



SetaPDF-Stamper API SetaPDF-Stamper API Pro

Manual and Reference

Version 1.7.1, 2010-06-23 16:00:26

Setasign - Jan Slabon
Major-Hirst-Straße 11
38442 Wolfsburg
Germany

<http://www.setasign.de>
support@setasign.de

Table of contents

Introduction	4
System Requirements	5
Installation.....	6
Ioncube	8
Zend.....	9
Examples of Use.....	10
Constants / Configuration	15
Caching.....	21
SetaPDF	24
SetaPDF::isError()	25
SetaPDF_Error	26
SetaPDF_Parser.....	27
SetaPDF_Parser::cacheDir()	28
SetaPDF_Parser::cacheFlags().....	29
SetaPDF_Parser::cacheMkdirMode().....	30
SetaPDF_Parser::cacheNoOfObjectsPerInstance().....	31
SetaPDF_Parser::cacheHashFunction()	32
SetaPDF_Stamp.....	33
SetaPDF_Stamp::setAlpha()	34
SetaPDF_Stamp::setLink()	35
SetaPDF_Stamp::setVisibility().....	36
SetaPDF_TextStamp.....	37
SetaPDF_TextStamp::setText().....	38
SetaPDF_TextStamp::setFontFamily().....	39
SetaPDF_TextStamp::setFontStyle()	40
SetaPDF_TextStamp::setFontSize()	41
SetaPDF_TextStamp::setLineHeight()	42
SetaPDF_TextStamp::setAlign().....	43
SetaPDF_TextStamp::setRenderingMode()	44
SetaPDF_TextStamp::setOutlineWidth()	45
SetaPDF_TextStamp::setCharSpacing().....	46
SetaPDF_TextStamp::setWordSpacing()	47
SetaPDF_TextStamp::setTextWidth()	48
SetaPDF_TextStamp::setFont()	49
SetaPDF_TextStamp::setFontColor().....	51
SetaPDF_TextStamp::setOutlineColor().....	52
SetaPDF_TextStamp::setBackgroundColor().....	53

- SetaPDF_TextStamp::setDrawBackground()54
- SetaPDF_ImageStamp55
 - SetaPDF_ImageStamp::setFileName()56
 - SetaPDF_ImageStamp::setDimensions()57
 - SetaPDF_ImageStamp::setWidth()58
 - SetaPDF_ImageStamp::setHeight()59
 - SetaPDF_ImageStamp::getDimensions().....60
- SetaPDF_JPGStamp61
- SetaPDF_PNGStamp62
- SetaPDF_PdfStamp63
 - SetaPDF_PdfStamp::setFileName()65
 - SetaPDF_PdfStamp::setPageNo()66
 - SetaPDF_PdfStamp::getPageNo()67
 - SetaPDF_PdfStamp::getPageCount()68
 - SetaPDF_PdfStamp::readAllPages().....69
 - SetaPDF_PdfStamp::setDimensions()70
 - SetaPDF_PdfStamp::setWidth()71
 - SetaPDF_PdfStamp::setHeight().....72
 - SetaPDF_PdfStamp::getPageBoxes().....73
 - SetaPDF_PdfStamp::getOriginBox()74
 - SetaPDF_PdfStamp::getPageRotation()75
- SetaPDF_Stamper76
 - SetaPDF_Stamper::factory()77
 - SetaPDF_Stamper::cleanUp()78
 - SetaPDF_Stamper::setUseUpdate()79
 - SetaPDF_Stamper::addFile()80
 - SetaPDF_Stamper::addFiles().....81
 - SetaPDF_Stamper::getPageCount()82
 - SetaPDF_Stamper::getPageBoxes().....83
 - SetaPDF_Stamper::getOriginBox()84
 - SetaPDF_Stamper::getPageRotation()85
 - SetaPDF_Stamper::setStamp()86
 - SetaPDF_Stamper::updateCache().....89
 - SetaPDF_Stamper::stampit().....90
- SetaPDF_StamperPRO91
 - SetaPDF_StamperPRO::addFile().....92

SetaPDF-Stamper API - Introduction

The SetaPDF-Stamper API is a collection of PHP classes that allows PHP developers to add new content to existing PDF documents.

The implementation is designed in an intuitional way. The API offers one main class, the [SetaPDF_Stamper](#) class and some special classes which represents the stamps ([SetaPDF_TextStamp](#), [SetaPDF_PNGStamp](#), [SetaPDF_JPGStamp](#)). Stamps were created absoulutely independend of the main class and just represents the visual content of a stamp, but not positioning or rotation of it. This task is done when assigning such stamp to the main class ([SetaPDF_Stamper::setStamp\(\)](#)). When assigning a stamp object to the main class, you can f.g. defined the position and the page numbers on which the stamp should appear. It is also possible to reuse this stamp object and reassign it to different page numbers and different position or rotations.

SetaPDF-Stamper API - System Requirements

All SetaPDF APIs are written in pure PHP and does not need any other libraries installed except a PHP environment of a version later than 4.3 (until 2010) and an installed Zend Optimizer or installed Ioncube loader.

All releases since 2010 require PHP 5.

As shown in the next paragraph, it is also recommended to install MCrypt.

The SetaPDF APIs have their own integrated RC4 function for encrypting and decrypting the contents of a PDF file. For performance reasons, the APIs initially tries to use the MCrypt library, if that is installed. If MCrypt is available, the APIs require the arcfour algorithm.

The use of MCrypt increases the performance by up to 90% if encryption or decryption is needed.

If MCrypt is not available, the APIs automatically fall back to their internal RC4 function.

Depending on the file size of the PDF files to be processed, some adjustment to the php.ini directives `max_execution_time` and `memory_limit` are recommended.

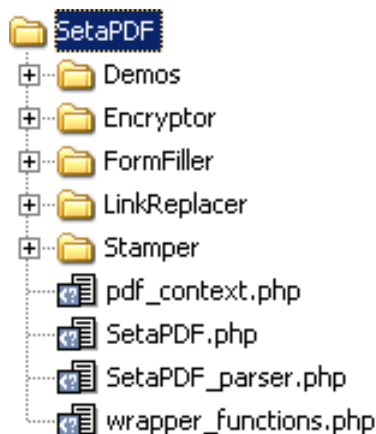
For performance optimization, all SetaPDF APIs provides a caching system that prevents the unnecessary reparsing of PDF files.

SetaPDF-Stamper API - Installation

The SetaPDF API collection includes a directory structure which should be kept, because of the internal usage of paths.

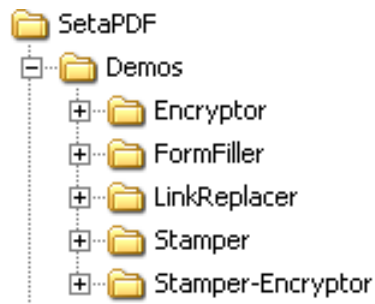
Files and directories

All packages includes a root directory called *SetaPDF*. In this directory you'll find the desired API directories. The directory structure for all current available SetaPDF APIs looks like this:



If you transfer the files via FTP make sure you use binary mode.

For each API or API combination you'll find demo files in the /Demo directory in nearly the same structure:



To use one of the SetaPDF APIs in your applications you have to add the SetaPDF-directory to your `include_path`:

```
set_include_path(get_include_path() . PATH_SEPARATOR . 'PathTo/SetaPDF/');
```

Now you can simply include the SetaPDF-Stamper API with the following line:

```
require_once("Stamper/SetaPDF_Stamper.php");
```

If you own a license for the PRO-Version you have to require the class this way:

```
require_once( "Stamper/SetaPDF_StamperPRO.php" );
```

SetaPDF-Stamper API - Ioncube encoded package

If you own a package of the API, which is encoded with Ioncube you need a loader installed on your server. There are 2 ways to get ioncube encoded files to run:

1. Install the loader in your php.ini
2. Load the loader at runtime

For details how to install ioncube or simply to check if it is installed, just download the loaders from <http://www.ioncube.com/loaders.php>, extract its content to /SetaPDF/ioncube and open the file ioncube-loader-helper.php in the directory /SetaPDF/ioncube in your webbrowser and follow the instructions. For further instructions go to <http://www.ioncube.com/>

Licensing with Ioncube

Each ioncubed package needs a valid license to run. The provided licensefiles for the SetaPDF API are named: **.htSetaPDF-<API-NAME>.icl**

You don't have to rename that file, because the package search for exactly that named file in one of its upper directories. All APIs first searches for this file in the their initial directory. F.g. The SetaPDF-Encryptor API searches first in /SetaPDF/Encryptor/. If the licensefile is not found it goes one directory upwards: /SetaPDF/ and so on...

Please notice that the filename is prefixed with .ht. Some systems hide such prefixed files automatically.

SetaPDF-Stamper API - Zend encoded package

If you own a package, which is encoded with the Zend Safeguard Suite you have to install the Zend Optimizer or Zend Guard Loader (as of PHP 5.3). For more information please go to http://www.zend.com/products/zend_optimizer.

The Zend Guard Loader (for PHP >=5.3) is actually only available as an Early Access version: <http://forums.zend.com/viewtopic.php?f=57&t=6595>

We release our products from time to time for that version. If you need a version encoded with the EA version, which is not available through your pickup depot, just send us an [email](#) and we will prepare an EA release.

Licensing with Zend

Also the zend encoded packages need valid licenses to run. The provided licensefiles for the SetaPDF API are named:

.htSetaPDF-<API-NAME>.z1

Please notice that the filename is also prefixed with .ht. Some systems hide such prefixed files automatically.

