone that was created by pr_autozone() call. Especially useful for VB programmers who can not get back config_zone structure from pr_autozone() call.

Required?:

Optional. (Requires that the TextBridge engine be one of the licensed engines.)

Timing:

Must be made after pr_autozone() call and before pr_set_number_of_zones(), and pr_configure_zone() calls.

Input Variables:

short zone_number
target zone number. Will range from 0 to number of zones returned by pr_autozone() call.

Return Variables:

short pr_get_autozone()
This value is the type of zone found, it can be NO_RESTRICTIONS, or PR_IMAGE.
short * x1, *y1, *x2, *y2
Bounding box of zone identified in BMUs (1/1200th of inch). All dimensions are relative to top left corner of image (0,0).

Typical Error Values (status values):

-114 Engine not open yet

-1901 zone_number >239

-1904 No autozoning data exists (probably did not call pr_autozone() yet).

Notes:

pr_set_number_of_zones()

```
short pr_set_number_of_zones(short num_zones);
```

Function:

Tells the PrimeOCR engine how many zones will be defined on this image.

Required?:

Required.

Timing:

Can be made at any time before the `pr_configure_zone()` call(s). Should be called only once per image. If this call is made multiple times before `recognize_image()` then only the last call is actually acted upon.

Input Variables:

`short num_zones`

Number of zones that will be defined on this image. This must be greater than 0.

Typical Return Values:

-160 `num_zones` is out of range (<1)

-162 `num_zones` is > than licensed capability for this version. Contact Prime Recognition to upgrade product to this number of zones.

Notes:

`num_zones` minus 1 defines the last zone number that is valid for this image. In other words if there are 3 zones on a page, they must be numbered 0,1,2. Zone numbers must start at 0, be consecutive, and stop at `num_zones` minus 1.

pr_configure_zone()

short pr_configure_zone(short zone_number, zone_content restrictions, lexical check,
long level, short zone_location[4]);

Function:

Sets configuration data for ONE zone.

Required?:

Required.

Timing:

Can be made at any time after number_of_zones() and pr_read_image_file/_RAM() and before the pr_recognize_image() call.

Input Variables:

short zone_number

Can be used to "name" or identify a zone for developer's purposes. zone_number is 0 based and must be between 0 and num_zones-1. The only significance to the PrimeOCR engine of a zone number is that the engine will process and report its results in order by zone number.

zone_content restrictions

zone_content is a PrimeOCR enum. Allowed values for restrictions are:

NO_RESTRICTIONS

PR_ALPHA (or ALPHA_ONLY for backward compatibility)

PR_NUMERIC (or NUMERIC_ONLY for backward compatibility)

NO_RESTRICTIONS_UPPERCASE

NO_RESTRICTIONS_LOWERCASE

PR_ALPHA_UPPERCASE

PR_ALPHA_LOWERCASE

PR_BARCODE (See prbarcode.ini for more configuration options)

The characters in each set are defined in Appendix A. If it is known that text in zone is alpha only, or numeric only then OCR will be faster and more accurate if the text is restricted. Otherwise set to NO_RESTRICTIONS.

The PRTRAIN custom engine also supports the following content types:

PRTRAIN_ALL (all defined character groups)

PRTRAIN_0 (Characters in group 0)

PRTRAIN_1 (Characters in group 1)

PRTRAIN_2 (Characters in group 2)

Contact Prime Recognition for more information on the PRTRAIN engine.

lexical check

lexical is a PrimeOCR enum. Allowed values are:

NO_LEXICAL

LEXICAL

LEXICAL performs a series of checks based on the restriction setting. For example, a NO_RESTRICTIONS setting with LEXICAL causes spelling checks, and tri-gram analysis to be performed.

A LEXICAL setting strongly encourages the recognition of characters in that group. All other characters are reported back as "?" with confidence level of "1".

A NO_LEXICAL setting causes recognition to encourage the recognition of characters in these groups, however, characters that are not in these groups may still be reported, particularly if they are high confidence.

long level

PrimeOCR has predefined several values for level:

- LEVEL_1 -provides conventional OCR accuracy and speed (e.g., M/Text OCR).
- LEVEL_2 -provides 15% better OCR accuracy (and other PrimeOCR benefits) at roughly 2.1 times slower speed than standard OCR.
- LEVEL_3 -provides 58% better OCR accuracy (and other PrimeOCR benefits) at roughly 3.1 times slower speed than standard OCR.
- LEVEL_4 -provides 74% better OCR accuracy (and other PrimeOCR benefits) at roughly 5.8 times slower speed than standard OCR.
- LEVEL_5 -provides 76% better OCR accuracy (and other PrimeOCR benefits) at roughly 7.1 times slower speed than standard OCR.
- LEVEL_6 -provides 80% better OCR accuracy (and other PrimeOCR benefits) at roughly 8 times slower speed than standard OCR.

(Levels 4, 5, and 6 are options to the base product. These test results are from a range of documents, your results could vary.)

Zones that require high accuracy should set accuracy improvement to the highest possible setting. If processing speed is more important than accuracy, then lower settings are available. Most OCR applications will benefit much more significantly from increased OCR accuracy than increased speed.

We encourage most developers to use these predefined values for level. However, developers may input their own values for level. An example value would be "770199". This tells the PrimeOCR engine to run engine five and six and weight each engine's output with a "7", do not run engine 4: "0", run engine three with a weight of "1", run engine 2 with a weight of "9", and run engine 1 with a weight of "9". (This example obviously assumes a PrimeOCR engine that is licensed for LEVEL_6) Another example is LEVEL_3 which is predefined as "999". This tells PrimeOCR to run engines 3, 2, and 1 with "9" weights. Leading zeros are assumed, therefore "999" implicitly means do not run engines 5 & 4 on this zone.

The weight of an engine can vary from 0-9. "0" tells PrimeOCR to not run the engine on this zone. "9" gives the highest weight to this engine. It is OK to give any number of engines the "9" weight.

If you are processing a range of document types then we strongly recommend staying with the LEVEL_X settings. If you are always running a very specific type of document then it may be worth experimenting with engine weights to see if accuracy/speed can be improved.

The potential benefits of engine weight tuning are two fold:

- You may choose to not run an engine. On a specific document type one engine may not be helping recognition enough to merit its processing requirements.
- You may give lower weights to the output of an engine that performs poorly on a specific document type. This results in higher PrimeOCR accuracy.

short zone_location[4]

Proper values must be >-32000 and <32000.

Coordinates of a rectangle that enclose the zone on the original image that was passed to PrimeOCR. These coordinates are measured in 1/1200 of an inch from the top left corner of the image (point 0,0). Zones may overlap but this is not recommended as text that is present in two zones will usually have higher error rates.

zone_location[0] is the x value of the top left hand corner of the zone.

zone_location[1] is the y value of the top left hand corner of the zone.

zone_location[2] is the x value of the bottom right hand corner of the zone.

zone_location[3] is the y value of the bottom right hand corner of the zone.

If each zone location coordinate is entered as 0 (zero), PrimeOCR will automatically set the zone coordinates for the full page. This feature is helpful for applications which want to OCR the whole page as one zone. A negative number will move the zone coordinate in from the edge of the page by the value of the number.

Typical Return Values:

- 153 restrictions was not a valid zone_content value
- 154 level was not a valid accuracy_improvement value.
- 161 number_of_zones <1 or number_of_zones > number of zones defined for this image
- 170 zone_number <0
- 171 zone_number > number of zones defined for this image
- 180 zone_location[0] either <0 or larger than image width
- 181 zone_location[1] either <0 or larger than image height
- 182 zone_location[2] either <0 or larger than image height or <= zone_location[0]
- 183 zone_location[3] either <0 or larger than image height or <= zone_location[1]

Notes:

If consecutive images contain the same zones then 1 or more zones can be configured once and then multiple images processed. Zone coordinates should obviously be contained within the actual image. Zones do not have to be defined in a particular order. If possible do not have the same text in two overlapping zones.

If you attempt to set an accuracy level which uses an engine which has not been installed,

or whose appropriate language file has not been installed then the engine will be automatically turned off and the action noted in the PROCR.LOG file.

pr_configure_OMR_zone()

```
short pr_configure_OMR_zone(short zone_number, short zone_location[4]);
```

Function:

Sets configuration data for one OMR zone. OMR stands for Optical Mark Recognition. A typical example would be a check box that is filled in on a form. The output for this zone will be a number 0-100, which describes the percentage of the zone that was "black".

