



## SetaPDF-FormFiller API

Manual and Reference

Version 1.5.4, 2010-03-25 16:55:46

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

<http://www.setasign.de>  
[support@setasign.de](mailto:support@setasign.de)

## Table of contents

- Introduction .....4
- System Requirements .....5
- Installation.....6
  - Ioncube .....7
  - Zend.....8
- Constants / Configuration .....9
- Caching.....13
- SetaPDF .....16
  - SetaPDF::isError() .....17
- SetaPDF\_Error .....18
- SetaPDF\_Parser.....19
  - SetaPDF\_Parser::cacheDir() .....20
  - SetaPDF\_Parser::cacheFlags() .....21
  - SetaPDF\_Parser::cacheMkdirMode() .....22
  - SetaPDF\_Parser::cacheNoOfObjectsPerInstance() .....23
  - SetaPDF\_Parser::cacheHashFunction() .....24
- SetaPDF\_FormFiller .....25
  - SetaPDF\_FormFiller::factory() .....26
  - SetaPDF\_FormFiller::destruct() .....28
  - SetaPDF\_FormFiller::\_\_destruct() .....29
  - SetaPDF\_FormFiller::setCompatMode() .....30
  - SetaPDF\_FormFiller::setUseUpdate() .....31
  - SetaPDF\_FormFiller::getFields() .....32
  - SetaPDF\_FormFiller::getField() .....33
  - SetaPDF\_FormFiller::getRelatedFields() .....34
  - SetaPDF\_FormFiller::fillForms() .....35
- SetaPDF\_FormField .....36
  - SetaPDF\_FormField::setValue() .....37
  - SetaPDF\_FormField::getValue() .....38
  - SetaPDF\_FormField::getName() .....39
  - SetaPDF\_FormField::getMD5Name() .....40
  - SetaPDF\_FormField::setReadOnly() .....41
- SetaPDF\_TextField .....42
  - SetaPDF\_TextField::setValue() .....43
  - SetaPDF\_TextField::getValue() .....44
  - SetaPDF\_TextField::setTranslateLeading() .....45
  - SetaPDF\_TextField::setLink() .....46

SetaPDF\_ChoiceField .....47

    SetaPDF\_ChoiceField::getOptions() .....48

    SetaPDF\_ChoiceField::setValue() .....49

    SetaPDF\_ChoiceField::getValue() .....50

SetaPDF\_ButtonField .....51

    SetaPDF\_ButtonField::push() .....52

    SetaPDF\_ButtonField::pull() .....53

    SetaPDF\_ButtonField::setValue() .....54

    SetaPDF\_ButtonField::getValue() .....55

    SetaPDF\_ButtonField::isPushed() .....56

SetaPDF\_ButtonField\_Group .....57

    SetaPDF\_ButtonField\_Group::getButtons() .....58

    SetaPDF\_ButtonField\_Group::getName() .....59

    SetaPDF\_ButtonField\_Group::getMD5Name() .....60

    SetaPDF\_ButtonField\_Group::setReadOnly() .....61

    SetaPDF\_ButtonField\_Group::setValue() .....62

## SetaPDF-FormFiller - Introduction

This API allows PHP developers to query and edit values of PDF form fields (also known as AcroForms).

All field types are accessible through an object orientated way: a form field is a simple PHP object.

To get access to the form fields you have to create an instance of the [SetaPDF\\_FormFiller](#) object by the [factory\(\)](#)-method.

The [SetaPDF\\_FormFiller](#)-object offers you the method [SetaPDF\\_FormFiller::getFields\(\)](#), which will return all found form fields as an array of object references.

You can edit and/or query the values of the form fields by their specific method, which were described in this manual.

After you edited the values you can simply output or save the updated PDF form with the [SetaPDF\\_FormFiller::fillForms\(\)](#)-method.