For zend encoded packages the name of the license file can be changed and has no real meaning. You can load the licensefile dynamically at runtime in your php script before you use the API:

```
$licensePath = realpath('../path/to/.htSetaPDF-  
.z1');  
zend_loader_install_license($licensePath);
```

...or change or add the license path to the following directive in your php.ini:

Zended packages are only available for development- and serverlicenses.

SetaPDF-Stamper API - Examples of Use

Simple Text-Stamp

This example creates a simple text stamp and add it to the PDF document.

```
/**
 * set the includepath for SetaPDF APIs
 * You have to point the the root directory "SetaPDF"
 */
set_include_path(get_include_path() . PATH_SEPARATOR .
realpath(dirname(__FILE__) . '/../..'));
// require the SetaPDF_Stamper-class
require_once("Stamper/SetaPDF_Stamper.php");
// Define the path, where the SetaPDF-Stamper will find the font-definition
files
define('SetaPDF_STAMPER_FONTPATH', 'Stamper/font/');
// Create a new instance of SetaPDF_Stamper with output method "Inline"
$stamper =& SetaPDF_Stamper::factory('I');
// Add a pdf-file and define its output and use of compression
$stamper->addFile(
    array(
        "in" => "docs/License_Agreement_SetaPDF.pdf",
        "out" => "demo.pdf",
        "compression" => true
    )
);
// Let's create the text stamp
// require the SetaPDF_TextStamp-class
require_once("Stamper/SetaPDF_Stamp/SetaPDF_TextStamp.php");
// Initiate a new instance of text_stamp
$textstamp =& new SetaPDF_TextStamp();
// Set the text
$textstamp->setText("Simple Text-Stamp");
// Now we assign the text stamp to our stamper
$stamper->setStamp($textstamp);
// Let's stamp!
$stamper->stampit();
```

Watermark

This example creates a text stamp with red colored text "TOP SECRET". The stamp will appear on every page. Furthermore the stamp will be rotated about 60 degrees and the opacity will be set to 10%.

```
/**
 * set the includepath for SetaPDF APIs
 * You have to point the the root directory "SetaPDF"
 */
set_include_path(get_include_path() . PATH_SEPARATOR .
realpath(dirname(__FILE__).'../../..'));
// require the SetaPDF_Stamper-class
require_once("Stamper/SetaPDF_Stamper.php");
// Define the path, where the SetaPDF-Stamper will find the font-definition
files
define('SetaPDF_STAMPER_FONTPATH', 'Stamper/font/');
// Create a new instance of SetaPDF_Stamper with output method "Inline"
$stamper =& SetaPDF_Stamper::factory('I');
$stamper->addFile(
    array(
        "in" => "docs/License_Agreement_SetaPDF.pdf",
        "out" => "demo.pdf",
        "compression" => true
    )
);
// Let's create the text stamp
// require the SetaPDF_TextStamp-class
require_once("Stamper/SetaPDF_Stamp/SetaPDF_TextStamp.php");
// Initiate a new instance of text_stamp
$textstamp =& new SetaPDF_TextStamp();
// Set the text
$textstamp->setText("TOP SECRET");
// Set font definitions
$textstamp->setFont("Arial", "B", 90);
// Set text color to red
$textstamp->SetFontColor(255,0,0);
// Set the charspacing
$textstamp->setCharSpacing(15);
// Set the rendering mode
$textstamp->setRenderingMode(2);
// Set the opacity
$textstamp->setAlpha(0.1);
/**
 * Now we assign the text stamp to our stamper
 * - we define the position on the center and middle of a page (CM)
 * - the stamp should appear on "all" pages
 * - no translation of the x-axis nor the y-axis will be done
 * - we rotate the stamp about 60 degree
```

```
*/
$stamper->setStamp($textstamp,"CM", "all", 0, 0, 60);
// Let's stamp!
$stamper->stampit();
```

Companylogo and text on different pages

The following example will create an image stamp and a text stamp, which will be positioned in the right bottom corner. The image stamp will be used on odd pages, while the text stamp will be used on all pages (or better even and odd).

The example uses a PNG file. So it uses the SetaPDF_PNGStamp class.

```
/**
 * set the includepath for SetaPDF APIs
 * You have to point to the root directory "SetaPDF"
 */
set_include_path(get_include_path() . PATH_SEPARATOR .
realpath(dirname(__FILE__).'../../'));
// require the SetaPDF_Stamper-class
require_once("Stamper/SetaPDF_Stamper.php");
// Define the path, where the SetaPDF-Stamper will find the font-definition
files
define('SetaPDF_STAMPER_FONTPATH', 'Stamper/font/');
// Create a new instance of SetaPDF_Stamper with output method "Inline"
$stamper =& SetaPDF_Stamper::factory('I');
$stamper->addFile(
    array(
        "in" => "docs/License_Agreement_SetaPDF.pdf",
        "out" => "demo.pdf",
        "compression" => true
    )
);
// require the SetaPDF_PNGStamp-class
require_once("Stamper/SetaPDF_Stamp/SetaPDF_ImageStamp/SetaPDF_PNGStamp.php");
// initiate a new instance of png_stamp
$logo =& new SetaPDF_PNGStamp();
// Set the image-path
$logo->setFileName("docs/setasign_logo.png");
// Set the image-width
$logo->setWidth(100);
/**
 * Now we assign the image-stamp to our stamper
 * - we define the position on the right bottom of each page (RB)
```

```
* - the stamp should appear only on "odd" pages
* - and translate the position about -5 dots
*/
$stamper->setStamp($logo, "RB", "odd", -5, -5);
require_once("Stamper/SetaPDF_Stamp/SetaPDF_TextStamp.php");
// Initiate a new instance of text_stamp
$textstamp =& new SetaPDF_TextStamp();
// Set text with more lines
$textstamp->setText("Stamped by\nSetaPDF-Stamper API\n".chr(169)." Setasign");
// Set Fontdefinitions
$textstamp->setFont("Arial", "", 10, null, "R");
/**
 * Now we assign the text stamp to our stamper
 * - we define the position on the right bottom of each page (RB)
 * - the stamp should appear only on "odd" pages
 * - translation of the x-axis will be -110 and the y-axis will be -5
 */
$stamper->setStamp($textstamp,"RB", "odd", -110, -5);
/**
 * Now we reuse the text stamp and assign it to the "even" pages
 * while we use another translation.
 */
$stamper->setStamp($textstamp,"RB", "even", -5, -5);

$stamper->stampit();
```

Don't update but create the document from scratch

```
/**
 * same as demol.php but we'll not just update the PDF but rewrite it from
 * scratch.
 */
set_include_path(get_include_path() . PATH_SEPARATOR .
realpath(dirname(__FILE__).'../../..'));
require_once("Stamper/SetaPDF_Stamper.php");
define('SetaPDF_STAMPER_FONTPATH', 'Stamper/font/');
$stamper =& SetaPDF_Stamper::factory('I');
/**
 * NEW: Define that the document will not be updated but recreated at all.
 * This feature makes it much more complicated to remove the stamped content
 * from the new document. But (of course) it is also slower.
 */
$stamper->setUseUpdate(false);
```

```
$stamper->addFile(
    array(
        "in" => "docs/License_Agreement_SetaPDF.pdf",
        "out" => "demo.pdf",
        "compression" => true
    )
);
require_once("Stamper/SetaPDF_Stamp/SetaPDF_TextStamp.php");
$textstamp =& new SetaPDF_TextStamp();
$textstamp->setText("This simple Text-Stamp is not just an update but the
document is completely rewritten.");
/**
 * We want the stamp to be centered on top of each
 * "odd" page.
 */
$stamper->setStamp($textstamp, "CT", "odd");
$stamper->stampit();
```

SetaPDF-Stamper API - Constants / Configuration

The API needs some constants which are hard coded into the API or have to be defined by you. There is also a constant that requires definition prior to first use.

Also you can define global variables which affects the behaviour of specific tasks.

Global Configuration Variables

`$GLOBALS['SETAPDF_PARSE_INVALID_FILES']` (boolean)

(DEPRECATED) If this global variable is set the pdf parser tries to read/repair invalid PDF documents. This setting could affect the processtime on huge files very much.

This variable isn't used by the parser as of version 1.3 (all current version of the SetaPDF APIs)

`$GLOBALS['SETAPDF_SEARCH_FOR_XREF_OFFSET']` (integer)

With this global variable you can adjust the offset position from which the pdf parser should search for the pointer to the xref table. If not defined the default value of 1500 is used.

The pdf specification says it has to be in the last 1024 bytes of a file. But sometimes there are errorious document in the wild that have some garbage at the end so we need the possibility to do a kind of finetuning for them.

Predefined Version Constants

The following constants defines the versions of specific files of the SetaPDF core:

`SETAPDF_CORE_VERSION` (string)1.3

Version of the abstract SetaPDF class. Defined in /SetaPDF/SetaPDF.php

`SETAPDF_PARSER_VERSION` (string)1.3

Version of the SetaPDF_Parser class. Defined in /SetaPDF/SetaPDF_parser.php

`SETAPDF_PDF_CONTEXT_VERSION` (string)1.3

Version of the pdf_context class. Defined in /SetaPDF/pdf_context.php

`SETAPDF_WRAPPER_FUNCTIONS_VERSION` (string)1.2.1

Version of the wrapper functions file. Defined in /SetaPDF/wrapper_functions.php

Constants to define

`SetaPDF_STAMPER_CACHEPATH` (string)

The constant `SetaPDF_STAMPER_CACHEPATH` has to contain a path to which the PHP process can

write and read. This constant **must** be set as soon as the caching system is activated. Otherwise the caching system will not be used.