Required?:

Optional.

Timing:

Can be made at any time after number_of_zones() and pr_read_image_file/_RAM() and before the pr_recognize_image() call.

Input Variables:

short zone_number

Can be used to "name" or identify a zone for developer's purposes. zone_number is 0 based and must be between 0 and num_zones-1. The only significance to the PrimeOCR engine of a zone number is that the engine will process and report its results in order by zone number. This zone_number must be an original number and part of a sequence with any other zone types such as regular zones or image zones.

short zone_location[4]

Proper values must be >-32000 and <32000.

Coordinates of a rectangle that enclose the zone on the original image that was passed to PrimeOCR. These coordinates are measured in 1/1200 of an inch from the top left corner of the image (point 0,0). Zones may overlap.

zone_location[0] is the x value of the top left hand corner of the zone.

zone_location[1] is the y value of the top left hand corner of the zone.

zone_location[2] is the x value of the bottom right hand corner of the zone.

zone_location[3] is the y value of the bottom right hand corner of the zone.

If each zone location coordinate is entered as 0 (zero), PrimeOCR will automatically set the zone coordinates for the full page. This feature is helpful for applications which want to OCR the whole page as one zone. A negative number will move the zone coordinate in from the edge of the page by the value of the number.

Typical Return Values:

-161 number_of_zones <1 or number_of_zones > number of zones defined for this image

-170 zone_number <0

-171 zone_number > number of zones defined for this image

- 180 zone_location[0] either <0 or larger than image width
- 181 zone_location[1] either <0 or larger than image height
- 182 zone_location[2] either <0 or larger than image height or <= zone_location[0]
- 183 zone_location[3] either <0 or larger than image height or <= zone_location[1]

Notes:

If consecutive images contain the same zones then 1 or more zones can be configured once and then multiple images processed. Zone coordinates should obviously be contained within the actual image. Zones do not have to be defined in a particular order.

pr_configure_image_zone()

```
short pr_configure_image_zone(short zone_number, short zone_location[4]);
```

Function:

Sets configuration data for one image zone. The output for this zone will be a G4 TIFF file comprising only the area described by the zone location (if the original image was color/greyscale or a non standard file format then the image zones will be in JPEG format). On execution of this call a file with the name “~PRIMG”, and the extension “AAAABBBB” (where A and B is defined below), is created in the application’s directory (typically “...\PRDEV\BIN). A call to pr_convert_output_to_file() then changes the name and moves the file to the output directory.

Required?:

Optional.

Timing:

Can be made at any time after number_of_zones() and pr_read_image_file/_RAM() and before the pr_recognize_image() call.

Input Variables:

short zone_number

Can be used to "name" or identify a zone for developer's purposes. zone_number is 0 based and must be between 0 and num_zones-1. The only significance to the PrimeOCR engine of a zone number is that the engine will process and report its results in order by zone number. This zone_number must an original number and part of a sequence with any other zone types such as regular zones or image zones.

short zone_location[4]

Proper values must be 0 and <32000 and less than image width/image height.

Coordinates of a rectangle that enclose the zone on the original image that was passed to PrimeOCR. These coordinates are measured in 1/1200 of an inch from the top left corner of the image (point 0,0). Zones may overlap.

zone_location[0] is the x value of the top left hand corner of the zone.

zone_location[1] is the y value of the top left hand corner of the zone.

zone_location[2] is the x value of the bottom right hand corner of the zone.

zone_location[3] is the y value of the bottom right hand corner of the zone.

Typical Return Values:

- 161 number_of_zones <1 or number_of_zones > number of zones defined for this image
- 170 zone_number <0
- 171 zone_number > number of zones defined for this image
- 180 zone_location[0] either <0 or larger than image width
- 181 zone_location[1] either <0 or larger than image height
- 182 zone_location[2] either <0 or larger than image height or <= zone_location[0]
- 183 zone_location[3] either <0 or larger than image height or <= zone_location[1]

Notes:

Image zones are saved with an eight character extension that denotes the page number and image zone on that page. The first four characters of the extension refer to the page number. The last four characters refer to the zone number on that page. (Zone numbers start at 0). Each character can vary from 0-9. For example, an image zone on the 12th page, that has a zone number of 2 will be named “~PRIMG.00120002”. Therefore “0012” refers to the 12th page in the document, and “0002” refers to the 3rd zone on that page (the first zone is 0, second zone is 1, third zone is 2, etc.).

pr_load_page_template()

short pr_load_page_template(char * filename, short * num_zones);

Function:

Reads a template file and makes relevant calls to configure engine. Some functionality supported in a template file can not be set by this call (see "Notes below") and any document settings (such as output format) are also ignored. This call is useful to set the page and zone data for a single page.

Required?:

Optional.

Timing:

Can be called anytime after pr_read_image_xxx() and before pr_recognize_image() call.

Input Variables:

char * filename

The full path including file name and extension of the PrimeOCR format template file. This template file must be in a version that is compatible with engine.

Return Variables:

short * num_zones

The number of zones described in the template file.

Typical Return Values:

- 140 filename was a null pointer
- 141 filename was >260 characters in length
- 142 filename could not be found
- 144 could not open filename file
- 146 num_zones was a null pointer

Notes:

If autorotation, deskew, image preprocessing, or save image settings are non zero then these functions will be called in that order within this call. The image saved in the pr_save_image() call will be the same name as the original file but with a ".fix" extension.

Any document and autozoning settings are ignored.

pr_recognize_image()

```
zonestructure * pr_recognize_image(short *status);
```

Function:

Performs recognition on image based on all configuration settings and returns recognized characters and other output.

Required?:

Required.

Timing:

Should be performed only after all configuration data for the engine, page, and zones have been set.

Input Variables:

None (status is a returned value).

Return Values:

The PrimeOCR results are an array of zonestructures (defined in PRSTRCT.H). There are "num_zones" number of elements in the array. ("num_zones" was defined in pr_set_number_of_zones().)

If an error occurs the return pointer is NULL, no output is returned.

Typical "status" Return Values:

- 283 Too little RAM available. Free up RAM by closing other applications or rebooting. PrimeOCR's memory requirements are described in *Chapter 2*.
- 400 Process exceeded maximum time length. Most likely cause is too little RAM available.

- 887 License problem.
to
-997

- 998 Hardware key functioning but incorrect key attached to the port. **ONLY RUN THE SOFTWARE WITH THE KEY THAT CAME WITH THE PRODUCT.** Call Prime Recognition.

- 9999 Too little RAM available. Free up RAM by closing other applications or rebooting. PrimeOCR's memory requirements are described in *Chapter 2*.
- XXX Please report any other error messages to Prime Recognition.

Notes:

This function will not return until PrimeOCR processing is complete. On poor quality images with a lot of text, on a slow PC the PrimeOCR engine could take several minutes to return.

pr_language_id()

short pr_language_id(short confidence_in, long accuracy_level, short asian_language_check, short * confidence_out, short * language1, short * language1_chars, short * language2, short * language2_chars);

Function:

Reports back two most prevalent languages of text on page. Includes confidence of analysis, and number of characters of each language.

Required?:

Optional.

Timing:

Should be performed only after all configuration data for the engine, page, and zones have been set. The timing is the same as the pr_recognize_image() call.

Input Variables:

short confidence_in

If the confidence level of the roman language identification exceeds this level, asian languages will not be tested (800 is a good level).

long accuracy_level

Sets which engines will be used for processing in pr_language_id(). We recommend 99 (Level 2), other common choices would be 9 (Level 1), or 90099(Level 3). Level 2 is the best combination of speed and accuracy.

short asian_language_check

If non zero then Asian languages will be checked. (Can slow down processing considerably, if Asian languages are not present set this to 0). Requires an Asian Language Option license.

Return Values:

short * confidence_out

Reports the confidence of language identification, range is 100-900, with 900 meaning very high confidence. Very roughly, 800 + is high confidence, 700-800 is marginal, <700 is low confidence.

short * language1

Reports the language that is most prevalent on page. Values can range from 0 to number of languages supported by PrimeOCR (for example, 15 in version 4.2, 11 roman languages, 4 Asian languages). The values for valid languages can be found in prstrct.h (look for enum language).

short * language1_chars

Reports the number of characters of language1 found on page. This number is an approximation, it is not precise.

short * language2

Reports the language that is second most prevalent on page.

short * language2_chars

Reports the number of characters of language2 found on page.