## SetaPDF-FormFiller - 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.

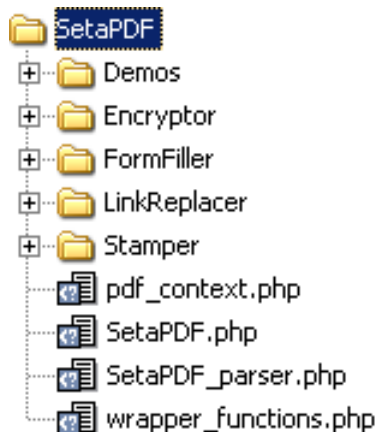
The SetaPDF-FormFiller API did not offers such a caching system.

## SetaPDF-FormFiller - Installation

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

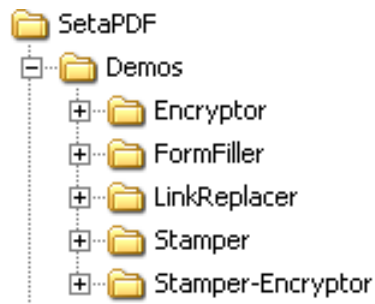
### *Files and directories*

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/');
```

## SetaPDF-FormFiller 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-FormFiller API - Zend encoded package

If you own a package, which is encoded with the Zend Safeguard Suite you have to install the Zend Optimizer. For more information please go to [http://www.zend.com/products/zend\\_optimizer](http://www.zend.com/products/zend_optimizer).

### *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 on runtime in your php script ([zend\\_loader\\_install\\_license\(\)](#)) or by default in your [php.ini](#).

Zended packages are only available for development- and serverlicenses.

## SetaPDF-FormFiller API - Constants / Configuration

The API needs some constants which are hard coded into the API or have to be defined by you.

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

### *Defined Constants*

```
AREADER_5 (integer)0
```

Represents the compatibility to Acrobat (Reader) version 5.

```
AREADER_6 (integer)1
```

Represents the compatibility to Acrobat (Reader) version 6.

**AREADER\_7** (integer)2

Represents the compatibility to Acrobat (Reader) version 7.

### *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-FormFiller API starts at 100 and ends at 199.

**E\_SETAPDF\_FF\_NO\_FORM\_FIELDS\_FOUND** (integer)100

No Formfields found!

**E\_SETAPDF\_FF\_FONT\_NOT\_SUPPORTED** (integer)101

Font 'XXXXXXX' is currently not supported.

**E\_SETAPDF\_FF\_NOTHING\_TODO** (integer)102

Nothing to do...? Please define some fields

**E\_SETAPDF\_FF\_LISTBOX\_ACCEPTS\_ONLY\_ONE\_VALUE** (integer)103

This listbox only accept one value!

**E\_SETAPDF\_FF\_LISTBOX\_NO\_ELEMENT\_ON\_INDEX** (integer)104

No Element found on Index 'X'!

**E\_SETAPDF\_FF\_NO\_RELATED\_FIELDS\_FOUND** (integer)105

No related fields found for field XXX.

**E\_SETAPDF\_FF\_NO\_FONT\_FOUND** (integer)106

No font found for textfield.

### *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-FormFiller - 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\\_FormFiller](#)

### *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\_FormFiller - Main class

This is the main class of the SetaPDF-FormFiller API.

### *Class Overview*



### *Methods*

- ◆ [SetaPDF\\_FormFiller::factory\(\)](#)
- ◆ [SetaPDF\\_FormFiller::destruct\(\)](#)
- ◆ [SetaPDF\\_FormFiller:: \\_destruct\(\)](#)
- ◆ [SetaPDF\\_FormFiller::setCompatMode\(\)](#)
- ◆ [SetaPDF\\_FormFiller::setUseUpdate\(\)](#)
- ◆ [SetaPDF\\_FormFiller::getFields\(\)](#)
- ◆ [SetaPDF\\_FormFiller::getField\(\)](#)
- ◆ [SetaPDF\\_FormFiller::getRelatedFields\(\)](#)
- ◆ [SetaPDF\\_FormFiller::fillForms\(\)](#)

### *Inherited Methods*

**Class:** [SetaPDF](#)

- ◆ [SetaPDF::isError\(\)](#)

### SetaPDF\_FormFiller::factory()

#### Description