As of version 1.6 a new global caching function exists and the API own functionality will be removed in coming version.

SetaPDF_STAMPER_FONTPATH (string)

Defines the path where the API will find the fontdefinition files.

Predefined Constants for Errorhandling

Possible errorcodes for the SetaPDF main class starts at 1 and ends at 99.

E_SETAPDF_CANNOT_OPEN_FILE (integer)1

cannot open XXXX !

E_SETAPDF_UNABLE_TO_POINT_TO_XREF_TABLE (integer)2

Unable to find pointer to xref table

E_SETAPDF_UNABLE_TO_FIND_XREF (integer)3

Unable to find xref table - Maybe a Problem with 'auto_detect_line_endings'

E_SETAPDF_UNEXPECTED_HEADER_IN_XREF_TABLE (integer)4

Unexpected header in xref table

E_SETAPDF_UNEXPECTED_DATA_IN_XREF_TABLE (integer)5

Unexpected data in xref table

E_SETAPDF_FILE_IS_ENCRYPTED (integer)6

File is encrypted!

E_SETAPDF_WRONG_TYPE (integer)7

Wrong Type of Element

E_SETAPDF_UNABLE_TO_FIND_OBJECT (integer)8

Unable to find object at expected location

E_SETAPDF_ENC_UNSUPPORTED_FILTER (integer)9

E_SETAPDF_ENC_UNSUPPORTED_ALGO (integer)10

E_SETAPDF_ENC_UNSUPPORTED_REVISION (integer)11

E_SETAPDF_ENC_NO_RIGHTS_FOR_SPECIFIC_ACTION (integer)12

E_SETAPDF_ENC_WRONG_OWNER_PW (integer)13

E_SETAPDF_CANNOT_COPY_FILE (integer)14

Cannot copy file XXXX to YYYY

E_SETAPDF_HEADER_ALREADY_SEND (integer)15

Some data has already been output to browser, can't send PDF file

E_SETAPDF_UNABLE_TO_FIND_TRAILER (integer)16

Trailer keyword not found after xref table

E_SETAPDF_UNSUPPORTED_FILTER (integer)17

An unsupported compression filter is required.

E_SETAPDF_ZLIB_REQUIRED (integer)18

To handle /FlateDecode filter, php with zlib support is needed.

E_SETAPDF_DECOMPRESSION_ERROR (integer)19

Error while decompressing stream.

E_SETAPDF_UNABLE_TO_CREATE_CACHE_DIR (integer)20

Unable to create directories in cache directory.

API Related Predefined Constants for Errorhandling

Possible errorcodes for the SetaPDF-Stamper API starts at 200 and ends at 299.

E_SETAPDF_STMP_MULTIPLE_NOT_ALLOWED_IF_FILE (integer)200

Adding multiple files is only allowed when output is set to 'F' (File)

E_SETAPDF_STMP_FILEPARA_SHOULD_BE_ARRAY (integer)201

Parameter should be an array: array('in' => path, 'out' => path)

E_SETAPDF_STMP_PARA_SHOULD_BE_ARRAY (integer)202

Parameter should be an array

E_SETAPDF_STMP_NOSUBCLASS_OF_STAMP (integer)203

Only subclasses of stamp are allowed.

E_SETAPDF_STMP_ERR_POSITION_PARA (integer)204

Wrong position parameter

E_SETAPDF_STMP_ERR_SHOWONPAGE_PARA (integer)205

Wrong showOnPage parameter

E_SETAPDF_STMP_ERR_CACHEUPDATE_NOT_POSSIBLE (integer)206

A Cache Update is only possible if \$this->usecache is true.

E_SETAPDF_STMP_ERR_NO_FILES (integer)214

No sourcefiles added

E_SETAPDF_STMP_PARSER_NO_KIDS (integer)207

Cannot find /Kids in current /Page-Dictionary

E_SETAPDF_STMP_PARSER_WRONG_PAGE_NO (integer)230

Wrong page number

E_SETAPDF_STMP_TXTSTMP_ERR_FONTFAMILY_PARA (integer)208

Wrong fontfamily or fontstyle

E_SETAPDF_STMP_TXTSTMP_ERR_FONTSTYLE_PARA (integer)209

Wrong fontstyle

E_SETAPDF_STMP_TXTSTMP_ERR_ALIGN_PARA (integer)210

Wrong align parameter

E_SETAPDF_STMP_TXTSTMP_ERR_RMODE_PARA (integer)211

Wrong rendering mode

E_SETAPDF_STMP_TXTSTMP_ERR_FONTMFILE (integer)212

Could not include font metric file

E_SETAPDF_STMP_OWTEXTSTMP_ERR_FONTFILE (integer)224

Could not load font file

E_SETAPDF_STMP_IMGSTMP_ERR_FILE (integer)213

No such file

E_SETAPDF_STMP_JPGSTMP_ERR_FORMAT (integer)215

Not a JPEG-File

E_SETAPDF_STMP_PNGSTMP_ERR_FORMAT (integer)216

Not a PNG-File

E_SETAPDF_STMP_PNGSTMP_ERR_INCORRECT (integer)217

Incorrect PNG file

E_SETAPDF_STMP_PNGSTMP_ERR_16BIT_NOT_SUPPORTED (integer)218

16-bit depth not supported

E_SETAPDF_STMP_PNGSTMP_ERR_ALPHA_NOT_SUPPORTED (integer)219

Alpha channel not supported

E_SETAPDF_STMP_PNGSTMP_ERR_COMPRESSION (integer)220

Unknown compression method

E_SETAPDF_STMP_PNGSTMP_ERR_FILTER (integer)221

Unknown filter method

E_SETAPDF_STMP_PNGSTMP_ERR_INTERLACING (integer)222

Interlacing not supported

E_SETAPDF_STMP_PNGSTMP_ERR_PALETTE (integer)223

Missing palette

E_SETAPDF_STMP_IMGSTMP_ERR_NO_FILE (integer)229

No image defined.

E_SETAPDF_STMP_PDFSTMP_NO_BOX_FOUND (integer)231

Box not found.

Additional predefined constant in Pro version

E_SETAPDF_STMPPRO_PARSER_ERR_ENC_FILTER (integer)225

Unsupported encryption filter: ...

E_SETAPDF_STMPPRO_PARSER_ERR_ENC_ALGO (integer)226

Unsupported encryption algorithm: ...

E_SETAPDF_STMPPRO_PARSER_ERR_REV (integer)227

Unsupported revision: ...

E_SETAPDF_STMPPRO_PARSER_ERR_WRONG_OPW (integer)228

Wrong password. Access denied.

Constans for Cache Mechanism

The following constants are used to control the behaviour of the caching mechanism of the pdf parser.

```
SETAPDF_P_CACHE_NO (integer)0x00
```

Don't read and write cache.

```
SETAPDF_P_CACHE_READ_XREF (integer)0x01
```

Try to read the cached xref table.

```
SETAPDF_P_CACHE_WRITE_XREF (integer)0x02
```

Write the xref table to cache.

```
SETAPDF_P_CACHE_XREF (integer)0x01 | 0x02
```

Try to read and write the xref table.

```
SETAPDF_P_CACHE_READ_OBJECTS (integer)0x04
```

Try to read cached objects.

```
SETAPDF_P_CACHE_WRITE_OBJECTS (integer)0x08
```

Write read objects to cache.

```
SETAPDF_P_CACHE_OBJECTS (integer)0x04 | 0x08
```

Try to read and write objects to cache.

```
SETAPDF_P_CACHE_ALL (integer)0x01 | 0x02 | 0x04 | 0x08
```

Read and write objects and xref-tables.

SetaPDF-Stamper API - Caching

PDF parsing and handling can be an expensive task in view of needed cpu-power.

To avoid doing default tasks for a single document a few times the parser class offers a caching mechanism to reduce the overhead and avoid reparsing of PDF documents a few times.

The parser simply saves serialized data in the filesystem and load them back if needed. This data can be used with ANY SetaPDF API. So if for example the [SetaPDF-Merger API](#) creates the cache data, the [SetaPDF-Stamper API](#) can benefit from them.

As of this, the handling of the cache mechanism is done through static methods of the [SetaPDF_Parser class](#). Calls to this methods will change [static variables](#) in their method contexts, so that changes doesn't depend on the object instance but applies to all instances of a parser object. (We used static variable because of compatibility to PHP4)

There are 2 parts that the parser can cache:

1. The Xref Table

This is a kind of table of contents of a PDF document. It includes information about all objects in a document and their byte-offset positions in the document. Often documents include several hundreds or thousands of entries in that table. Further more a PDF document can include more than one xref table, which relays on several updates of a document (incremental updates). But at least all tables have to be processed to get the final state of the document... By caching that data, the parser don't have to reparse the xref table out of the document.

2. Objects

Each entry in the above described xref table points to an object representing specific data, like Images, Fonts, Pages,... If the parser should read such an object it have to go to the desired byte-offset position in the document, known from the xref-table, and have to parse the object token-wise. This process needs several string comparisons and also runs recursive until the object is totally read.

The parser can cache the read objects and use the cached versions at the next situation when it is needed. No byte-position change or parsing of any string is done but simply unserializing the data from the cached data.

Usage

As already written the handling of the cache functionality is done by static methods of the [SetaPDF_Parser class](#).

