



SetaPDF-Merger API

Manual and Reference

Version 1.5.5, 2010-08-24 12:34:09

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

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

Table of contents

Introduction	3
System Requirements	4
Installation.....	5
Ioncube	6
Zend.....	7
Constants / Configuration	8
Caching.....	12
SetaPDF	15
SetaPDF::isError()	16
SetaPDF_Error	17
SetaPDF_Parser.....	18
SetaPDF_Parser::cacheDir()	19
SetaPDF_Parser::cacheFlags().....	20
SetaPDF_Parser::cacheMkdirMode().....	21
SetaPDF_Parser::cacheNoOfObjectsPerInstance().....	22
SetaPDF_Parser::cacheHashFunction()	23
SetaPDF_Merger.....	24
SetaPDF_Merger::factory().....	25
SetaPDF_Merger::cleanUp()	26
SetaPDF_Merger::addFile().....	27
SetaPDF_Merger::getPageCount()	29
SetaPDF_Merger::setAuthor()	30
SetaPDF_Merger::setCreator().....	31
SetaPDF_Merger::setDisplayMode()	32
SetaPDF_Merger::setKeywords().....	33
SetaPDF_Merger::setPageMode()	34
SetaPDF_Merger::setSubject().....	35
SetaPDF_Merger::setTitle().....	36
SetaPDF_Merger::setHandleNamedReferences()	37
SetaPDF_Merger::setAddBookmarks()	38
SetaPDF_Merger::setPdfVersionCallback()	39
SetaPDF_Merger::setFormFieldNamesCallback()	40
SetaPDF_Merger::merge()	42
SetaPDF_Merger::extract().....	43
Bookmarks.....	45
Dynamic content and PDF forms.....	51

SetaPDF-Merger API - Introduction

The SetaPDF-Merger API allows PHP developers to concatenate or split PDF documents in pure PHP.

It's simple interface allows merging documents with just a few lines of php code.

SetaPDF-Merger 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.

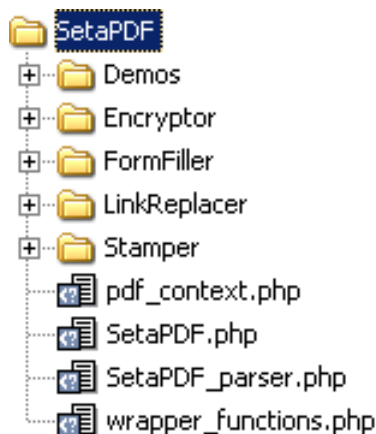
The SetaPDF-Merger API did not offers such a caching system.

SetaPDF-Merger API - Installation

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

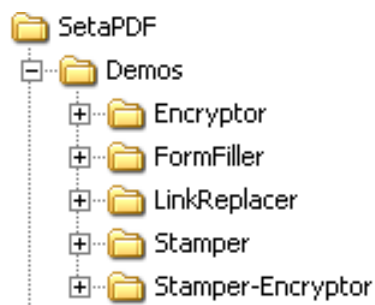
Files and directories

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-Merger API with the following line:

```
require_once("Merger/SetaPDF_Merger.php");
```

SetaPDF-Merger 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-Merger 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-Merger API - Constants

The API needs some constants which are hard coded into the API.

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

Predefined Constants

```
SETAPDF_M_BOOKMARK_FORMAT_ITALIC (integer)0x01
```

Display a bookmark in italic style.

```
SETAPDF_M_BOOKMARK_FORMAT_BOLD (integer)0x02
```

Display a bookmark in bold style.

```
SETAPDF_M_COPY_BOOKMARKS_AS_CHILD (integer)0x01
```

Defines that existing bookmarks should be copied as a tree under the entry for the desired page range/document.

```
SETAPDF_M_COPY_BOOKMARKS_TO_SAME_LEVEL (integer)0x02
```

Defines that the bookmarks will be imported without a parent entry node in the current level (defined by the parent-key in the given \$bookmark-array).

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-Merger API starts at 500 and ends at 599.