Typical "status" Return Values:

Notes:

Make sure appropriate language packs are loaded. (Standard installation only includes English, two languages packs are available, roman and Asian languages).

The following are strong recommendations but not requirements:

pr_set_language() should be set to French for most accurate recognition of roman languages.

Use pr_autozone() prior to pr_language_id to find zones unless you know images are single zone. The define the autozone results to the engine using pr_set_number_of_zones() and pr_configure_zone().

pr_set_user_dictionary()

short pr_set_user_dictionary (char * user_dictionary);

Function:

Configures PrimeOCR to use a user dictionary during character recognition and, optionally, during pr_lexical_plus() call.

Required?:

Optional.

Timing:

Should be called prior to any other lexical check call. Can be called multiple times, however, if user_dictionary value does not change then user_dictionary will not be reloaded.

Input Variables:

None.

Typical Return Values:

0	Dictionary loaded successfully
-114	Engine not loaded
-143	Could not find dictionary file/path

Notes:

This call only sets a configuration flag. The actual usage of dictionary occurs later during processing.

Dictionary file should be an ASCII text file, with one word per line. Words must be between 2-32 characters in length. We recommend you place only uncommon words in this file, common words are most likely to be already in the existing main dictionary. One way to see if you should add a word to the user dictionary is to turn on the Lexical Plus functionality and run through some sample documents, with the log file turned on. Check ...\\lexical.log to see what words were not found in the existing main dictionary (or existing user dictionary), add these words to user dictionary.

There is no hard limit to the number of words you can add to the user dictionary, however, speed and effectiveness start to decline if dictionary exceeds approximately 5000 words.

pr_configure_lexical()

short pr_lexical_change_options(short language, short score_threshold, short debug);

Function:

Configure the lexical plus engine.

Required?:

Optional.

Timing:

Should be called before pr_lexical_plus().

Input Variables:

short language

Language to be used for lexical checks (Only "10", which corresponds to short value of language enum for US English, is allowed in current release).

short score_threshold

The minimum value a lexical check word must score to be considered a valid solution for replacing the original word.

short debug

If a non zero value then the activities of the lexical plus engine are reported in a ASCII file, LEXICAL.LOG, in the application's directory (e.g. PRDEV\BIN).

Typical Return Values:

0 Threshold changed successfully
-20949 Lexical engine has not been opened

Notes:

The lexical plus engine will generate scores for possible changes to a suspect word. Higher scores indicate higher confidence that the word suggested is the correct match for the candidate word. Any lexicon solutions that have a score lower than the score_threshold are discarded. The highest scoring solution is selected as the solution.

The score_threshold range is 60-98. For example, a score_threshold of 95 will narrow solutions to words that match candidate words very closely. Erroneous solutions are minimized with a score_threshold set this high. With a score threshold of 65, on the other hand, the lexicon engine will provide more solutions but the chances of getting in-correct solutions increases. The default score threshold of 85 reduces in-correct solutions while maximizing opportunities to find lexical solutions to candidate words.

We strongly suggest you turn on the debug setting when first using the lexical plus engine so as to better understand the changes that this technology makes to the output.

Special features of Lexical Plus are implemented in LEXPLUSPRO.INI in the INSTALLATION\BIN directory.

pr_lexical_plus()

short pr_lexical_plus (zonestructure * good, short number_of_zones, short zone_number, short spell_check, short suspect_level, short lexical_good_confidence_weight, short lexical_fix_confidence_weight, short lexical_no_fix_confidence_set);

Function:

Uses intelligence based on language understanding to improve OCR results. Can be used to modify confidence levels of output and/or change recognized characters.

Required?:

Optional.

Timing:

Should be called only once, following pr_recognize_image(). Should be called after pr_configure_lexical_plus() and pr_set_dictionary() (if a user dictionary is desired).

Input Variables:

zonestructure * pr_output

The pointer (or equivalent) that was returned from pr_recognize_image().

short number_of_zones

Number of zones to send to the file. This number must be equal to or less than the original number of zones reported by PrimeOCR.

short zone_number

Zone number to check with lexical plus process. "-1" will process all zones.

short spell_check

-Range: 0-9, default=8 (if non zero)

-Any word containing one or more characters at this confidence level or below will be "suspect" and will be checked via a sophisticated spell check process.

-A higher number is more aggressive at applying spell check to more and more words in document.

-This number can usually be set higher than LexicalCheck below, since no changes are made to characters, and any confidence changes are likely to be correct.

short lexical_good_confidence_weight

-Range: 0-9, default=8

-0: no changes to confidence values if suspect word is changed by lexical process

-Corresponds to spell_check setting. ("spell_check" turns on this portion of lexical process, this setting controls what happens if characters/words are found in).

-If spell_check is non zero, and a word is found in lexical plus process, then confidence of characters in word are increased in proportion to this value.

short suspect_level

-Range: 0-9, default=6 (if non zero)

-Any word containing one or more characters at this confidence level or below

will be "suspect" and sent to lexical process for possible change. A higher number is more aggressive at applying lexical check even to characters more likely to be correct. Be cautious in increasing this value above 6 or 7, unless your application lends itself clearly to lexical processing.

short lexical_fix_confidence_weight

-Range: 0-9, default=6 (if non zero)

-0: no changes to confidence values if suspect word is changed by lexical process

-Corresponds to suspect_level setting. ("suspect_level" turns on this portion of lexical process, this setting controls what happens if characters/words are modified).

-Character confidence values are increased proportional to this weight factor (if changed by lexical plus process). Use this feature, for example, to reduce "suspicious" characters during later manual verification. Be cautious in increasing this value above 6 or 7, unless your application lends itself clearly to lexical processing.

short lexical_no_fix_confidence_set

-Range: 0-9, default=2 (if non zero)

-0: no changes to confidence values if suspect word is changed by lexical process

-Corresponds to suspect_level setting. ("suspect_level" turns on this portion of lexical process, this setting controls what happens if characters/words are not found in lexical database).

-Indicates the confidence level to which a character will be changed if the word in which it was placed was "suspicious" and not found in lexical database. Use this feature, for example, to find "suspicious" characters for later manual verification. In general you should not use this feature unless your application words conform very closely to a user dictionary and/or our general language dictionary.

Typical Return Values:

- 0 Output successfully checked
- 5000 Confidence mask was not valid
- 5001 AddItem function failed, lexical_word_in not valid
- 20949 Output could not be checked because lexical engine was not started, or lexicons were not loaded.
- 20959 Could not check zone because lexical engine was not started, or lexicons were not loaded.
- 20969 Could not check word because lexical engine was not started, or lexicons were not loaded.

Notes:

Do not use this call unless your application lends itself to lexical analysis.

Always turn on debug setting in pr_configure_lexical() and look at results in LEXICAL.LOG in a trial run to make sure the changes caused by pr_lexical_plus() are positive to your application.

NOTES ON OUTPUT

You can access PrimeOCR's output in three ways.

1. Access the output in memory directly by manipulating the zonestructure returned by the `pr_recognize_image()` call. The C source code example included shows how to do this. Programmers comfortable with C will find this method easy and fast.
2. Access the output in memory by using the `pr_output_xxx()` calls. All output information is available through these calls. Visual Basic programmers will find these calls very useful, it is much easier to return a single dimension array to Visual Basic from a DLL than it is to return a pointer to a data structure and then try to manipulate the data structure directly. C programmers may also wish to use these calls if they like to have explicit calls to get output.
3. Use the `pr_convert_output_to_file()` call to move output into a file. From here it can be accessed as any text file can be accessed by C, Visual Basic, or any other programming language.

pr_output_OMR_percent()

```
short pr_output_OMR_percent(short zone_number);
```

Function:

Returns the percent of OMR zone that is black pixels.

Required?:

Optional.

Timing:

Should be performed only after `pr_recognize_image()` and before `pr_del_results()`.

Input Variables:

short zone_number

The target zone number. Must be between 0 and number of zones -1.

Return Values:

Returns the percent of OMR zone that is black pixels. If the number is negative (0 is a possible outcome if zone was totally white pixels) then the return is an error.