You can use the static method right after including a desired API like the [SetaPDF-Merger API](#):

```
require_once( 'Merger/SetaPDF_Merger.php' );  
// at this point you can access the SetaPDF_Parser class
```

First of all you have to tell the API where you would like to save the cached data. You have to use the [SetaPDF_Parser::cacheDir\(\)](#)-method for this:

```
SetaPDF_Parser::cacheDir(realpath('../..'/path/for/cached/data/'));
```

Now you were able to activate the caching by calling the [SetaPDF_Parser::cacheFlags\(\)](#)-method with special flags. The flags are predefined in [Constants](#):

```
// Will read and write all data (xref table and objects) from/to cache.
SetaPDF_Parser::cacheFlags(SETAPDF_P_CACHE_ALL);
// Will just read and write the xref table from/to cache.
SetaPDF_Parser::cacheFlags(SETAPDF_P_CACHE_XREF);
// Will just read and write objects from/to cache.
SetaPDF_Parser::cacheFlags(SETAPDF_P_CACHE_OBJECTS);
```

After this the cache is active for all instances of any SetaPDF API.

Furthermore you can do some finetuning:

Build the cache slowly

If you want the cache to be build piecemeal you can use the [SetaPDF_Parser::cacheNoOfObjectsPerInstance\(\)](#)-method to define a maximum of objects to cache in a single script instance. With this method you can avoid performance peaks because the cache writing process, for sure, also needs cpu time.

```
// cache a maximum of 100 objects per script instance
SetaPDF_Parser::cacheNoOfObjectsPerInstance(100);
```

How is a file identified and how you can control it

By default the cache mechanism uses the [md5_file\(\)](#)-function to get an unique file identifier of the document. This file identifier is used as the directoryname in the [cache output directory](#). To give you the possibility to use another method for the fileidentification you can define your own function/method, which will be called when a fileidentifier is needed, with the [SetaPDF_Parser::cacheHashFunction\(\)](#)-method.

An Example: You already have your documents arranged in a database. This data have already unique ids related to the documents local path in your filesystem. As the ids are already known and are unique you should use the ids as a fileidentifier to avoid creating a hash with [md5_file\(\)](#).

Furthermore it is easier for you to manage the cache data, as you can for example delete the cache data if the data in the database table were deleted or changed.

The passed argument is of the pseudo-type [callback](#) and will be used with [call_user_func\(\)](#)-function.

```
function mapFilenameToId($filename) {
    // just pseudo code
    $db = YourDbClass::getInstance();
```

```
$id = $db->getOne("SELECT id FROM documents WHERE filename =  
".$db->quote($filename));  
return $id;  
}  
SetaPDF_Parser::cacheHashFunction('mapFilenameToId');
```

SetaPDF - Class

This class is the base class for nearly all SetaPDF APIs. It offers some public static helper methods.

Class Overview

SetaPDF

Child Classes

▶ [SetaPDF_Stamper](#)

Methods

▶ [SetaPDF::isError\(\)](#)

SetaPDF::isError()

Description

```
SetaPDF {  
    boolean isError ( mixed $obj )  
}
```

Determines if a variable is a SetaPDF_Error object.

Parameters

\$obj

Variable to check

Return Values

True if *\$obj* is a SetaPDF_Error object

SetaPDF_Error - Class

This class represents an error object thrown by a SetaPDF API. You can get more information about the error by checking the following properties `$obj->message` and `$obj->code`.

You can add your own error handling by defining your own class named `SetaPDF_Error` before you include any SetaPDF-File. The original class looks like this:

```
class SetaPDF_Error {
    var $message;
    var $code;

    function SetaPDF_Error($message = 'unknown error', $code = null,
                           $mode = null, $options = null, $userinfo = null) {
        $this->message = $message;
        $this->code = $code;
    }
}
```

SetaPDF_Parser - Class

The SetaPDF_Parser class is the base class for all individual SetaPDF parser classes. It is for example responsible for reading the xref table or objects of a document.

The SetaPDF_Parser class is an abstract class and *just* offers some static methods which let you control the cache functionality.

Class Overview

SetaPDF_Parser

Methods

- ◆ [SetaPDF_Parser::cacheDir\(\)](#)
- ◆ [SetaPDF_Parser::cacheFlags\(\)](#)
- ◆ [SetaPDF_Parser::cacheMkdirMode\(\)](#)
- ◆ [SetaPDF_Parser::cacheNoOfObjectsPerInstance\(\)](#)
- ◆ [SetaPDF_Parser::cacheHashFunction\(\)](#)

SetaPDF_Parser::cacheDir()

Description

```
SetaPDF_Parser {  
    mixed cacheDir ( [string $dir=null] )  
}
```

Sets the directory for cache data.

This method should be called static.

Parameters

\$dir

Path to the directory where to write the cache data. If *null* the directory will not be changed.

Return Values

The actual path.

SetaPDF_Parser::cacheFlags()

Description

```
SetaPDF_Parser {  
    mixed cacheFlags ( [string $flags=null] )  
}
```

Sets the flags how the parser should handle read and write processes of objects or xref-tables.

This method should be called static.

You can use this flags to do fine tuning of the caching mechanism. The flags can be combined using a bitwise AND (&) operation.

If any flag is set, except `SETAPDF_P_CACHE_NO`, a valid writeable path should be set with [SetaPDF_Parser::cacheDir\(\)](#).

Parameters

\$flags

The parameter defines the caching behaviour of the API. Available values are:

- ▶ `SETAPDF_P_CACHE_NO` - Don't read and write cache.
- ▶ `SETAPDF_P_CACHE_READ_XREF` - Try to read the cached xref table.
- ▶ `SETAPDF_P_CACHE_WRITE_XREF` - Write the xref table to cache.
- ▶ `SETAPDF_P_CACHE_XREF` - Try to read and write the xref table.
- ▶ `SETAPDF_P_CACHE_READ_OBJECTS` - Try to read cached objects.
- ▶ `SETAPDF_P_CACHE_WRITE_OBJECTS` - Write read objects to cache.
- ▶ `SETAPDF_P_CACHE_OBJECTS` - Try to read and write objects to cache.
- ▶ `SETAPDF_P_CACHE_ALL` - Read and write objects and xref-tables.

Return Value [\(see also Constants / Configurations\)](#)

The actual value.

SetaPDF_Parser::cacheMkdirMode()

Description

```
SetaPDF_Parser {  
    mixed cacheMkdirMode ( [integer $mode=null] )  
}
```

As the caching mechanism creates directories for each pdf document the API internally uses mkdir to create the directory. With this method you can define if and which parameter should be passed as the \$mode parameter of the [mkdir](#)-function.

This method should be called static.

Parameters

\$mode

The file mode.

The parameter consists of three octal number components specifying access restrictions for the owner, the user group in which the owner is in, and to everybody else in this order. More informations about the mode-parameter can be found [here](#).

Return Values

The actual value.

SetaPDF_Parser::cacheNoOfObjectsPerInstance()

Description

```
SetaPDF_Parser {  
    mixed cacheNoOfObjectsPerInstance ( [integer $no=null] )  
}
```

For sure a caching process needs more process power as the cached data have to be written to the file system. Often a PDF document is build with more hundres or thousands of objects which can increase the process time to a bad value.

With this method you can define how many maximum objects should be cached per script instance. So you can chop the cache creation over several script executions.

This method should be called static.

If you set the \$no-parameter, for example, to 100, the parser will cache 100 objects per script instance maximum, until all objects are cached.

By default the parser will cache ALL objects.

Parameters

\$no

The maximum number of objects to cache per instance.

Return Values

The actual value.

SetaPDF_Parser::cacheHashFunction()

Description

```
SetaPDF_Parser {  
    mixed cacheHashFunction ( [callback $hashFunction=null] )  
}
```

To identify a pdf document the API uses the [md5_file\(\)](#)-function by default.

If you want to create your own identification process or if you already know a hash or unique property of the document you can use this method to define an own function/method which will be called when the parser needs the hash.

This hash/value will be used as the directory name in the cache directory (see [SetaPDF_Parser::cacheDir\(\)](#)).

The given value will be used as the function parameter of a [call_user_func\(\)](#)-call.

This method should be called static.

Parameters

\$hashFunction

The function to be called.
(See also informations about the [callback](#) type.)

Return Values

The actual value.

SetaPDF_Stamp - Abstract Stamp Class

The class SetaPDF_Stamp is the abstract class for all a extended stamp classes. It offers some non-public and some public methods, like methods for defining transparency or link targets.

Class Overview

SetaPDF_Stamp

Child Classes

- ▶ [SetaPDF_TextStamp](#)
- ▶ [SetaPDF_ImageStamp](#)
- ▶ [SetaPDF_PdfStamp](#)

Methods

- ▶ [SetaPDF_Stamp::setAlpha\(\)](#)
- ▶ [SetaPDF_Stamp::setLink\(\)](#)
- ▶ [SetaPDF_Stamp::setVisibility\(\)](#)

SetaPDF_Stamp::setAlpha()

Description

```
SetaPDF Stamp {  
    void setAlpha ( float $alpha[, string $blendmode='Normal'] )  
}
```

With this method you can define the opacity and the blendmode of a stamp.

Parameters

\$alpha

The opacity: A value between 0 and 1, whereas 1 is defined as 100% opacity

\$blendmode

The blendmode. There are following modes available, which are described in detail in the PDF specification (7.2.4):

- ▶ Normal (*standard*)
- ▶ Multiply
- ▶ Screen
- ▶ Overlay
- ▶ Darken
- ▶ Lighten
- ▶ ColorDodge
- ▶ ColorBurn
- ▶ HardLight
- ▶ SoftLight
- ▶ Difference
- ▶ Exclusion
- ▶ Hue
- ▶ Saturation
- ▶ Color
- ▶ Luminosity

The declaration is case-sensitive.