```
SetaPDF_FormFiller extends SetaPDF {  
    mixed factory ( string $sourcefile[, string $password="", string  
        $dest="F"[, boolean $stream=false[, boolean  
        $renderAppearancesByViewer=false[, string  
        $encoding="windows-1252//IGNORE" ]]]] )  
}
```

This method has to be called static and will return an instance of the SetaPDF\_FormFiller class or an SetaPDF\_Error object.

#### Parameters

##### *\$sourcefile*

A string that defines the path (relative or absolute) to the original document. Only local paths are allowed.

##### *\$password*

Expects the owner password of the original PDF document if it is encrypted and no rights are granted for editing form fields.

##### *\$dest*

Defines how the resulting document is handled:

- ▶ "F" saves the file to the file system
- ▶ "D" the file will be send to the client with a download dialogue
- ▶ "I" the file will be displayed in the client's browser window.

##### *\$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.

##### *\$renderAppearancesByViewer*

Defines that the rendering of form fields will be done by the viewer instead of the API. With this option you also can use UTF-16 strings in form field values and fill in rich text fields (see [SetaPDF\\_TextField::setValue\(\)](#))

### *\$encoding*

The encoding used internally for field names. Also it is used as the default encoding for field values returned by any `getValue()/getOptions()` method.

Internally the encoding is passed to a wrapper class (`SetaPDF/Tools/Encoding.php`) which utilize `iconv` for the conversion.

### *Return Values*

In case of success you get a new instance of the [SetaPDF\\_FormFiller](#) class.

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

### *Version*

The `$encoding` parameter is available since version 1.5.4

## SetaPDF\_FormFiller::destruct()

### *Description*

```
SetaPDF FormFiller extends SetaPDF {  
    void destruct ( void )  
}
```

This method will releases memory. You have to call it before an unset()-call on the instance variable to release memory.

### *Version*

*Avilable since 1.5.1*

## SetaPDF\_FormFiller::\_\_destruct()

### *Description*

```
SetaPDF_FormFiller extends SetaPDF {  
    void __destruct ( void )  
}
```

**(DEPRECATED)** Alias for [SetaPDF\\_FormFiller::destroy\(\)](#).

### *Version*

Deprecated since version 1.5.1

## SetaPDF\_FormFiller::setCompatMode()

### *Description*

```
SetaPDF FormFiller extends SetaPDF {  
    setCompatMode ( integer $compatMode )  
}
```

Sets the compatibility mode for different Acrobat versions.

### *Parameters*

#### ***\$compatMode***

Potential values include:

- ▶ AREADER\_5
- ▶ AREADER\_6
- ▶ AREADER\_7

## SetaPDF\_FormFiller::setUseUpdate()

### *Description*

```
SetaPDF FormFiller 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\_FormFiller::getFields()

### *Description*

```
SetaPDF FormFiller extends SetaPDF {  
    array getFields ( void )  
}
```

Returns an array of field objects. All entries should be handled as references.

### *Return value*

An array of field objects.

The keys are the form field names (see [SetaPDF\\_FormField::getName\(\)](#)).

## SetaPDF\_FormFiller::getField()

### *Description*

```
SetaPDF FormFiller extends SetaPDF {  
    mixed getField ( string $name )  
}
```

Returns a reference to the specific form field.

### *Parameters*

#### ***\$name***

The name of the form field

### *Return value*

A reference to the specific form field. If no form field with the give name is found the method will return *false*

.

## SetaPDF\_FormFiller::getRelatedFields()

### *Description*

```
SetaPDF FormFiller extends SetaPDF {  
    mixed getRelatedFields ( [string $byFieldName=null] )  
}
```

This method can be used to query related field names (same named fields).