Typical Error Return Values:

-1901 zone_number was larger than the number of zones -1

Notes:

See *Notes on Output* section.

pr_output_zone_lines()

```
short pr_output_zone_lines(short zone_number);
```

Function:

Returns the number of lines in a zone.

Required?:

Optional.

Timing:

Should be performed only after `pr_recognize_image()` and before `pr_del_results()`.

Input Variables:

`short zone_number`

The target zone number. Must be between 0 and number of zones -1.

Return Values:

Returns the number of lines in a zone. If the number is negative (0 is a possible outcome if no text was recognized in the zone) then the return is an error.

Typical Error Return Values:

-1901 `zone_number` was larger than the number of zones -1

Notes:

See *Notes on Output* section.

pr_output_line_length()

short pr_output_line_length(short zone_number, short line_number);

Function:

Returns the length of a specific line in a specific zone.

Required?:

Optional.

Timing:

Should be performed only after pr_recognize_image() and before pr_del_results(). Will probably be used after pr_output_zone_lines() was used to identify the number of lines in a zone.

Input Variables:

short zone_number

The target zone number. Must be between 0 and number of zones -1.

short line_number

The target line number. Must be between 0 and number of lines -1.

Return Values:

Returns the number of characters in the target line, or, in other words, the length of the line in characters. If the number is negative then the return is an error. 0 is not a possible outcome as a line will not be reported in the first place if no text was recognized in the line.)

Typical Error Return Values:

-1901 zone_number was larger than the number of zones -1

-1902 line_number was larger than the number of lines -1

-1904 Operation assumes OCR output but no OCR output has been created yet or it was already erased.

Notes:

See *Notes on Output* section.

pr_output_line_data()

```
short pr_output_line_data(short zone_number, short line_number, short * x1, short *y1, short *x2,  
    short *y2, short * height, char * text, char * confidence, char * chardata, char *pointsize);
```

Function:

Returns all OCR data for a specific line except for character bounding box data. (See `pr_output_char_box()` for character bounding box data.)

Required?:

Optional.

Timing:

Should be performed only after `pr_recognize_image()` and before `pr_del_results()`. Will probably be used after `pr_output_line_length()` was used to identify the size of the line so that the proper amount of memory can be allocated for the line results.

Input Variables:

`short zone_number`

The target zone number. Must be between 0 and number of zones -1.

`short line_number`

The target line number. Must be between 0 and number of lines -1.

Return Values:

All return values are returned through pointers supplied by the developer through the call. If any pointer value is NULL the value will not be returned. In other words, if you don't want a particular output then enter NULL in its place in the call.

`short * x1, short *y1, short *x2, short *y2, short * height`

These are pointers to integers supplied by the developer's application. The call will set the value of these pointers to the baseline coordinates (x1, y1, x2, y2) of the line plus the height of the line. Refer to *Chapter 5* for a discussion of these attributes.

`char * text, char * confidence, char * chardata, char * pointsize`

These are pointers to strings supplied by the developer's application. The call will copy the output to these strings. The strings represent the recognized ASCII text, the confidence level for each character, the character data (bold, underline, etc.) for each character, and the point size of each character. Refer to *Chapter 5* for a discussion of these attributes. The developer's application is responsible for allocating enough memory in the strings to hold the output. The size of each string is exactly equal to the line length reported by `pr_output_line_length()` plus 1 (for the terminating 0).

Typical Error Return Values:

- 1901 zone_number was larger than the number of zones -1
- 1902 line_number was larger than the number of lines -1
- 1904 Operation assumes OCR output but no OCR output has been created yet or it was already erased.

Notes:

See *Notes on Output* section.

pr_output_char_box()

```
short pr_output_char_box(short zone_number, short line_number, short character_number,  
                        short * x1, short *y1, short *x2, short *y2);
```

Function:

Returns the bounding box for a specific character.

Required?:

Optional.

Timing:

Should be performed only after `pr_recognize_image()` and before `pr_del_results()`. Will be probably used after `pr_output_line_length()` was used to identify the proper range of characters.

Input Variables:

short zone_number

The target zone number. Must be between 0 and number of zones -1.

short line_number

The target line number. Must be between 0 and number of lines -1.

short character_number

The target character number. Must be between 0 and number of characters in the line -1.

Return Values:

All return values are returned through pointers supplied by the developer through the call. If any pointer value is NULL the value will not be returned. In other words, if you don't want a particular output then enter NULL in its place in the call.

short * x1, short *y1, short *x2, short *y2

These are pointers to integers supplied by the developer's application. The call will set the value of these pointers to the character bounding box (x1, y1, x2, y2). Refer to Chapter 5 for a discussion of the character bounding box.

Typical Error Return Values:

- 1901 zone_number was larger than the number of zones -1
- 1902 line_number was larger than the number of lines -1
- 1903 character_number was larger than the length of line -1
- 1904 Operation assumes OCR output but no OCR output has been created yet or it was already erased.

Notes:

See *Notes on Output* section.

pr_configure_output()

short pr_configure_output(enum config_output flag);

Function:

Adds word data to “in memory” and PRO file based output.

Required?:

Optional.

Timing:

Should be performed only after pr_recognize_image() and before pr_del_results(). If used to modify PRO file based output it should be performed before pr_convert_output_to_file(). This call is not “sticky”, i.e. it must be called for every page.

Input Variables:

enum config_output flag

If set to 1 it will add word data to in memory data structure and PRO file output.

Typical Error Return Values:

- 114 PrimeOCR engine is not open
- 1561 flag value was illegal
- 1904 Operation assumes OCR output but no OCR output has been created yet or it was already erased.

Notes:

The impact on PRO file based output is as follows:

“Format” on line 1 of output will be “1” instead of “0”.

After the standard output for each line of text, 7 lines of data are added:

- # of words on line of text
- Character positions (zero based) that start each word (separated by commas)
(e.g. Word 2 starts at character position 13)
- Character positions (zero based) that end each word (separated by commas)
(e.g. Word 2 ends at character position 22)
- Left edge in BMUs of each word (separated by commas)
- Top edge in BMUs of each word (separated by commas)
- Right edge in BMUs of each word (separated by commas)
- Bottom edge in BMUs of each word (separated by commas).

The “in memory” impact is defined in *prstrct.h* (see, for example, prword structure).

pr_convert_output_to_file()

```
short pr_convert_output_to_file(zonestructure * pr_output, short number_of_zones, char *
    out_filename, output_type output, char * image_name, char * user_tag, short append);
```

Function:

Converts RAM based PrimeOCR output to a file based output. If image zones were defined the image zone files created during the pr_configure_image_zone() may be converted as appropriate depending on the output format chosen.

Required?:

Optional.

Timing:

Should be performed after pr_recognize_image() and before pr_del_results().

Input Variables:

zonestructure * pr_output

The pointer (or equivalent) that was returned from pr_recognize_image().

short number_of_zones

Number of zones to send to the file. This number must be equal to or less than the original number of zones reported by PrimeOCR.

output_type output

output_type is a PrimeOCR enum. Allowed values are:

```
ASCII
FASCII
PDA2
PR
ASCII_NOZONEBUFFER
RRI3
FASCIITAG
ZY3
PRSTAR
PR_RTF
PDFTO
PDFIO
PDFIT
COMMA
HTML
```

Please see *Chapter 6 – Output Formats* for a description of each format.

char * image_name

A pointer to a character string supplied by the developer. Should include the full path/file name of the image. This string is optional in general but required in a number of cases including PDF output, if PRO will be read by PrimeVerify and PrimePost, etc.. (If input is a PDF color image then this string should be defined as 256 characters in length, and PrimeOCR will change this path to a path to a temporary TIFF image). Enter a blank string (“”) if no entry is desired. Output as the third line in a PR output file. Note that if this string is present the third line of output will include not just the image name, but a tab character, followed

by the page number, another tab character, followed by a flag that indicates whether the image is black&white (0), or color/gray scale (1). These later two parameters are automatically supplied by PrimeOCR.

char * user_tag

A pointer to an arbitrary character string supplied by the developer. This string is optional. Enter a blank string (“”) if no entry is desired. If the string is present it will be output as the second line in a PR output file.

char * out_filename

A pointer to a character string which includes full path and DOS file name. This string is required. This string points to the location to which to write the output file. Developer may attach any file extension desired, PrimeOCR does not supply a default extension.

short append

Allowed values are:

PR_APPEND
NO_APPEND
PR_LAST_PAGE

PR_APPEND and NO_APPEND are mutually exclusive of each other.
PR_LAST_PAGE may OR'd to either value.