SetaPDF_Stamp::setLink()

Description

```
SetaPDF_Stamp {  
    void setLink ( string $link )  
}
```

Creates a link annotation above the stamp.

Only in development branch - 20. September 2006

Parameters

\$link

The uniform resource identifier (URI) to resolve.

SetaPDF_Stamp::setVisibility()

Description

```
SetaPDF_Stamp {  
    boolean setVisibility ( [string $visibility='all'] )  
}
```

Sets the visibility of a stamp for printing or viewing-purpose.

Parameters

\$visibility

Follwing visibility states are available:

all

Default behaviour. The stamp appears in every state.

view

The stamp is only visible when the document is viewed interactively.

print

The stamp is only visible when the document is printed.

Return value

True if the passed parameter is an available visibility state.

False if the passed parameter is an unavailable visibility state.

Version

Available since Version 1.5

SetaPDF_TextStamp - Simple Text Stamp Class

This class represents a simple text stamp.

Class Overview

File:

SetaPDF_Stamp
└─ SetaPDF_TextStamp

Methods

- ◆ [SetaPDF_TextStamp::setText\(\)](#)
- ◆ [SetaPDF_TextStamp::setFontFamily\(\)](#)
- ◆ [SetaPDF_TextStamp::setFontStyle\(\)](#)
- ◆ [SetaPDF_TextStamp::setFontSize\(\)](#)
- ◆ [SetaPDF_TextStamp::setLineHeight\(\)](#)
- ◆ [SetaPDF_TextStamp::setAlign\(\)](#)
- ◆ [SetaPDF_TextStamp::setRenderingMode\(\)](#)
- ◆ [SetaPDF_TextStamp::setOutlineWidth\(\)](#)
- ◆ [SetaPDF_TextStamp::setCharSpacing\(\)](#)
- ◆ [SetaPDF_TextStamp::setWordSpacing\(\)](#)
- ◆ [SetaPDF_TextStamp::setTextWidth\(\)](#)
- ◆ [SetaPDF_TextStamp::setFont\(\)](#)
- ◆ [SetaPDF_TextStamp::setFontColor\(\)](#)
- ◆ [SetaPDF_TextStamp::setOutlineColor\(\)](#)
- ◆ [SetaPDF_TextStamp::setBackground-color\(\)](#)
- ◆ [SetaPDF_TextStamp::setDrawBackground\(\)](#)

Inherited Methods

Class: SetaPDF_Stamp

- ◆ [SetaPDF_Stamp::setAlpha\(\)](#)
- ◆ [SetaPDF_Stamp::setLink\(\)](#)
- ◆ [SetaPDF_Stamp::setVisibility\(\)](#)

SetaPDF_TextStamp::setText()

Description

```
SetaPDF_TextStamp extends SetaPDF_Stamp {  
    void setText ( string $text[, boolean $eval=false] )  
}
```

Sets the text of the stamp object.

Parameters

\$text

The string, that should be stamped. Single- and multiline text is allowed.

\$eval

The text will be passed through eval(), before it will be stamped into the PDF document.

SetaPDF_TextStamp::setFontFamily()

Description

```
SetaPDF_TextStamp extends SetaPDF_Stamp {  
    mixed setFontFamily ( string $fontfamily[, string $fontstyle=null] )  
}
```

Sets the fontfamily and style of the font.

Parameters

\$fontfamily

The name of a standard Adobe Type 1 font. Available font names are:

- ▶ Times
- ▶ Helvetica (Arial)
- ▶ Courier
- ▶ Symbol
- ▶ ZapfDingbats

Standard: Helvetica

\$fontstyle

The fontstyle of the given font. Possible values are:

- ▶ empty string: regular
- ▶ B: bold
- ▶ I: italic

or any combination. Bold and italic styles do not apply to Symbol and ZapfDingbats

Return value

True, if everything works as expected. A *SetaPDF_Error* object if an error occurs.

SetaPDF_TextStamp::setFontStyle()

Description

```
SetaPDF_TextStamp extends SetaPDF_Stamp {  
    mixed setFontStyle ( string $fontstyle )  
}
```

Sets the fontstyle of the font.

Parameters

\$fontstyle

The fontstyle of the given font. Possible values are:

- ▶ empty string: regular
- ▶ B: bold
- ▶ I: italic

or any combination. Bold and italic styles do not apply to Symbol and ZapfDingbats

Return value

True, if everything works as expected. A SetaPDF_Error object if an error occurs.

SetaPDF_TextStamp::setFontSize()

Description

```
SetaPDF_TextStamp extends SetaPDF_Stamp {  
    void setFontSize ( float $fontsize )  
}
```

Sets the fontsize in points.

Parameters

\$fontsize

The fontsize in points.

SetaPDF_TextStamp::setLineHeight()

Description

```
SetaPDF_TextStamp extends SetaPDF_Stamp {  
    void setLineHeight ( float $lineHeight )  
}
```

Sets the lineheight in points.

If the lineheight is not specified the API calculates it automatically.

Parameters

\$lineHeight

The lineheight in points.

SetaPDF_TextStamp::setAlign()

Description

```
SetaPDF_TextStamp extends SetaPDF_Stamp {  
    mixed setAlign ( float $align )  
}
```

The value which is passed to this method only takes affect, if the text is a multilinetext. With this method you can define the alignment of the text.

Parameters

\$align

Allows to center or align the text. Possible values are:

- ▶ L / left: left align (default value)
- ▶ C / center: center
- ▶ R / right: right align

Return value

True, if everythings works as expected. A *SetaPDF_Error* object if an error occurs.

SetaPDF_TextStamp::setRenderingMode()

Description

```
SetaPDF_TextStamp extends SetaPDF_Stamp {  
    mixed setRenderingMode ( integer $mode )  
}
```

Defines the Text Rendering Mode.

Parameters

\$mode

An integer value, defining the Text Rendering Mode. Possible values can be found in the PDF specification (5.2.3).

Return value

True, if everything works as expected. A SetaPDF_Error object if an error occurs.

SetaPDF_TextStamp::setOutlineWidth()

Description

```
SetaPDF_TextStamp extends SetaPDF_Stamp {  
    void setOutlineWidth ( float $outlinewidth )  
}
```

Defines the outlinewidth if a specific Text Rendering Mode is chosen.

Parameters

\$outlinewidth

The outlinewidth in points.

SetaPDF_TextStamp::setCharSpacing()

Description

```
SetaPDF_TextStamp extends SetaPDF_Stamp {  
    void setCharSpacing ( float $charSpacing )  
}
```

Sets the character spacing.

Parameters

\$charSpacing

The space between the characters in points. (default: 0)

SetaPDF_TextStamp::setWordSpacing()

Description

```
SetaPDF_TextStamp extends SetaPDF_Stamp {  
    void setWordSpacing ( float $wordSpacing )  
}
```

Sets the spacing between words.

Parameters

\$wordSpacing

The space between words in points. (default: 0)

SetaPDF_TextStamp::setTextWidth()

Description

```
SetaPDF_TextStamp extends SetaPDF_Stamp {  
    void setTextWidth ( float $textWidth )  
}
```

This method can be used to calculate the fontsize by a given width. This method only takes affect if the fontsize is previously (or afterwards) set to -1.

Parameters

\$textWidth

The width of the text in points.

SetaPDF_TextStamp::setFont()

Description

```
SetaPDF_TextStamp extends SetaPDF_Stamp {  
    mixed setFont ( string $fontfamily[, string $fontstyle='', float  
        $fontsize=12[, float $lineHeight=null[, string $align='L'[, integer  
        $renderingMode=0[, float $outlinewidth=1[, float $charSpacing=0[, float  
        $wordSpacing=0]]]]]]]] )  
}
```

This method is a kind of wrapper for defining all text layout settings within one method-call.

Parameters

\$fontfamily

The name of a standard Adobe Type 1 font. Available font names are:

- ▶ Times
- ▶ Helvetica (Arial)
- ▶ Courier
- ▶ Symbol
- ▶ ZapfDingbats

Standard: Helvetica

\$fontstyle

The fontstyle of the given font. Possible values are:

- ▶ empty string: regular
- ▶ B: bold
- ▶ I: italic

or any combination. Bold and italic styles do not apply to Symbol and ZapfDingbats

\$fontsize

The fontsize in points.

\$lineHeight

The lineheight in points.

\$align

Allows to center or align the text. Possible values are:

- ▶ L: left align (default value)
- ▶ C: center
- ▶ R: right align

\$renderingMode

An integer value, defining the Text Rendering Mode. Possible values can be found in the PDF specification (5.2.3).

\$outlinewidth

The outlinewidth in points.

\$charSpacing

The space between the characters in points.

\$wordSpacing

The space between words in points.

Return value

True, if everythings works as expected. A `SetaPDF_Error` object if an error occurs.

SetaPDF_TextStamp::setFontColor()

Description

```
SetaPDF_TextStamp extends SetaPDF_Stamp {  
    void setFontColor ( integer $g | $r | $c[, integer $g | $m[, integer $b  
        | $y[, integer $k]]) )  
}
```

Defines the color used for text. It can be expressed in RGB components, CMYK or gray scale.

Parameters

\$g / \$r / \$c

If \$g and \$b are given (3 parameter passed), red component; if \$m, \$y and \$k are given (4 parameter passed) it's the cyan component; if only one parameter is passed it indicates the gray level.

Values for rgb and gray are between 0 and 255.

For CMYK the values are between 0 and 100.

\$g / \$m

Green component (between 0 and 255) if 3 parameters were passed.