E_SETAPDF_M_PARSER_NO_KIDS (integer)500

Cannot find /Kids in current /Page-Dictionary

E_SETAPDF_M_WRONG_ZOOM_MODE (integer)501

Incorrect zoom display mode: XXXXX

E_SETAPDF_M_WRONG_DISPLAY_MODE (integer)502

Incorrect layout display mode: XXXXXX

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-Merger 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-Stamp 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_Merger](#)

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-Merger- Main class

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

Class Overview



Methods

- ◆ [SetaPDF_Merger::factory\(\)](#)
- ◆ [SetaPDF_Merger::cleanUp\(\)](#)
- ◆ [SetaPDF_Merger::addFile\(\)](#)
- ◆ [SetaPDF_Merger::getPageCount\(\)](#)
- ◆ [SetaPDF_Merger::setAuthor\(\)](#)
- ◆ [SetaPDF_Merger::setCreator\(\)](#)
- ◆ [SetaPDF_Merger::setDisplayMode\(\)](#)
- ◆ [SetaPDF_Merger::setKeywords\(\)](#)
- ◆ [SetaPDF_Merger::setPageMode\(\)](#)
- ◆ [SetaPDF_Merger::setSubject\(\)](#)
- ◆ [SetaPDF_Merger::setTitle\(\)](#)
- ◆ [SetaPDF_Merger::setHandleNamedReferences\(\)](#)
- ◆ [SetaPDF_Merger::setAddBookmarks\(\)](#)
- ◆ [SetaPDF_Merger::setPdfVersionCallback\(\)](#)
- ◆ [SetaPDF_Merger::setFormFieldNamesCallback\(\)](#)
- ◆ [SetaPDF_Merger::merge\(\)](#)
- ◆ [SetaPDF_Merger::extract\(\)](#)

Inherited Methods

Class: [SetaPDF](#)

- ◆ [SetaPDF::isError\(\)](#)

SetaPDF_Merger::factory()

Description

```
SetaPDF_Merger extends SetaPDF {  
    factory ( void )  
}
```

This method has to be called static and will return an instance of the SetaPDF_Merger class.

Return Values

A new instance of the SetaPDF_Merger class.

SetaPDF_Merger::cleanUp()

Description

```
SetaPDF_Merger extends SetaPDF {  
    boolean cleanUp ( void )  
}
```

This method will close all opened parser objects and will close any currently opened filehandle.

This method is called normally internal only. But it gets public as the [extract\(\)](#)-method was added which will not close parser objects automatically.

Return Values

Always *true*.

SetaPDF_Merger::addFile()

Description

```
SetaPDF_Merger extends SetaPDF {  
    void addFile ( string $fileName[, mixed $pages=false[, array  
        $bookmark=null]] )  
}
```

Add a given file and the desired pages to the resulting pdf document.

Parameters

\$fileName

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

\$pages

In this parameter you can define the page(s) which should be read from the given document.

A single page can be defined as an integer value. A range can be defined as an array with 2 values:

```
array($startPage, $endPage)
```

\$bookmark

This parameter defines the handling of a bookmark entry for this page range/document. There are a few possibilities for this parameter:

1. Pass *null* or omit the parameter will tell the API to resolve the title text for the bookmark entry from the meta data of the merged document.
2. Pass a simple string to this parameter, which will become the title text in the bookmark entry.
3. Pass *false* to prevent a bookmark creation for this page range/document.
4. Pass an array with the following structure:

```
array(  
    'title' => (string) "The bookmark title"  
    'color' => array(red, green, blue) // Text color  
    'format' => SETAPDF_M_BOOKMARK_FORMAT_ITALIC |  
    SETAPDF_M_BOOKMARK_FORMAT_BOLD,  
    'parent' => an Id of a parent node returned by addFile,  
    'copyBookmarks' => SETAPDF_M_COPY_BOOKMARKS_AS_CHILD ||
```

SETAPDF M COPY BOOKMARKS TO SAME LEVEL

)

All keys are optional and are described [here](#) with examples and detailed description.

Return Value

Will return an unique id, if the method call adds a new bookmark entry, which can be reused in the bookmark parameter to build a child/parent structure.

If the method call will not create a bookmark entry or will just import existing bookmark nodes into the same level, the method will return *null*.

SetaPDF_Merger::getPageCount()

Description

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

Returns the page count of a given PDF document.

Parameters

\$fileName

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

Return Value

The page count, if everything works as expected - an SetaPDF_Error object if an error occurs.

SetaPDF_Merger::setAuthor()

Description

```
SetaPDF_Merger extends SetaPDF {  
    void setAuthor ( string $author )  
}
```

Defines the author of the resulting document.

Parameters

\$author

The name of the author.

SetaPDF_Merger::setCreator()

Description

```
SetaPDF_Merger extends SetaPDF {  
    void setCreator ( string $creator )  
}
```

Defines the creator of the document. (Normally the name of your application)

Parameters

\$creator

The name of the creator

SetaPDF_Merger::setDisplayMode()

Description

```
SetaPDF_Merger extends SetaPDF {  
    mixed setDisplayMode ( mixed $zoom[, mixed $layout=false] )  
}
```

Defines the way the document is to be displayed by the viewer.