PR_APPEND will add the new output to the end of an existing file, if it exists. If a file does not exist it will create a new file. The added output is not changed in any way by PR_APPEND (except for RTF and PDF formats), in other words, for example, under the PR output format, the added material will include the 3 header lines, etc. No attempt is made to insert any other delimiter between previous output and this output.

NO_APPEND will overwrite any existing file, or will create a new file if one does not already exist.

PR_LAST_PAGE signals to the engine that this is the last page of output in this file. **This value must be set for the PDF output option**, however, no other output options require this flag.

Typical Return Values:

-140 No output filepath/name given
-141 Output filepath/name exceeded 260 characters
-143 PrimeOCR engine could not find output path or did not have writing privileges to this path
-156 flag was not a valid output_type value
-1562 flag was not a valid output_option value
-161 number_of_zones <1 or number_of_zones > number of zones defined for this image
-190 pr_output pointer is not valid

Notes:

If RTF or PRO formats are chosen and image zones were defined then the image zone files that were produced to the application directory (see pr_configure_image_zone()) are renamed with the image's file name and moved to the same directory as the RTF or PRO

file.

If the PDFTO (PDF Normal) format was chosen then the image zone files are removed from the application directory and incorporated directly into the PDF format file.

If PDF is the image input format and output format is PRO then we recommend saving the internal OCR image to a file and putting this file in the image path. Tools such as PrimeVerify and PrimePost that may need access to the input image can not read PDF files.

Bookmarks may be added to PDF output. See ...\\BIN\\BOOKMARK.INI.EXAMPLE for more information.

Document properties may be added to PDF output. See ...\\BIN\\DOCPROP_PDF.INI.EXAMPLE for more information.

pr_del_results()

```
short pr_del_results(zonestructure * pr_output, short delete_flag);
```

Function:

Frees up memory allocated by PrimeOCR for one image/page. Memory allocations occur within the PrimeOCR engine occur during `pr_read_image_file()`, `pr_autozone()`, and `pr_recognize_image()`.

Required?:

Yes. If the memory used by PrimeOCR is not freed then eventually PrimeOCR output will consume all available memory and Windows will not function properly.

Timing:

This call should be made after the last relevant call has been made, for most developers that will be `pr_recognize_image()`.

So, for example, this call should be performed after PrimeOCR output has been processed by the end user. Freeing the memory will effectively erase the PrimeOCR output (`pr_del_results()` does not blank the results but it tells Windows that this memory is now free for other uses.)

Input Variables:

`zonestructure * pr_output`

The pointer that was returned from `pr_recognize_image()`(or string "CURRENT RESULTS" which is handy if you are in VB). If `pr_recognize_image()` has not been called (and will not be called), then you can input NULL here.

`short delete_flag`

If this flag is less than 0 then the image in memory will not be deleted (and may be used again). Any number greater than -1 will cause all memory to be deleted.

Typical Return Values:

-13XXX

Exception caused by deleting memory. If you are manipulating the output in memory you may be causing this problem (copy the output first to your own datastructure before manipulating it). Or this error may be caused by Windows memory getting scrambled over time, and application should be restarted, and in some more extreme cases the PC may need to be rebooted. If this problem continues we would recommend a weekly (or daily schedule in extreme cases) of rebooting the PC.

Notes:

Converting RAM based PrimeOCR output to a file does not automatically free up the RAM based PrimeOCR output memory.

pr_string_options()

```
short pr_string_options(char * command_string, char * answer);
```

Function:

Allows Prime Recognition to add custom functionality without changing the API. The actual functions performed by the string options can vary dramatically. Each call to this function can perform one option, in other words, if multiple options are desired you will need to call this function multiple times.

Required?:

Optional.

Timing:

Often the timing of this call is very important, however, the specific timing is solely based on the actual string option functionality.

Input Variables:

char * command_string.

A pointer to a string which includes all parameters of the option separated by commas. Since comma is used as a delimiter you cannot include a comma as one of the parameters.

Return Value:

char * answer

A pointer to a string which may be used to supply data back to developer. However, many string options may not use this parameter, in which case the developer may call the function with answer=NULL. If the string option does use this parameter then the developer must make sure that the length of the answer string is large enough to hold the return value.

short return from function

Some functions may also return a positive integer to the developer through the function's return value. A negative return value is always an error.

Typical Error Return Values:

- 12001 Command string was NULL
- 12002 Answer string was NULL (Only reported if non NULL answer string is required)
- 12003 Command string did not include any known commands
- 12004 Command string did not include the correct number of commands
- 12005 One or more of expected commands was blank
- 12006 One or more of expected commands was incorrect/unrecognized

Notes:

The currently supported string option calls are documented in *Chapter 5 – String Options*.

pr_pre_process_PDF()

```
short pr_pre_process_PDF(short setting, char * pdf_input_path, short * result, char * txt_output);
```

Function:

Checks a PDF file to see if the file includes searchable text. The function can also extract text from the PDF file and save the extracted text to a file.

The function will identify the file as being searchable if any page includes searchable text.

This call is totally independent of the PrimeOCR engine overall.

Required?:

Optional.

Timing:

Can be made at any time, including BEFORE the pr_open() call.

Input Variables:

short setting

0 = do not extract text from the PDF file

1 = extract text from the PDF file

char * pdf_input_path

Pointer to a string which includes the full path and filename for the PDF file to read, e.g. "d:\images\4931.pdf".

Return Value:

short * result

0 = no searchable text found

1 = searchable text found

char * txt_output

Pointer to a string which includes the full path and filename for where the searchable text should be saved to, e.g. "d:\images\4931.txt". The searchable text will only be saved to the output file if the setting variable is set to "1".

Chapter 11

Example Applications

Two sample applications are included when the Developer files are installed from the installation program:

-C Source Code Sample: PREXC32.EXE

-Visual Basic Source Code Sample: PREXVB32.EXE

Two NET variations of these programs are also included.

C Source Code Sample Application

A simple application is included in the Toolkit that interfaces to the PrimeOCR API. The sample was written in C using Microsoft Visual C++. We highly recommend that you look at the sample application early in your reading, as it will illustrate quickly how to interface, control, and interpret the output of the PrimeOCR engine.

You can easily perform a basic test of the PrimeOCR engine accuracy by modifying the path noted below to read a user supplied image.

Functionality

The sample application:

- initialize PrimeOCR engine
- loads supplied test image
- configures PrimeOCR engine
- recognizes image
- converts PrimeOCR RAM output to PrimeOCR file output
- releases PrimeOCR RAM memory
- closes PrimeOCR engine

The sample application makes almost all configuration calls for each image. This is not necessary if the configuration items do not change between images, and does slow down processing to some extent.

Files

This application includes:

PREXC32.EXE	compiled application (put in \BIN directory)
PRAPIN.C	C source code
PRAPIN.RC, PRAPIN.H RESOURCE.H, PREXC.ICO	Resource & other files
PREXCVC2.MAK	Make file for VC++ 2.0
PREXCVC5.DSP	Project file for VC++6.0
PREXC VC5.DSW	Project file for VC++6.0
TEST1.TIF, TEST2.TIF	Test images (put in \JOB directory)

Output

The example program will produce a PrimeOCR format ASCII file in the image directory, with the same file name as the image file, except with a ".PRO" extension, that includes all recognized text, confidence levels, line and character location data. (If the confidence level\character coordinate option was not purchased then only the recognized text will be output.) A text only (no line coordinates, etc.) version will be put in the same directory with a ".TXT" extension added to the image file name.

Example Source Code

The example application has virtually no user interface and is not "event" driven as a "good" Windows program should be, it is intentionally simplistic to focus attention on the PrimeOCR code. All PrimeOCR code and processing occurs within the lines noted below in the WinMain function.

```
//*****Prime Recognition Example Code Start*****  
  
PrimeOCR Code...  
  
//*****Prime Recognition Example Code End*****
```

Except for the required PrimeOCR header files, all other code outside of these lines is used to create a minimal Windows program to "host" the PrimeOCR process. The example program reports errors, if any, through a message box and terminates. Feel free to modify and use the sample application code to create an interface to the PrimeOCR engine.



Warning

During the use of a development tool such as Visual C++, set the output file name (for the application) to the \BIN directory. Otherwise, if you let the output executable default to \WINDEBUG, for example, then the engine will fail to initialize since the executable is not in the \BIN directory.