Or magenta component (between 0 and 100) if 4 parameters were passed.

\$b / \$y

Blue component (between 0 and 255) if 3 parameters were passed.

Or yellow component (between 0 and 100) if 4 parameters were passed.

\$k

Key/black component (between 0 and 100) if 4 parameters were passed.

SetaPDF_TextStamp::setOutlineColor()

Description

```
SetaPDF_TextStamp extends SetaPDF_Stamp {  
    void setOutlineColor ( integer $g | $r | $c[, integer $g | $m[, integer  
        $b | $y[, integer $k]] ] )  
}
```

Defines the color used for text outlines. It can be expressed in RGB components, CMYK or gray scale.

Parameters

\$g / *\$r* / *\$c*

If *\$g* and *\$b* are given (3 parameter passed), red component; if *\$m*, *\$y* and *\$k* are given (4 parameter passed) it's the cyan component; if only one parameter is passed it indicates the gray level.

Values for rgb and gray are between 0 and 255.

For CMYK the values are between 0 and 100.

\$g / *\$m*

Green component (between 0 and 255) if 3 parameters were passed.

Or magenta component (between 0 and 100) if 4 parameters were passed.

\$b / *\$y*

Blue component (between 0 and 255) if 3 parameters were passed.

Or yellow component (between 0 and 100) if 4 parameters were passed.

\$k

Key/black component (between 0 and 100) if 4 parameters were passed.

SetaPDF_TextStamp::setBackgroundColor()

Description

```
SetaPDF_TextStamp extends SetaPDF_Stamp {  
    void setBackgroundColor ( integer $g | $r | $c[, integer $g | $m[,  
        integer $b | $y[, integer $k]]) )  
}
```

Defines the color used as background. It can be expressed in RGB components, CMYK or gray scale.

If this method is called it'll automatically call [SetaPDF_TextStamp::setDrawBackground\(true\)](#).

Parameters

\$g | \$r | \$c

If ***\$g*** and ***\$b*** are given (3 parameter passed), red component; if ***\$m***, ***\$y*** and ***\$k*** are given (4 parameter passed) it's the cyan component; if only one parameter is passed it indicates the gray level.

Values for rgb and gray are between 0 and 255.

For CMYK the values are between 0 and 100.

\$g | \$m

Green component (between 0 and 255) if 3 parameters were passed.

Or magenta component (between 0 and 100) if 4 parameters were passed.

\$b | \$y

Blue component (between 0 and 255) if 3 parameters were passed.

Or yellow component (between 0 and 100) if 4 parameters were passed.

\$k

Key/black component (between 0 and 100) if 4 parameters were passed.

SetaPDF_TextStamp::setDrawBackground()

Description

```
SetaPDF_TextStamp extends SetaPDF_Stamp {  
    void setDrawBackground ( [boolean $drawbackground=true] )  
}
```

With this method you can control the drawing of a background rect.

Parameters

\$drawbackground

If you pass *true* the stamp a background rect will lay under the text stamp. If you pass *false* the background rect will not be putted out.

SetaPDF_ImageStamp - Abstract class for image stamps

This class is an abstract class for image handling and offers some standard methods needed by different image stamp classes.

Class Overview



Child Classes

- ▶ [SetaPDF_JPGStamp](#)
- ▶ [SetaPDF_PNGStamp](#)

Methods

- ▶ [SetaPDF_ImageStamp::setFileName\(\)](#)
- ▶ [SetaPDF_ImageStamp::setDimensions\(\)](#)
- ▶ [SetaPDF_ImageStamp::setWidth\(\)](#)
- ▶ [SetaPDF_ImageStamp::setHeight\(\)](#)
- ▶ [SetaPDF_ImageStamp::getDimensions\(\)](#)

Inherited Methods

Class: [SetaPDF_Stamp](#)

- ▶ [SetaPDF_Stamp::setAlpha\(\)](#)
- ▶ [SetaPDF_Stamp::setLink\(\)](#)
- ▶ [SetaPDF_Stamp::setVisibility\(\)](#)

SetaPDF_ImageStamp::setFileName()

Description

```
SetaPDF_ImageStamp extends SetaPDF_Stamp {  
    mixed setFileName ( string $filename )  
}
```

Sets the path to the image file. Only local files are allowed.

Parameters

\$filename

The path to the image file.

Return value

True - if the file could be read and interpreted - an SetaPDF_Error object if an error occurs.

SetaPDF_ImageStamp::setDimensions()

Description

```
SetaPDF_ImageStamp extends SetaPDF_Stamp {  
    mixed setDimensions ( [float $width=null[, float $height=null[, boolean  
        $aspectRatio=false]] )  
}
```

Sets the width and/or height of an image.

Parameters

\$width

The width of the image in points. If *null*, it is automatically calculated.

\$height

The height of the image in points. If *null*, it is automatically calculated.

\$aspectRatio

Keep aspect ratio.

Return value

An array holding the new dimensions of the image. The key 'w' holds the width and the key 'h' holds the height.

SetaPDF_ImageStamp::setWidth()

Description

```
SetaPDF ImageStamp extends SetaPDF Stamp {  
    mixed setWidth ( float $width[, boolean $aspectRatio=true] )  
}
```

An alias for `SetaPDF_ImageStamp::setDimensions($width, null, $aspectRatio)`

Parameters

\$width

The width in points.

\$aspectRatio

Keep aspect ratio.

Return value

An array holding the new dimensions of the image. The key 'w' holds the width and the key 'h' holds the height.

SetaPDF_ImageStamp::setHeight()

Description

```
SetaPDF_ImageStamp extends SetaPDF_Stamp {  
    mixed setHeight ( float $height[, boolean $aspectRatio=true] )  
}
```

An alias for `SetaPDF_ImageStamp::setDimensions(null, $height, $aspectRatio)`

Parameters

\$height

The height in points.

\$aspectRatio

Keep aspect ratio.

Return value

An array holding the new dimensions of the image. The key 'w' holds the width and the key 'h' holds the height.

SetaPDF_ImageStamp::getDimensions()

Description

```
SetaPDF_ImageStamp extends SetaPDF_Stamp {  
    array getDimensions ( void )  
}
```

Returns the current image dimensions.

Return value

An array holding the dimensions of the image. The key 'w' holds the width and the key 'h' holds the height.

SetaPDF_JPGStamp - Class

This class represents an image stamp based on a JPEG file.

Class Overview

File:



Inherited Methods

Class: SetaPDF_ImageStamp

- ◆ SetaPDF_ImageStamp::setFileName()
- ◆ SetaPDF_ImageStamp::setDimensions()
- ◆ SetaPDF_ImageStamp::setWidth()
- ◆ SetaPDF_ImageStamp::setHeight()
- ◆ SetaPDF_ImageStamp::getDimensions()

Class: SetaPDF_Stamp

- ◆ SetaPDF_Stamp::setAlpha()
- ◆ SetaPDF_Stamp::setLink()
- ◆ SetaPDF_Stamp::setVisibility()

SetaPDF_PNGStamp - Class

This class represents an image stamp based on a PNG file.

Class Overview

File:



Inherited Methods

Class: SetaPDF_ImageStamp

- ◆ SetaPDF_ImageStamp::setFileName()
- ◆ SetaPDF_ImageStamp::setDimensions()
- ◆ SetaPDF_ImageStamp::setWidth()
- ◆ SetaPDF_ImageStamp::setHeight()
- ◆ SetaPDF_ImageStamp::getDimensions()

Class: SetaPDF_Stamp

- ◆ SetaPDF_Stamp::setAlpha()
- ◆ SetaPDF_Stamp::setLink()
- ◆ SetaPDF_Stamp::setVisibility()

SetaPDF_PdfStamp - Class

This class represents a stamp object based on a page of another existing PDF document.

To avoid an unneeded overhead, the class holds the parser objects in a static variable. The class automatically only touches a single parser if different pages of a single document were needed in different stamp objects. Also internal PDF objects were shared among the stamp instances.

Some things you should keep in mind:

The Page Tree

If you want to stamp several pages of a PDF document, you should initiate the stamp object with the biggest page number at first. This will make sure that the page tree of the original document will only be read once.

This is because the parser simply reads the page tree until it finds the needed page. This process will start at the root entry of the page tree. An example: Your first stamp is based on page 1. The lookup in the page tree will stop at page 1. After that you initiate a second stamp which is based on page 2 the lookup will find page 1 and 2. Then you initiate a third stamp which is based on page 3 the lookup will find page 1,2 and 3. If you would start with page 3 all other pages are already known for the other stamp objects.

Dynamic Content

The stamp class will only use the content stream of a page for the appearance of the stamp. Dynamic content like links, form fields or any other annotation will not be used.

Class Overview

File:

```
SetaPDF_Stamp
└─ SetaPDF_PdfStamp
```

Methods

- ◆ [SetaPDF_PdfStamp::setFileName\(\)](#)
- ◆ [SetaPDF_PdfStamp::setPageNo\(\)](#)
- ◆ [SetaPDF_PdfStamp::getPageNo\(\)](#)
- ◆ [SetaPDF_PdfStamp::getPageCount\(\)](#)
- ◆ [SetaPDF_PdfStamp::readAllPages\(\)](#)
- ◆ [SetaPDF_PdfStamp::setDimensions\(\)](#)
- ◆ [SetaPDF_PdfStamp::setWidth\(\)](#)
- ◆ [SetaPDF_PdfStamp::setHeight\(\)](#)

- ▶ [SetaPDF_PdfStamp::getPageBoxes\(\)](#)
- ▶ [SetaPDF_PdfStamp::getOriginBox\(\)](#)
- ▶ [SetaPDF_PdfStamp::getPageRotation\(\)](#)

Inherited Methods

Class: [SetaPDF_Stamp](#)

- ▶ [SetaPDF_Stamp::setAlpha\(\)](#)
- ▶ [SetaPDF_Stamp::setLink\(\)](#)
- ▶ [SetaPDF_Stamp::setVisibility\(\)](#)

SetaPDF_PdfStamp::setFileName()

Description

```
SetaPDF PdfStamp extends SetaPDF Stamp {  
    mixed setFileName ( string $filename[, integer $pageNo=1] )  
}
```

Sets the path to the pdf file. Only local files are allowed.