### *Parameters*

#### *\$byFieldName*

The name of a form field.

### *Return value*

An array of field names if the \$byFieldName isn't *NULL* and a form field with the given name exists.

If \$byFieldName is *NULL* an array with all related field names will be returned. For each field with related fields an array entry will be created where by the original field names will be the key in the main array.

## SetaPDF\_FormFiller::fillForms()

### *Description*

```
SetaPDF FormFiller extends SetaPDF {  
    mixed fillForms ( string $target )  
}
```

This method fills in all changed form fields and send the resulting PDF file to the client or saves it to the local filesystem (as defined in the \$dest-parameter of the [factory-method](#)).

### *Parameters*

#### **\$target**

The resulting filename or path to the local filesystem where the resulting document will be saved.

### *Return value*

*True* - if everything works as expected - an SetaPDF\_Error object if an error occurs.

## SetaPDF\_FormField - Abstract Form Field Class

This class is the abstract class for all available form field types.

### *Class Overview*

SetaPDF\_FormField

### *Child Classes*

- ▶ [SetaPDF\\_TextField](#)
- ▶ [SetaPDF\\_ChoiceField](#)
- ▶ [SetaPDF\\_ButtonField](#)

### *Methods*

- ▶ [SetaPDF\\_FormField::setValue\(\)](#)
- ▶ [SetaPDF\\_FormField::getValue\(\)](#)
- ▶ [SetaPDF\\_FormField::getName\(\)](#)
- ▶ [SetaPDF\\_FormField::getMD5Name\(\)](#)
- ▶ [SetaPDF\\_FormField::setReadOnly\(\)](#)

## SetaPDF\_FormField::setValue()

### *Description*

```
SetaPDF FormField {  
    void setValue ( void )  
}
```

Sets a value of a specific form field. Details for each form field type are described separately.

## SetaPDF\_FormField::getValue()

### *Description*

```
SetaPDF FormField {  
    void getValue ( void )  
}
```

Gets the current value of a form field. Details for each form field type are described separately.

## SetaPDF\_FormField::getName()

### Description

```
SetaPDF FormField {  
    string getName ( [mixed $oname=false] )  
}
```

This method returns the name of a form field. This is also the index, which is used in the fields-array returned by [SetaPDF\\_FormFiller::getFields\(\)](#). Furthermore this name can be used to query a form field by [SetaPDF\\_FormFiller::getField\(\)](#).

### Parameters

#### *\$oname*

This parameter is only available as of version 1.2.3.

It has only a meaning if you use form field references in your PDF document. That means, you use a single name for different form fields. If you change one of this fields the value will be updated in each related field.

The API suffixes such form field names with '#N', where N is an incremental number. To get the real name (without '#N') you simply have to pass *true* to this method.

To get all related fields you can use the 'relatedFields'-property of the SetaPDF\_FormFiller instance.

Before version 1.2.3 you can simply check the 'oname'-property.

### Return Value

Returns the specific name of a form field. This is the value which was defined in f.g. Acrobat for a form field.

## SetaPDF\_FormField::getMD5Name()

### *Description*

```
SetaPDF FormField {  
    string getMD5Name ( void )  
}
```

Acrobat accepts any name for form fields - also special chars, which cannot be used within normal html forms. This method returns the md5-hash of the form fields name to give you the possibility to match the md5-string to the original form field.

### *Return Value*

Returns the names md5-hash of a specific form field.

## SetaPDF\_FormField::setReadOnly()

### *Description*

```
SetaPDF FormField {  
    void setReadOnly ( [boolean $value=true] )  
}
```

Sets or removes the read-only flag from the desired form field.

### *Parameters*

#### *\$value*

Set (*true*) or remove (*false*) the read-only flag.

### *Version*

available since version 1.5

## SetaPDF\_TextField

This class modifies text fields with one or more lines.