Visual Basic Source Code Sample

Overview

Visual Basic applications can access Windows based DLLs such as PrimeOCR. Located in the SAMPLES\VB directory are all source files for a working VB 6.0 application, called PREXVB32.EXE, that can process an image through the PrimeOCR engine. This example is extremely simplistic and does not attempt to use all PrimeOCR calls.

Files included are:

-PROCR32.BAS

Includes prototypes and global constants required to interface to the PrimeOCR engine DLL through VB. You must add this file to any project that will interface to the PrimeOCR engine DLL.

-PRVBEX32.FRM, PREXVB32.VBP, PREXVB32.VBW

The source code to the VB example. This source code can be modified and used in any way by the developer without royalty or other legal restriction so long as it is used with the PrimeOCR engine.

-PREXVB32.EXE

Compiled version of source code. You can run this program immediately if you installed PrimeOCR to the C:\PRDEV directory. If not you will have to edit the PRVBEX32.FRM to point to the TEST1.TIF correct path and recompile application.



Warning

When using the “Make EXE file” command in VB we recommend you create the EXE into the \BIN directory. This means that you do not have to create a PATH to \BIN.

Special Considerations of Interfacing to PrimeOCR via Visual Basic

1. You must have Visual Basic installed on your machine. We do not provide VB files.
2. Callback functions are not supported by Visual Basic therefore the callback function pointer in pr_open must be NULL (see PRVBEX32.FRM source code for an example).
3. It is difficult to return and send complex data structures in Visual Basic. Therefore the calls that require the zonestructure returned from the pr_recognize_image() call: pr_del_results() and pr_convert_output_to_file(), can instead be sent the string “CURRENT RESULTS”. This will cause these calls to use the latest results, which have been kept in the engine, for their operation (see PRVBEX32.FRM source code for an example).

4. In C based example programs we often show how to manipulate the PrimeOCR output in memory by operations on the zonestructure returned from the `pr_recognize_image()` call. In Visual Basic we strongly recommend that you instead use the `pr_output_xxx()` calls to get output in memory.

5. A small number of non-critical calls have not been tested under VB. These are highlighted in the PROCR32.BAS file.

6. VB has improved significantly over time. We recommend strongly that you use VB 6.0 or VB NET for your development.

Other Source Code Samples

Other samples are provided in C#, VB.NET, and C++.NET (in some cases with VS2003 and VS2005 versions).

Chapter 12

OCR Job Server as a Service

PrimeOCR, in the form of the OCR Job Server application, can be run as a Windows Service. The application will run with no user interface, and can be configured to start automatically when Windows starts without a user login (useful in a high security environment.)

Installation of Service

From the command prompt or the Run command, enter:

```
C:\prdev\bin\ocrjob32service.exe -Install (use your actual path)
```

Removal of Service

From the command prompt or the Run command, enter:

```
C:\prdev\bin\ocrjob32service.exe -Uninstall (use your actual path)
```

Configuration of OCR parameters

Run the OCR Job Server program (ocrjob.exe), and configure normally prior to starting PrimeOCR as a service. All settings will be used by the Service version of product, except:

- “Setup\Input\Job Directory\Once all jobs are complete setting” is ignored, it will always be set to “Continuously poll for next job” by Service version.
- PrimeOCR.ini\ShowProgressWindow and ShowErrorsDialogBox are ignored.

Configuration of Service

Stop/Start

Go to Windows Start\Control Panel\Administrative Tools\Services.

You should see PrimeOCR as one of the Services. Select PrimeOCR. Right click to bring up actions you can take on PrimeOCR service. You can start/stop, etc. from this point.

Auto Start

Select Properties via the right click. You can change PrimeOCR service to startup when Windows starts. (General tab/Startup type/Automatic).

Auto Restart

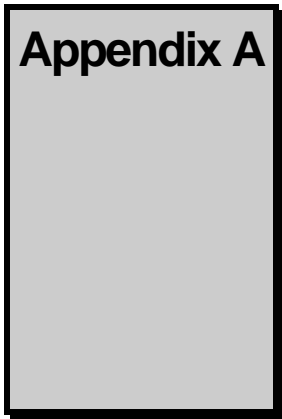
You can setup PrimeOCR service to restart if it encounters an error in the Setup\OCR Engine\Error Handling\Restart\Reboot Error List. You must have selected Setup\OCR Engine\Error Handling Restart the Job Server...”.

Go to the Recovery Tab (in the right click within Services window). Select “Restart the Service” in the “First Failure” setting. If your “Restart Service after” setting is set to 1 minute, if the PrimeOCR encounters an error in the list, it will restart the service after 1 minute has elapsed.

Error Conditions

You can not run any other instance of PrimeOCR at the same time as the Service is running. If you attempt to do so you will get –8890 errors or –4701 warnings in the log file.

Note that email notifications are not yet supported when the OCR Job Server is run as a service. If this functionality is of interest please contact Prime Recognition.



Appendix A

Recognized Characters

The PrimeOCR engine software can recognize most ANSI characters. In some cases recognized characters will be mapped to other characters. The characters recognized and the corresponding output characters are noted below.

Restricted Sets

The contents of zones can be restricted to increase accuracy and speed of OCR. Below are the PrimeOCR defined restrictions and the subset of characters that are supported in each set.

NO_RESTRICTIONS

All characters listed below

PR_ALPHA

a-z A-Z , . plus international characters

PR_NUMERIC

0-9 , . + - () \$ / @ # £

UPPERCASE (applies to **NO_RESTRICTIONS** and **PR_ALPHA**)

a-z is not allowed

LOWERCASE (applies to **NO_RESTRICTIONS** and **PR_ALPHA**)

A-Z is not allowed

Basic Character Recognition and Mapping

<u>Character</u>	<u>PrimeOCR Output</u>
0-9	0-9
a-z	a-z
A-Z	A-Z

Punctuation and Other Character Recognition and Mapping

(ANSI Numbering)

Dec	Hex	Char	PrimeOCR Output
32	20	(space)	(space)
33	21	!	!
34	22	"	"
35	23	#	#
36	24	\$	\$
37	25	%	%
38	26	&	&
39	27	'	'
40	28	((
41	29))
42	2A	*	*
43	2B	+	+
44	2C	,	,
45	2D	-	-
46	2E	.	.
47	2F	/	/
58	3A	:	:
59	3B	;	;
60	3C	<	<
61	3D	=	=

Dec	Hex	Char	PrimeOCR Output
62	3E	>	>
63	3F	?	?
64	40	@	@
91	5B	[[
92	5C	\	\
93	5D]]
94	5E	^	^
95	5F	_	_
96	60	`	`
123	7B	{	{
124	7C		
125	7D	}	}
126	7E	~	~
128	80	€	€ euro symbol
145	91	'	' (open single quote)
146	92	'	' (close single quote)
147	93	“	” (open quote)
148	94	”	” (close quote)
149	95	•	* (bullet)
150	96	-	-
151	97	—	-
163	A3	£	£
167	A7	§	§
169	A9	©	©

171	AB	«	«
173	AD	-	-
176	B0	◦	◦
182	B6	¶	¶
187	BB	»	»

International Language: International Character Recognition and Mapping

(ANSI Numbering)

Dec	Hex	Char	PrimeOCR Output
161	A1	ı	ı
191	BF	ı	ı
192	C0	À	À
193	C1	Á	Á
194	C2	Â	Â
195	C3	Ã	Ã
196	C4	Ä	Ä
197	C5	Å	Å
198	C6	Æ	Æ
199	C7	Ç	Ç
200	C8	È	È
201	C9	É	É
202	CA	Ê	Ê
203	CB	Ë	Ë
204	CC	Ì	Ì
205	CD	Í	Í

Dec	Hex	Char	PrimeOCR Output
206	CE	Î	Î
207	CF	Ï	Ï
209	D1	Ñ	Ñ
210	D2	Ò	Ò
211	D3	Ó	Ó
212	D4	Ô	Ô
213	D5	Õ	Õ
214	D6	Ö	Ö
216	D8	Ø	Ø
217	D9	Ù	Ù
218	DA	Ú	Ú
219	DB	Û	Û
220	DC	Ü	Ü
223	DF	ß	ß
224	E0	à	à
225	E1	á	á
226	E2	â	â
227	E3	ã	ã
228	E4	ä	ä
229	E5	å	å
230	E6	æ	æ
231	E7	ç	ç
232	E8	è	è
233	E9	é	é

234 EA ê ê

Dec	Hex	Char	PrimeOCR Output
235	EB	ë	ë
236	EC	ì	ì
237	ED	í	í
238	EE	î	î
239	EF	ï	ï
241	F1	ñ	ñ
242	F2	ò	ò
243	F3	ó	ó
244	F4	ô	ô
245	F5	õ	õ
246	F6	ö	ö
248	F8	ø	ø
249	F9	ù	ù
250	FA	ú	ú
251	FB	û	û
252	FC	ü	ü
255	FF	ÿ	ÿ

Asian Fonts/Characters

The following fonts are supported in the Asian languages:

Simplified Chinese: Hei, Song, Kai, FangSong

Traditional Chinese: Ming, Kai, Hei, Yuan

Korean: Batang, Myeongjo, Gothic

Japanese: MS Gothic, MS Mincho

The output is in Unicode format.