Additionally you can define which page you want to use with this stamp object.

Parameters

\$filename

The path to the image file.

\$pageNo

The page number

Return value

True - if the file could be read and interpreted - an SetaPDF_Error object if an error occurs.

Version

as of version 1.7

SetaPDF_PdfStamp::setPageNo()

Description

```
SetaPDF PdfStamp extends SetaPDF Stamp {  
    mixed setPageNo ( string $pageNo )  
}
```

Sets the page to use for the appearance.

Parameters

\$pageNo

The page number which should be used as the appearance.

Return value

True - if everything works as expected - an SetaPDF_Error object if an error occurs.

Version

as of version 1.7

SetaPDF_PdfStamp::getPageNo()

Description

```
SetaPDF PdfStamp extends SetaPDF Stamp {  
    integer getPageNo ( void )  
}
```

Returns the page number which will be used as the appearance of the stamp.

Return value

The page number.

Version

as of version 1.7

SetaPDF_PdfStamp::getPageCount()

Description

```
SetaPDF PdfStamp extends SetaPDF Stamp {  
    mixed getPageCount ( void )  
}
```

Returns the page count of the source document.

Return value

The page count or a SetaPDF_Error object if an error occurs.

Version

as of version 1.7

SetaPDF_PdfStamp::readAllPages()

Description

```
SetaPDF PdfStamp extends SetaPDF Stamp {  
    mixed readAllPages ( void )  
}
```

This method reads in all pages of the source document.

You should use this method if you want to use all pages of a document with different stamp objects (the parser will be the same for all stamp objects).

If you don't call this method it could be that the parser has to reread the page tree of the document a few times. To avoid this, you can force the parser to read all pages at once.

Return value

True or a SetaPDF_Error object if an error occurs.

Version

as of version 1.7

SetaPDF_PdfStamp::setDimensions()

Description

```
SetaPDF PdfStamp extends SetaPDF Stamp {  
    mixed setDimensions ( [float $width=null[, float $height=null[, boolean  
        $aspectRatio=false]] )  
}
```

Sets the width and/or height of the stamp.

Parameters

\$width

The width of the stamp in points. If *null*, it is automatically calculated.

\$height

The height of the stamp in points. If *null*, it is automatically calculated.

\$aspectRatio

Keep aspect ratio.

Return value

An array holding the new dimensions of the stamp. The key 'w' holds the width and the key 'h' holds the height.

SetaPDF_PdfStamp::setWidth()

Description

```
SetaPDF PdfStamp extends SetaPDF Stamp {  
    mixed setWidth ( float $width[, boolean $aspectRatio=true] )  
}
```

An alias for [SetaPDF PdfStamp::setDimensions\(\\$width, null, \\$aspectRatio\)](#)

Parameters

\$width

The width in points.

\$aspectRatio

Keep aspect ratio.

Return value

An array holding the new dimensions of the stamp. The key 'w' holds the width and the key 'h' holds the height.

SetaPDF_PdfStamp::setHeight()

Description

```
SetaPDF PdfStamp extends SetaPDF Stamp {  
    mixed setHeight ( float $height[, boolean $aspectRatio=true] )  
}
```

An alias for [SetaPDF_PdfStamp::setDimensions\(null, \\$height, \\$aspectRatio\)](#)

Parameters

\$height

The height in points.

\$aspectRatio

Keep aspect ratio.

Return value

An array holding the new dimensions of the stamp. The key 'w' holds the width and the key 'h' holds the height.

SetaPDF_PdfStamp::getPageBoxes()

Description

```
SetaPDF PdfStamp extends SetaPDF Stamp {  
    mixed getPageBoxes ( [integer $pageNo=null[, string $fileName=null]] )  
}
```

Gets the page boxes of a PDF document.

Parameters

\$pageNo

The page number

\$fileName

The path of the PDF file. If not set, the document which is set in the [setFileName\(\)](#)-method is taken.

Return value

An array of boxes, while the keys are the box-names (f.g. /MediaBox, /TrimBox,...).

Each box has following entries:

- ◆ x: the abscissa
- ◆ y: ordinate
- ◆ w: the width
- ◆ h: the height
- ◆ llx: lower left abscissa
- ◆ lly: lower left ordinate
- ◆ urx: upper right abscissa
- ◆ ury: upper right ordinate

If an error occurs this method will return a [SetaPDF_Error-Object](#).

Version

as of version 1.7

SetaPDF_PdfStamp::getOriginBox()

Description

```
SetaPDF PdfStamp extends SetaPDF Stamp {  
    mixed getOriginBox ( integer $pageNo[, mixed $fileName=null] )  
}
```

Gets the origin page box. This will be the /CropBox or the /MediaBox. The method will check the existence of the desired box for you.

Width and height will be calculated automatically in relation to the rotation of the page.

Parameters

\$pageNo

The page number

\$fileName

The path of the PDF file. If not set, the document which is set in the [factory\(\)](#)-method or by [setSourceFileName\(\)](#) is taken.

Return Values

An array with the following entries:

- ◆ x: the abscissa
- ◆ y: ordinate
- ◆ w: the width
- ◆ h: the height
- ◆ llx: lower left abscissa
- ◆ lly: lower left ordinate
- ◆ urx: upper right abscissa
- ◆ ury: upper right ordinate

If an error occurs this method will return a [SetaPDF_Error-Object](#).

Version

as of version 1.7

SetaPDF_PdfStamp::getPageRotation()

Description

```
SetaPDF PdfStamp extends SetaPDF Stamp {  
    mixed getPageRotation ( integer $pageNo[, mixed $fileName=null] )  
}
```

Gets the number of degrees by which the page is rotated.

Parameters

\$pageNo

The page number

\$fileName

The path of the PDF file. If not set, the document which is set in the [setFileName\(\)](#)-method is taken.

Return Values

An integer value of a factor 90 or false if no value is given.

If an error occurs this method will return a [SetaPDF_Error](#)-Object.

Version

as of version 1.7

SetaPDF_Stamper - The main API class

This class is the main class of the API. It manages the whole stamping part.

Class Overview



Child Classes

▶ [SetaPDF_StamperPRO](#)

Methods

- ▶ [SetaPDF_Stamper::factory\(\)](#)
- ▶ [SetaPDF_Stamper::cleanUp\(\)](#)
- ▶ [SetaPDF_Stamper::setUseUpdate\(\)](#)
- ▶ [SetaPDF_Stamper::addFile\(\)](#)
- ▶ [SetaPDF_Stamper::addFiles\(\)](#)
- ▶ [SetaPDF_Stamper::getPageCount\(\)](#)
- ▶ [SetaPDF_Stamper::getPageBoxes\(\)](#)
- ▶ [SetaPDF_Stamper::getOriginBox\(\)](#)
- ▶ [SetaPDF_Stamper::getPageRotation\(\)](#)
- ▶ [SetaPDF_Stamper::setStamp\(\)](#)
- ▶ [SetaPDF_Stamper::stampit\(\)](#)

Inherited Methods

Class: [SetaPDF](#)

- ▶ [SetaPDF::isError\(\)](#)

SetaPDF_Stamper::factory()

Description

```
SetaPDF_Stamper extends SetaPDF {  
    mixed factory ( [string $dest='F'[, boolean $stream=false]] )  
}
```

This method has to be called static and will return an instance of the SetaPDF_Stamper class or an SetaPDF_Error object.

Parameters

\$dest

Defines how the resulting documents are handled:

- ▶ "F" saves the file to the file system
- ▶ "D" the file will be send to the client with a download dialogue (not possible in batch mode)
- ▶ "I" the file will be displayed in the client's browser window. (not possible in batch mode)

\$stream

This parameter is only used if *\$dest* is set to "D" or "I". If it is set to *true*, the document will be sent immediately as soon as the first content bytes are available. In this case the length-header will not be sent. If this parameter is set to *false*, the whole document is held in memory until it is completely assembled.

The streaming facility is very effective, because the client does not become aware of any script processing time.

Return Values

In case of success you get a new instance of the SetaPDF_Stamper class.

On failure an SetaPDF_Error object will be returned. It is strongly recommended to check this return value with SetaPDF::isError().

SetaPDF_Stamper::cleanUp()

Description

```
SetaPDF_Stamper extends SetaPDF {  
    void cleanUp ( void )  
}
```

This method will close all opened parser objects, remove the stamp objects from the stamper instance and frees memory.

You should call this method after you'd called [SetaPDF_Stamper::stampit\(\)](#) and you want to execute additional code to free memory.

Version

Available since version 1.6.2.

SetaPDF_Stamper::setUseUpdate()

Description

```
SetaPDF_Stamper extends SetaPDF {  
    void setUseUpdate ( [boolean $useUpdate=true] )  
}
```

Defines if the resulting document should only be updated (very fast) or if the resulting document should be rebuilt from scratch.