### *Class Overview*



### *Methods*

- ▶ [SetaPDF\\_TextField::setValue\(\)](#)
- ▶ [SetaPDF\\_TextField::getValue\(\)](#)
- ▶ [SetaPDF\\_TextField::setTranslateLeading\(\)](#)
- ▶ [SetaPDF\\_TextField::setLink\(\)](#)

### *Inherited Methods*

**Class:** [SetaPDF\\_FormField](#)

- ▶ [SetaPDF\\_FormField::getName\(\)](#)
- ▶ [SetaPDF\\_FormField::getMD5Name\(\)](#)
- ▶ [SetaPDF\\_FormField::setReadOnly\(\)](#)

## SetaPDF\_TextField::setValue()

### Description

```
SetaPDF_TextField extends SetaPDF_FormField {  
    mixed setValue ( mixed $value[, boolean $dontReCreateAppearance=false] )  
}
```

Sets the new value for a form field.

### Parameters

#### *\$value*

The new value of the text field.

Experimental: If the text field is defined as a rich text field and the `$renderAppearancesByViewer`-parameter of the `factory`-method is set to `true`, the method accepts an array with 2 keys:

#### *Key 0:*

The value for the style-attribute of the body-tag.

#### *Key 1:*

The new content of the body tag.

#### *\$dontReCreateAppearance*

This flag is experimental and is recognized only if the `$renderAppearancesByViewer`-parameter of the `factory`-method is set to `true`. If this parameter is set to `true` the API will not create the appearance for the text field (this will be done by the viewer application), which will lead into a faster form filling process.

### Return Value

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

## SetaPDF\_TextField::getValue()

### Description

```
SetaPDF_TextField extends SetaPDF_FormField {  
    mixed getValue ( [boolean $richText=false[, string $encoding=null]] )  
}
```

Gets the current value of the text field.

### Parameters

#### *\$richText*

If this value is set to true and the form field is defined as a richtext field, the API returns an array with the following keys:

#### *body:*

The current content of the body-tag.

#### *bodystyle:*

The value of the style-attribute of the body-tag.

#### *value:*

The plaintext value.

#### *\_xml:*

The complete xml data.

#### *\$encoding*

Defines the encoding of the returned string. If left to "null" the encoding provided in the [factory](#)-method is used (default: windows-1252//IGNORE). The conversion is handle through a wrapper class around [iconv](#) (SetaPDF/Tools/Encoding.php).

Pass false to get the untouched value in PdfDocEncoding or UTF-16BE.

### Return Value

A normal text field will return the current value.

A rich text field will return an array, as described above.

## SetaPDF\_TextField::setTranslateLeading()

### *Description*

```
SetaPDF_TextField extends SetaPDF_FormField {  
    void setTranslateLeading ( integer $translateLeading )  
}
```

In some cases the API cannot recalculate the desired leading of a font in a specific formfield.

This method can be used to adjust this kind of leading.

Be aware that nearly every Acrobat Reader version handles leading in multiline textfields in a different way.

Also see [SetaPDF\\_FormFiller::setCompatMode\(\)](#)

### *Parameters*

#### *\$translateLeading*

The translation of the leading in pt.

## SetaPDF\_TextField::setLink()

*Description (Experimental!)*

```
SetaPDF_TextField extends SetaPDF_FormField {  
    boolean setLink ( string $link )  
}
```

Adds a link annotation above the formfield.

### *Parameters*

#### *\$link*

The link value. For example: <http://www.setasign.de>

### *Return Value*

*True*, if everything works as expected. *False* if the API was not able to add a link annotation above the formfield.

## SetaPDF\_ChoiceField

This class modifies choice fields.