Parameters

\$zoom

The zoom to use. It can be one of the following string values:

- ▶ *fullpage*: displays the entire page on screen
- ▶ *fullwidth*: uses maximum width of window
- ▶ *real*: uses real size (equivalent to 100% zoom)
- ▶ *default*: uses viewer default mode

or a number indicating the zooming factor to use.

\$layout

The page layout. Possible values are:

- ▶ *single*: displays one page at once
- ▶ *continuous*: displays pages continuously
- ▶ *two*: displays two pages on two columns
- ▶ *default*: uses viewer default mode

Default value is *continuous*.

Return Value

Just returns an SetaPDF_Error object if an error occurs. In case of success the method returns nothing.

SetaPDF_Merger::setKeywords()

Description

```
SetaPDF_Merger extends SetaPDF {  
    void setKeywords ( string $keywords )  
}
```

Defines the associated keywords for the resulting document.

Parameters

\$keywords

The keywords

SetaPDF_Merger::setPageMode()

Description

```
SetaPDF_Merger extends SetaPDF {  
    mixed setDisplayMode ( string $pageMode )  
}
```

Defines the way the document is to be displayed when opened.

Parameters

\$pageMode

A name specifying how the document shall be displayed when opened:

- ▶ *UseNone*: Neither document outline nor thumbnail images are visible
- ▶ *UseOutlines*: Document outlines are visible
- ▶ *UseThumbs*: Thumbnail images are visible
- ▶ *FullScreen*: Full-screen mode
- ▶ *UseOC*: Optional content group panel is visible
- ▶ *UseAttachments*: The attachments panel is visible

The default value is *UseNone*.

Return Value

Just returns an *SetaPDF_Error* object if an error occurs. In case of success the method returns nothing.

Version

since version 1.6

SetaPDF_Merger::setSubject()

Description

```
SetaPDF_Merger extends SetaPDF {  
    void setSubject ( string $subject )  
}
```

Defines the subject of the resulting document.

Parameters

\$subject

The subject

SetaPDF_Merger::setTitle()

Description

```
SetaPDF_Merger extends SetaPDF {  
    void setTitle ( string $title )  
}
```

Defines the title of the resulting document.

Parameters

\$title

The title

SetaPDF_Merger::setHandleNamedReferences()

Description

```
SetaPDF_Merger extends SetaPDF {  
    void setHandleNamedReferences ( boolean $handleNamedReferences )  
}
```

A PDF document can include named references, which are defined outside the content part. They are for example used for linking inside a document. To avoid naming problems the API suffixes such names, which could increase the process time.

With this method you were able to turn this behaviour on or off.

Parameters

\$handleNamedReferences

A boolean value:

true: Import and suffix named relations (default)

false: Left named relations

SetaPDF_Merger::setAddBookmarks()

Description

```
SetaPDF_Merger extends SetaPDF {  
    void setAddBookmarks ( [boolean $addBookmarks=true[, boolean  
        $showOutline=true]] )  
}
```

With this method you can define that the API should add bookmarks to the resulting PDF document.

By default this behavior is set to *false*.

Parameters

\$addBookmarks

A boolean value:

true: add bookmarks

false: add no bookmarks (default)

\$showOutline

If this parameter is set to *true*, the bookmarks outline is visible when opening the resulting PDF document.

SetaPDF_Merger::setPdfVersionCallback()

Description

```
SetaPDF_Merger extends SetaPDF {  
    void setPdfVersionCallback ( callback $pdfVersionCallback )  
}
```

The API normally produces PDF documents with the highest version number of the read PDF documents.

In some special cases it is necessary to adjust the PDF version of the resulting document(s). You can use this method to define a callback function/method, which gives you the control over the PDF version.

You should only use this method, if you know what you're doing.

Parameters

\$pdfVersionCallback

A callback function or method which will be called to adjust the resulting PDF version.

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

The SetaPDF-Merger API will call your given function/method with one parameter:
(string)\$pdfVersion.

The function/method has to return the new, final PDF version.

Keep in mind that a change to a lower PDF version could result in errorious document.