Parameters

\$useUpdate

True or false

SetaPDF_Stamper::addFile()

Description

```
SetaPDF_Stamper extends SetaPDF {  
    mixed addFile ( array $file )  
}
```

With this method you can add a single pdf file to the stamping process.

Parameters

\$file

An array with the following keys:

- ▶ "in": The path to the original PDF document
- ▶ "out": The path oder the name of the resulting PDF document
- ▶ "compression": Turn compression of new content streams on or off

The key "compression" is optional. The default value is *false*

Return value

True - if everything works as expected - an SetaPDF_Error object if an error occurs.

SetaPDF_Stamper::addFiles()

Description

```
SetaPDF_Stamper extends SetaPDF {  
    mixed addFiles ( array $files )  
}
```

With this method you can add multiple PDF documents to the stamping process.

Parameters

\$files

An associative array of arrays, described in [SetaPDF_Stamper::addFile\(\)](#).

Return value

True - if everything works as expected - an SetaPDF_Error object if an error occurs.

SetaPDF_Stamper::getPageCount()

Description

```
SetaPDF_Stamper extends SetaPDF {  
    mixed getPageCount ( string $fileName )  
}
```

Returns the page count of the given PDF document.

The parser that is used to get the page count is cached in the stamper object instance and will be used later if it is needed by the stampit-method. If you don't need the parser, just release it from the stamper instance by calling the [cleanUp\(\)](#)-method.

Parameters

\$fileName

The path to the PDF document.

Return value

The page count or a SetaPDF_Error object if an error occurs.

Version

as of version 1.7

SetaPDF_Stamper::getPageBoxes()

Description

```
SetaPDF_Stamper extends SetaPDF {  
    mixed getPageBoxes ( integer $pageNo, string $fileName )  
}
```

Gets the page boxes of a PDF document.

Parameters

\$pageNo

The page number

\$fileName

The path of the PDF file.

Return value

An array of boxes, while the keys are the box-names (f.g. /MediaBox, /TrimBox,...).

Each box has following entries:

- ◆ x: the abscissa
- ◆ y: ordinate
- ◆ w: the width
- ◆ h: the height
- ◆ llx: lower left abscissa
- ◆ lly: lower left ordinate
- ◆ urx: upper right abscissa
- ◆ ury: upper right ordinate

If an error occurs this method will return a [SetaPDF_Error-Object](#).

Version

as of version 1.7

SetaPDF_Stamper::getOriginBox()

Description

```
SetaPDF_Stamper extends SetaPDF {  
    mixed getOriginBox ( integer $pageNo, mixed $fileName )  
}
```

Gets the origin page box. This will be the /CropBox or the /MediaBox. The method will check the existence of the desired box for you.

Width and height will be calculated automatically in relation to the rotation of the page.

Parameters

\$pageNo

The page number

\$fileName

The path of the PDF file.

Return Values

An array with the following entries:

- ◆ x: the abscissa
- ◆ y: ordinate
- ◆ w: the width
- ◆ h: the height
- ◆ llx: lower left abscissa
- ◆ lly: lower left ordinate
- ◆ urx: upper right abscissa
- ◆ ury: upper right ordinate

If an error occurs this method will return a [SetaPDF_Error-Object](#).

Version

as of version 1.7

SetaPDF_Stamper::getPageRotation()

Description

```
SetaPDF_Stamper extends SetaPDF {  
    mixed getPageRotation ( integer $pageNo, string $fileName )  
}
```

Gets the number of degrees by which the page is rotated.

Parameters

\$pageNo

The page number

\$fileName

The path of the PDF file.

Return Values

An integer value of a factor 90 or false if no value is given.

If an error occurs this method will return a [SetaPDF_Error-Object](#).

Version

as of version 1.7

SetaPDF_Stamper::setStamp()

Description

```
SetaPDF_Stamper extends SetaPDF {  
    mixed setStamp (    &$stamp[, string $position='LT'[, mixed  
        $showOnPage='all'[, float $translate_x=0[, float $translate_y=0[, float  
        $rotation=0[, boolean $underlay=false[, mixed $callback=null]]]]]] )  
}
```

With this method you can add multiple stamp objects to the stamping process.

You can add one stamp object more than one time on different positions, pages and/or rotations.

Parameters

&\$stamp

A reference to an instance of any stamp object.

\$position

The position of a Stamp is defined by special tokens like LT, LM, and so on. The letters are defined as follows:

- ▶ L: Left
- ▶ C: Center
- ▶ R: Right
- ▶ T: Top
- ▶ M: Middle
- ▶ B: Bottom

These letters can be combined to tokens like LT = LeftTop, RB = RightBottom,...

\$showOnPage

This parameter defines on which pages the stamp should appear.

Possible values are:

- ▶ "all": on all pages
- ▶ "odd": only on odd pages
- ▶ "even": only on even pages
- ▶ "first": only on the first page
- ▶ "last": only on the last page
- ▶ A one- or two-dimensional array which values represent the page numbers on which the stamp

should appear.

Also it's possible to create your own functions/methods which will check if the current page should be stamped. The possible values are:

- ▶ A callback functions name as a string
- ▶ An array of 2 values: key 0: an object instance, key 1: the method to call
- ▶ An array of 2 values: key 0: a class name (string), key 1: the static method to call (string)

The SetaPDF-Stamper API will call your given function/method with 2 parameter: "current page", "total number of pages in the document". The function/method should return *true* if the page should be stamped or *false* if it shouldn't.

So your function/method should look like this:

```
function shouldWeStamp($pageNo, $totalPageCount) {  
    return true||false;  
}
```

\$translate_x

The *\$translate_x*-parameter can be used to translate the position of the stamp on the x-axis. The input is done in points.

\$translate_y

The *\$translate_y*-parameter can be used to translate the position of the stamp on the y-axis. The input is done in points.

\$rotation

The rotation in degrees.

Be informed that the API will do an automatic correction of the position if an area of a stamp will be translated outside the viewing area.

\$underlay

This parameter let you define if the stamp should be layed above the existing content or under.

The default behaviour is to place them on top: *false*

\$callback

A callback function or method which should be called before the stamp is written on a specific page.

The possible values are:

- ▶ A callback functions name as a string
- ▶ An array of 2 values: key 0: an object instance, key 1: the method to call
- ▶ An array of 2 values: key 0: a class name (string), key 1: the static method to call (string)

The SetaPDF-Stamper API will call your given function/method with 3 parameter: "stamp properties" (a reference!), "current page", "total number of pages in the document". The function/method should return *true* if the stamp was changed.

So your function/method should look like this:

```
function adjustAlpha(&$stampProperties, $currentPageNo,  
                    $pageCount) {  
    // get the stamp object assigned in the first parameter of setStamp()  
    $stamp =& $stampProperties['stamp'];  
    // set the alpha/transparency in relation to the page count  
    $stamp->alpha += 1/$pageCount;  
    return true;  
}
```

More examples will follow in the coming days.

Return value

True - if everything works as expected - an SetaPDF_Error object if an error occurs.

SetaPDF_Stamper::updateCache()

As of version 1.6 a new global caching function exists and the API own functionality will be removed in coming version.

Description

```
SetaPDF_Stamper extends SetaPDF {  
    mixed updateCache ( void )  
}
```

This method creates a cache file for the PDF file defined in the [addFile](#) or [addFiles](#) method. It can be used to create the cache files beforehand or to update the cached versions manually.

The call of this method presupposes that [setUseCache\(true\)](#) is activated prior to this method call.

Return Values

True - if everything works as expected - an SetaPDF_Error object if an error occurs.

SetaPDF_Stamper::stampit()

Description

```
SetaPDF_Stamper extends SetaPDF {  
    mixed stampit ( void )  
}
```

This method starts the main stamp process.

Return Values

True - if everything works as expected - an SetaPDF_Error object if an error occurs.

SetaPDF_StamperPRO - Stamps encrypted PDF documents

This class offers the facility to stamp already encrypted PDF documents, if the owner password is known.

Because of this the only public method, which is overwritten is the [addFile](#)-method.

Class Overview



Methods

◆ [SetaPDF_StamperPRO::addFile\(\)](#)

Inherited Methods

Class: [SetaPDF_Stamper](#)

- ◆ [SetaPDF_Stamper::factory\(\)](#)
- ◆ [SetaPDF_Stamper::cleanUp\(\)](#)
- ◆ [SetaPDF_Stamper::setUseUpdate\(\)](#)
- ◆ [SetaPDF_Stamper::addFiles\(\)](#)
- ◆ [SetaPDF_Stamper::getPageCount\(\)](#)
- ◆ [SetaPDF_Stamper::getPageBoxes\(\)](#)
- ◆ [SetaPDF_Stamper::getOriginBox\(\)](#)
- ◆ [SetaPDF_Stamper::getPageRotation\(\)](#)
- ◆ [SetaPDF_Stamper::setStamp\(\)](#)
- ◆ [SetaPDF_Stamper::stampit\(\)](#)

Class: [SetaPDF](#)

- ◆ [SetaPDF::isError\(\)](#)

SetaPDF_StamperPRO::addFile()

Description

```
SetaPDF_StamperPRO extends SetaPDF_Stamper {  
    mixed addFile ( array $file )  
}
```

With this method you can add a single pdf file to the stamping process.

Parameters

\$file

An array with the following keys:

- ▶ "in": The path to the original PDF document
- ▶ "out": The path oder the name of the resulting PDF document
- ▶ "compression": Turn compression of new content streams on or off
- ▶ "ownerpw": The owner password if the document is encrypted.

The key "compression" is optional. The default value is *false*

Return value

True - if everything works as expected - an SetaPDF_Error object if an error occurs.