### *Class Overview*



### *Methods*

- ▶ [SetaPDF\\_ChoiceField::getOptions\(\)](#)
- ▶ [SetaPDF\\_ChoiceField::setValue\(\)](#)
- ▶ [SetaPDF\\_ChoiceField::getValue\(\)](#)

### *Inherited Methods*

**Class:** [SetaPDF\\_FormField](#)

- ▶ [SetaPDF\\_FormField::getName\(\)](#)
- ▶ [SetaPDF\\_FormField::getMD5Name\(\)](#)
- ▶ [SetaPDF\\_FormField::setReadOnly\(\)](#)

## SetaPDF\_ChoiceField::getOptions()

### *Description*

```
SetaPDF_ChoiceField extends SetaPDF_FormField {  
    array getOptions ( [string $encoding='windows-1252'] )  
}
```

This method returns the available options of the desired choice field.

### *Parameters*

#### *\$encoding*

Defines the encoding of the returned strings.

By default the API use [mb\\_convert\\_encoding\(\)](#) to convert the encoding. If the mb\_\* functions are not available [iconv\(\)](#) is used.

### *Return Value*

An array of name and value pairs.

## SetaPDF\_ChoiceField::setValue()

### Description

```
SetaPDF_ChoiceField extends SetaPDF_FormField {  
    mixed setValue ( mixed $value )  
}
```

Sets the new selected option or options of the choice field.

### Parameters

#### *\$value*

To select a single option just pass the zerobased index returned from [SetaPDF\\_ChoiceField::getOptions\(\)](#) to this method. In case to select multiple options (if allowed) pass an array with the zerobased indexes as values to this method.

A choice field allows different values depending on it's own definition:

If the choice field is defined as a combo box, it just can hold one value.

If it is defined as a list box and multiselect is set it can hold multiple values.

A special case is a choice field which offers an editable text box. In this case the value haven't to be an index of the option list but can be any text like a [text field](#).

### Return Value

*True*, if everythings works as expected. A *SetaPDF\_Error* object if an error occurs.

## SetaPDF\_ChoiceField::getValue()

### *Description*

```
SetaPDF_ChoiceField extends SetaPDF_FormField {  
    mixed getValue ( [string $encoding=null] )  
}
```

Returns the current selected option(s).

### *Parameters*

#### *\$encoding*

Defines the encoding of the returned string(s). If left to "null" the encoding provided in the [factory](#)-method is used (default: windows-1252//IGNORE). The conversion is handle through a wrapper class around [iconv](#) (SetaPDF/Tools/Encoding.php).

Pass false to get the untouched value in PdfDocEncoding or UTF-16BE.

### *Return Value*

A string or an array of strings representing the currently selected values.

## SetaPDF\_ButtonField

This class modifies button fields like checkboxes and radiobuttons. A radio button group is managed through the [SetaPDF\\_ButtonField\\_Group](#) class.

### *Class Overview*



### *Methods*

- ◆ [SetaPDF\\_ButtonField::push\(\)](#)
- ◆ [SetaPDF\\_ButtonField::pull\(\)](#)
- ◆ [SetaPDF\\_ButtonField::setValue\(\)](#)
- ◆ [SetaPDF\\_ButtonField::getValue\(\)](#)
- ◆ [SetaPDF\\_ButtonField::isPushed\(\)](#)

### *Inherited Methods*

**Class:** [SetaPDF\\_FormField](#)

- ◆ [SetaPDF\\_FormField::getName\(\)](#)
- ◆ [SetaPDF\\_FormField::getMD5Name\(\)](#)
- ◆ [SetaPDF\\_FormField::setReadOnly\(\)](#)

## SetaPDF\_ButtonField::push()

### *Description*

```
SetaPDF_ButtonField extends SetaPDF_FormField {  
    mixed push ( void )  
}
```

Pushes/activates a checkbox or radio button.

### *Return Value*

*True*, if everything works as expected. A SetaPDF\_Error object if an error occurs.

## SetaPDF\_ButtonField::pull()

### *Description*

```
SetaPDF_ButtonField extends SetaPDF_FormField {  
    mixed pull ( void )  
}
```

Deactivates a checkbox or radio button.

### *Return Value*

*True*, if everything works as expected. A *SetaPDF\_Error* object if an error occurs.