Russian Characters

The Russian characters within Unicode character set from 0401 – 044F are recognized.

Barcode

The list of barcode types is described and configured in prbarcode.ini

The output is as regular characters with associated confidence levels in memory or in the output files.

Index

6

64 bit OS, 2-2, 3-8

A

ACC template file, 4-7, 5-17, 5-18
 Accessible PDF files, 4-7, 5-18
 Accuracy, 1-4
 Accuracy level, 1-8, 4-12
 accuracy_improvement, 10-42
 Activity log, 7-3
 Admin Rights, 2-2, 7-2
 Alpha only content restrictions, 1-8
 ALPHA_ONLY, 10-41, A-1
 API Calls
 Configure page, 10-2
 Configure zone, 10-2
 Convert output, 10-2
 Custom functionality, 10-2
 Engine scope, 10-3
 Functionality, 10-5
 Functionality groups, 10-2
 Get output, 10-2
 Image scope, 10-3
 Input/Process image, 10-2
 Memory management, 10-2
 Open or close engine, 10-2
 Recognize page, 10-2
 Scope, 10-3
 Sequence, 10-4
 Zone calls, 10-3
 API error numbers, 7-3
 ASCII Defaults, 5-13
 ASCII format output, 1-10, 4-8, 6-1, 10-65
 Asian Font/Characters, A-7
 Asian Languages, 10-32
 Auto Rotate Accuracy Improvement, 5-37
 Autorotation, 4-11
 AutoZone, 4-10

B

Barcode, 1-5, 4-12, 10-41, A-7
 prbarcode.ini, 4-12
 Baseline coordinates, 6-5
 Bates Number Accuracy Improvement, 5-38
 bbox data structure, 6-14
 BIN directory, 3-1
 Binarize Color/Gray Image Output, 3-10
 Bitmap, 1-3
 BLK files, 4-4
 Bold character attribute, 6-6, 6-13
 Bounding box, 6-6, 6-11
 bbox, 6-14
 Business memo input, 1-4

C

Call return timing, 10-5
 Callback, 1-11, 10-5, 10-9
 Change Characters in Output, 5-2
 Change Strings in Output, 5-4
 char_data, 6-13
 Character attribute, 6-6, 6-11, 6-13
 Character size, 6-6, 6-11, 6-14
 Characters per line, 1-9
 Check4Text, 3-12
 CHINESE_SIMPLE, 1-5, 10-32
 CHINESE_TRADITIONAL, 1-5, 10-32
 Color image enhancement, 1-4
 Color images, 1-2, 1-3, 4-13, 10-17, 10-29
 Color/Gray Image Output, 3-10
 Comma delimited output, 6-3
 CONFID.TXT, 4-2
 Confidence, 6-10, 6-13
 Confidence level, 6-5
 Confidence log, 4-2
 Configuration
 Get multiple zones, 10-69
 Configuration memory, 10-5

Content restrictions, 1-8, 10-41
 CPU License, 2-3
 CPU Utilization, 2-3
 Customization features, 4-14
 Cut Repetitive Characters, 5-6

D

DANISH, 10-32
 DEFAULT, 10-34
 Deskew, 1-4, 4-10
 Despeckle, 1-4
 Disk space requirements, 2-1
 Distributing OCR engine, 3-19
 DLL, 1-1
 Document source, 4-9
 Dot matrix, 1-5
 DOT_MATRIX, 10-35
 DUTC, 10-32
 dynamic thresholding, 3-10

E

Ease of use, 1-11
 Email Notifications, 4-2, 12-2
 Engine training, 1-9
 Engine tuning, 1-10, 10-42
 Engine weights, 1-10, 10-42
 Error reporting, 7-2, 10-5
 Undocumented errors, 3-2, 3-3
 Example applications, 11-1

F

FASCII format output, 4-8, 10-65
 FAX, 1-4
 Input RAM images, 10-14
 Standard mode file input, 10-16
 FAX input, 1-3
 File based output, 6-8
 Font sizes, 1-5
 Font sizes accuracy, 1-3
 Font types, 1-4
 Formatted ASCII format output, 1-10, 6-2
 FRENCH, 10-32
 Full text, 1-7

G

GERMAN, 10-32
 Getting OCR output, 10-58, 10-59, 10-60, 10-61,
 10-63
 Greyscale images, 1-2, 1-3, 4-13, 10-17, 10-29

H

handle, 10-8
 Hardware key, 3-2
 USB installation, 3-3
 Hardware key errors, 3-2, 3-3
 Hardware key installation, 3-2
 Height
 Line, 6-13, 6-15
 HTML Defaults, 5-14
 HTML output, 6-3

I

Image
 Quality, 1-4
 Resolution, 1-3
 Size, 1-4, 10-29
 Image files, 1-2
 Image input, 10-14, 10-15
 Image output, 6-9, 6-12
 Image pointer, 1-3
 Image preprocess, 4-10
 Image Processing
 Deskew, 1-4
 Despeckle, 1-4
 Image quality, 4-9
 Image source, 1-4
 Image zone numbering, 4-13
 INCLUDE directory, 3-1
 Include files, 10-1
 InputAccel Module, 9-1
 Installation, 3-1
 instance, 10-8
 Insurance claim form input, 1-4
 International language, 4-9, 10-32
 International Language, A-4
 Internet address, 8-1
 ITALIAN, 10-32
 Italic character attribute, 6-6, 6-13

J

JAPANESE, 1-5, 10-32
 Japanese Font/Characters, A-7
 JBIG2 Compression, 5-18
 Job creation, 4-4
 Job file format, 4-5
 Job server, 11-4, 11-5
 Job status, 4-4
 JOB_RESULTS.LOG, 4-4

K

Korean Font/Characters, A-7

L

LAN, 1-1
 Language, 1-5, 10-54
 level, 10-42
 LEVEL_1, 10-42
 LEVEL_2, 10-42
 LEVEL_3, 10-42
 LEVEL_4, 10-42
 LEVEL_5, 10-42
 LEVEL_6, 10-42
 LEXICAL, 10-41
 Lexical check, 1-8, 4-12, 10-41
 Lexical check plus, 1-8, 1-9, 4-12
 LEXPLUSPRO.INI, 4-12, 10-54
 Libraries, 10-1
 Line
 Height, 6-13, 6-15
 Lines per zone, 1-9
 linestruct data structure, 6-13
 Location of OCR engine, 4-1
 Log File Settings - Change, 5-11
 LOW_QUALITY, 10-34
 LZW, 1-2

M

MACHINE, 10-35
 Machine print, 1-5
 Magazine input, 1-4
 Memory
 Management, 1-11, 10-68
 MICR, 1-5

Microfiche, 1-4
 Microfilm, 1-4

N

NEW_DOCUMENT, 10-22, 10-31
 Newspaper input, 1-4
 NO_APPEND, 10-66
 NO_LEXICAL, 10-41
 NO_RESTRICTIONS, 10-41, A-1
 Normal attribute, 6-6
 NORWEGIAN, 10-32
 Notes on output, 10-57
 Number of zones, 10-40
 Numeric only content restrictions, 1-8
 NUMERIC_ONLY, A-1