## SetaPDF\_ButtonField::setValue()

### Description

```
SetaPDF_ButtonField extends SetaPDF_FormField {  
    mixed setValue ( [boolean $active=true] )  
}
```

This method is a wrapper method for [pull\(\)](#) or [push\(\)](#) calls.

### Parameter

#### *\$active*

Use *true* for activate/check the checkbox or radiobutton. Use *false* to deactivate it.

### Return Value

*True*, if everything works as expected. A *SetaPDF\_Error* object if an error occurs.

## SetaPDF\_ButtonField::getValue()

### *Description*

```
SetaPDF_ButtonField extends SetaPDF_FormField {  
    string getValue ( void )  
}
```

Get the current exportvalue of the form field.

### *Return Value*

The exportvalue of the form field. If field is not active this method returns /Off

## SetaPDF\_ButtonField::isPushed()

### *Description*

```
SetaPDF_ButtonField extends SetaPDF_FormField {  
    boolean isPushed ( void )  
}
```

Checks if a checkbox or radiobutton is active.

### *Return Value*

*True* or *false* indication if the form field is active or deactive.

## SetaPDF\_ButtonField\_Group

This class represents a group of buttons as radiobutton groups known from html.

This class is not inherited from [SetaPDF\\_FormField](#), because it just acts as a collection and maps actions through the different form fields.

### *Class Overview*

SetaPDF\_ButtonField\_Group

### *Methods*

- ◆ [SetaPDF\\_ButtonField\\_Group::getButtons\(\)](#)
- ◆ [SetaPDF\\_ButtonField\\_Group::getName\(\)](#)
- ◆ [SetaPDF\\_ButtonField\\_Group::getMD5Name\(\)](#)
- ◆ [SetaPDF\\_ButtonField\\_Group::setReadOnly\(\)](#)
- ◆ [SetaPDF\\_ButtonField\\_Group::setValue\(\)](#)

## SetaPDF\_ButtonField\_Group::getButtons()

### *Description*

```
SetaPDF_ButtonField_Group {  
    array getButtons ( void )  
}
```

This method gives access to all [SetaPDF\\_ButtonField](#) objects that are part of the group.

### *Return Value*

An array of [SetaPDF\\_ButtonField](#) objects.

## SetaPDF\_ButtonField\_Group::getName()

### *Description*

```
SetaPDF_ButtonField_Group {  
    string getName ( void )  
}
```

This method returns the name of a form field. This is also the index, which is used in the fields-array returned by [SetaPDF\\_FormFiller::getFields\(\)](#). Furthermore this name can be used to query a form field by [SetaPDF\\_FormFiller::getField\(\)](#).

### *Return Value*

Returns the specific name of a form field. This is the value which was defined in f.g. Acrobat for a form field.

## SetaPDF\_ButtonField\_Group::getMD5Name()

### *Description*

```
SetaPDF_ButtonField_Group {  
    string getMD5Name ( void )  
}
```

Acrobat accepts any name for form fields - also special chars, which cannot be used within normal html forms. This method returns the md5-hash of the form fields name to give you the possibility to match the md5-string to the original form field.

### *Return Value*

Returns the names md5-hash of a specific form field.

## SetaPDF\_ButtonField\_Group::setReadOnly()

### *Description*

```
SetaPDF_ButtonField_Group {  
    void setReadOnly ( [boolean $value=true] )  
}
```

Sets or removes the read-only flag from the field group.

### *Parameters*

#### *\$value*

Set (*true*) or remove (*false*) the read-only flag.

### *Version*

available since version 1.5

## SetaPDF\_ButtonField\_Group::setValue()

### *Description*

```
SetaPDF_ButtonField_Group {  
    mixed setValue ( string $buttonName )  
}
```

Pushes the desired button in the button group.

### *Parameters*

#### ***\$buttonName***

The name of the button to push.

### *Return Value*

*True*, if everything works as expected. A *SetaPDF\_Error* object if an error occurs.