O

OCR Job Server as a Service, 12-1
 OCRA, 1-5
 OCRB, 1-5
 OCRJOB.LOG, 7-2
 OMR output, 6-9, 6-11
 Optimization, 5-18
 Output
 ASCII file content, 1-11
 bbox data structure, 6-14
 Character, 6-7, A-1
 Coordinate measurements, 6-5
 Data structure, 1-11
 Example, 6-15
 File based, 6-8
 File based example, 6-9
 Line level, 6-7
 linestruct data structure, 6-13
 Order, 1-8
 Pointer to, 1-11
 Zone level, 6-7
 zonestructure data structure, 6-13
 Output Point Size Change, 5-31
 Overview, 1-1

P

Page number, 4-8
 PDA Defaults, 5-27
 PDA format output, 1-10, 4-8, 6-2, 10-65

- PDF – Accessible/508 Compliance, 5-18
- PDF – Optimization, 5-18
- PDF – Thumbnails, 5-18
- PDF – Zip Compression, 5-18
- PDF 508, 6-4
- PDF Accessibility, 6-4
- PDF bookmark.ini, 6-4
- PDF bookmarks, 5-22, 10-67
- PDF Bookmarks, 6-4
- PDF deescription, 5-22
- PDF Desample, 6-4
- PDF descriptions, 6-4
- PDF docprop_pdf.ini, 6-4
- PDF format output, 1-10, 4-8, 10-65
- PDF hyperlinks, 5-22
- PDF Hyperlinks, 6-4
- PDF image input, 1-2, 3-11
- PDF initial view, 5-22
- PDF Initial view, 6-4
- PDF JBIG2, 6-4
- PDF Optimize, 6-4
- PDF output, 6-3
- PDF Output, 6-4
- PDF Password, 6-4
- PDF PDF/A, 6-4
- PDF PDF_DEFAULTS string option, 6-4
- PDF PDF_RETAIN string option, 6-4
- PDF PDF_SECURITY string option, 6-4
- PDF Restrictions, 6-4
- PDF Security, 6-4
- PDF Snippets, 6-4
- PDF Thumbnails, 6-4
- PDF Zip, 6-4
- PDF/A, 5-18
- PDF/A files, 5-17
- PDF/A format output, 1-10
- PDF_DEFAULTS, 5-17
- PDF_RETAIN, 5-22
- PDF_SECURITY, 5-24
- Pdf2Text, 3-12
- Phone books, 5-12
- Point Size Change, 5-30
- Point size, fonts, 1-5
- point_size, 6-14
- PORTUGUESE, 10-32
- PR_APPEND, 10-66
- pr_auto_rotate(), 10-2, 10-3, 10-24, 10-25
- PR_AUTOROTATE, 10-24
- pr_autozone(), 10-2, 10-3, 10-36
- PR_BARCODE, 10-41
- pr_blank_image_area(), 10-20
- pr_close(), 10-2, 10-3, 10-13
- pr_configure_ini(), 10-11
- pr_configure_lexical(), 10-54
- pr_configure_output(), 10-64
- pr_configure_zone(), 10-2, 10-3, 10-41, 10-45, 10-47
- pr_convert_output_to_file(), 10-2, 10-3, 10-65
- pr_convert_output_to_pda_memory(), 10-2, 10-3
- pr_del_results(), 7-3, 10-2, 10-3, 10-68
- pr_deskew_image(), 10-2, 10-3, 10-22
- PR_FIX_SIZE, 10-24
- pr_get_autozone(), 10-39
- pr_get_image_information(), 10-17
- pr_image_information(), 10-17
- pr_language_id(), 10-51
- PR_LAST_PAGE, 10-66
- pr_lexical_plus(), 10-55
- pr_load_page_template(), 10-2, 10-49
- pr_load_zone_template(), 10-69
- pr_open(), 10-2, 10-3, 10-8
- pr_output_char_box(), 10-63
- pr_output_line_data(), 10-61
- pr_output_line_length(), 10-60
- pr_output_xxx(), 10-2
- pr_output_zone_lines(), 10-58, 10-59
- pr_pre_process_PDF, 10-70
- pr_preprocess_image(), 10-2, 10-3, 10-24
- pr_read_image_file(), 10-2, 10-3, 10-15
- pr_read_image_RAM(), 10-2, 10-3, 10-14
- pr_recognize_image(), 10-3, 10-50
- pr_report_error(), 10-12
- pr_save_image(), 10-2, 10-3, 10-27
- pr_save_image_mpage(), 10-28
- PR_SEGMENT, 10-36
- pr_set_language(), 10-2, 10-3, 10-32
- pr_set_low_quality(), 10-2, 10-3, 10-34
- pr_set_new_document(), 10-2, 10-3, 10-31
- pr_set_number_of_zones(), 10-2, 10-3, 10-40
- pr_set_printer_type(), 10-2, 10-3, 10-35
- pr_set_user_dictionary(), 10-53
- pr_string_options(), 10-3
- PR_ZONE, 10-36
- prapi.h, 10-1
- prapin.c, 11-2
- prapin.rc, 11-2
- prbarcode.ini, 4-12, 10-41, A-7
- PREOUT.INI, 4-2

prexc.exe, 11-2
 prexc.mak, 11-2
 primage.ini, 3-10
 PrimeOCR format output, 6-2
 PrimeOCR Job Server, 4-1
 primeocr.ini, 3-5
 PrimeVerify, 1-1
 PrimeView, 1-1
 Print type, 4-9
 Printer support, 3-2
 Printer type, 1-5
 PRLEXICA.INI, 4-12
 PRO format options, 5-28
 PRO format output, 1-10, 4-8, 10-65
 PROCR.LOG, 7-2
 PROCRERR.TXT, 7-3, 10-12
 Product architecture, 1-1
 Progress feedback, 1-11
 Progress reporting, 10-5, 10-9
 prstrct.h, 10-1

R

RAM, 1-3, 1-11
 Raw bitmap, 1-3
 Reboot system, 7-1
 Resolution, 1-3
 rotate_hurdle, 10-26
 RRI3 format output, 1-10, 4-8, 6-2, 10-65
 RTF Defaults, 5-15
 RTF format output, 1-10, 4-8, 6-2, 10-65
 RUSSIAN, 10-32
 Russian Characters, A-7
 Russian Languages, 10-32, 10-33

S

SAME_DOCUMENT, 10-22, 10-31
 Save Image, 4-11
 Scanner, 1-4
 Section 508 Compliance, 4-7, 5-17, 5-18
 Service
 Ocr Job Server, 12-1
 Service, Running as, 4-5
 Simplified Chinese Font/Characters, A-7
 Software installation, 3-1
 Software key dependencies, 3-4
 Software key installation, 3-3
 Source code sample, 11-1

SPANISH, 10-32
 speed, 10-8
 Stastical measures, 5-7
 Subscript character attribute, 6-6, 6-13
 Subscript/Superscript Recognition Enhancement, 5-35
 Subscript/Superscript Tagging, 5-36
 Superscript character attribute, 6-6, 6-13
 SWEDISH, 10-32
 System requirements, 2-1

T

Template creation, 4-4
 Template file format, 4-7
 test.tif, 11-2
 Text orientation, 1-5
 Throughput, 1-2, 1-4
 Bad quality images, 10-6
 Thumbnails, 5-18
 TIFF, 1-2
 FAX, 1-2
 G3, 1-2
 G4, 1-2
 LZW, 1-2
 Traditional Chinese Font/Characters, A-7
 Troubleshooting suggestions, 7-1

U

UK, 10-32
 Underline character attribute, 6-6, 6-13
 US, 10-32
 USB hardware key, 3-3
 User Accounts, 2-3
 User Dictionaries, 1-9
 User dictionary(), 10-53
 UTF Output, 5-32
 UTF-16, 5-32
 UTF-8, 5-32

V

Variable Processing, 5-9
 Visual Basic, 10-1, 10-57
 Visual C++, 11-1

X

x64 bit OS, 2-2, 3-8
XML output, 6-3

Y

Yielding, 10-5

Z

Zip Compression, 5-18

Zone dimensions, 4-13
Zone limits, 1-9
Zone number, 1-8, 4-11
Zone restrictions, 4-12
zone_content, 10-41
zone_location, 10-43, 10-45, 10-47
zone_number, 10-41, 10-45, 10-47
Zones, 1-7
 per image, 1-9
zonestructure data structure, 6-13
Zoning tool, 4-1
ZY3 format output, 6-2
ZyIndex format output, 1-10, 4-8, 10-65