A prototype of a callback function could look like this:

```
function pdfVersion($pdfVersion) {  
    $maxVersion = '1.6';  
    return version_compare($maxVersion, $pdfVersion, '<') ? $maxVersion : $pdfVersion;  
}  
$merger->setPdfVersionCallback('pdfVersion');
```

Version

Available since Version 1.4

SetaPDF_Merger::setFormFieldNamesCallback()

Description

```
SetaPDF_Merger extends SetaPDF {
    void setFormFieldNamesCallback ( callback $formFieldNamesCallback )
}
```

The API renames PDF form fields by default by adding an underscore and the unique file identification number to the end of the form fields name. With this method you can get control of this behaviour by passing a callback function/method to it, which will be called when a form field name is handled.

In the function/method you can create your own logic for renaming the form fields or simply left their names as they are.

Parameters

\$formFieldNamesCallback

A callback function or method which will be called when a name of a form field occurs.

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

The SetaPDF-Merger API will call your given function/method with the following parameter:

- ▶ (string)\$fieldName - Could be in UTF16-BE or PDFDocEncoding/cp1252.
- ▶ (integer)\$currentFileId
- ▶ (string)\$fileName

The function/method has to return the new field name or false to leave the name as it is.

A prototype of a callback function could look like this (default behaviour):

```
function makeUniqueFieldName($fieldName, $fileId, $fileName) {
    // Check for UTF-16
    if (strlen($fieldName) > 1 && substr($fieldName,0,2) === "\xFE\xFF") {
        return $fieldName."_x00_x00".$fileId;
    } else {
        return $fieldName.'_'.$fileId;
    }
}
$merger->setFormFieldNamesCallback('makeUniqueFieldName');
```

If the form field names shouldn't be changed just return false:

```
function makeUniqueFieldName($fieldName, $fileId, $fileName) {  
    return false;  
}  
$merger->setFormFieldNamesCallback('makeUniqueFieldName');
```

Version

Available since Version 1.5

SetaPDF_Merger::merge()

Description

```
SetaPDF_Merger extends SetaPDF {  
    mixed merge ( [string $fileName='doc.pdf'[, string $dest='F'[, boolean  
        $stream=false[, boolean $keepParsers=false]]] )  
}
```

Defines the way the document is to be displayed by the viewer.

Parameters

\$fileName

A valid path and filename for the new document if the *dest*-parameter is set to "F". Otherwise the name of the new PDF document.

\$dest

Defines how the merged documents are 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.

\$keepParsers

If set to true the PDF parser are keepd and are reusable for different *merge()*-calls.

Return Value

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

SetaPDF_Merger::extract()

Description

```
SetaPDF_Merger extends SetaPDF {  
    mixed extract ( mixed $pages[, mixed $bookmark=null[, string  
        $fileName='doc.pdf'[, string $dest='F'[, boolean $stream=false]]] )  
}
```

This method gives you the possibility to use the API for extracting a single page or page range of an existing document.

As the method doesn't close any opened parser you should call the `cleanUp()`-method when you're done.

Parameters

\$pages

In this parameter you can define the page(s) which should be read from the given document.

A single page can be defined as an integer value. A range can be defined as an array with 2 values:

```
array($startPage, $endPage)
```

\$bookmark

This parameter defines the handling of a bookmark entry for this page range/document. There are a few possibilities for this parameter:

1. Pass *null* or omit the parameter will tell the API to resolve the title text for the bookmark entry from the meta data of the merged document.
2. Pass a simple string to this parameter, which will become the title text in the bookmark entry.
3. Pass *false* to prevent a bookmark creation for this page range/document.
4. Pass an array with the following structure:

```
array(  
    'title' => (string) "The bookmark title"  
    'color' => array(red, green, blue) // Text color  
    'format' => SETAPDF_M_BOOKMARK_FORMAT_ITALIC |  
    SETAPDF_M_BOOKMARK_FORMAT_BOLD,  
    'parent' => an Id of a parent node returned by addFile,  
    'copyBookmarks' => SETAPDF_M_COPY_BOOKMARKS_AS_CHILD ||  
    SETAPDF_M_COPY_BOOKMARKS_TO_SAME_LEVEL
```

)

All keys are optional and are described [here](#) with examples and detailed description.

\$fileName

A valid path and filename for the new document if the *dest*-parameter is set to "F". Otherwise the name of the new PDF document.

\$dest

Defines how the merged documents are 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.

Return Value

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

Version

As of Version 1.3

SetaPDF-Merger API - Bookmark handling

The SetaPDF-Merger API offers the facility to create a bookmark outline for the concatenated documents and/or to import the existing bookmark outline(s) of the original documents.

On this page we'll describe the different ways the bookmark feature can be used.

Generally you can turn the bookmark feature on or off with the [SetaPDF_Merger::setAddBookmarks\(\)](#)-method.

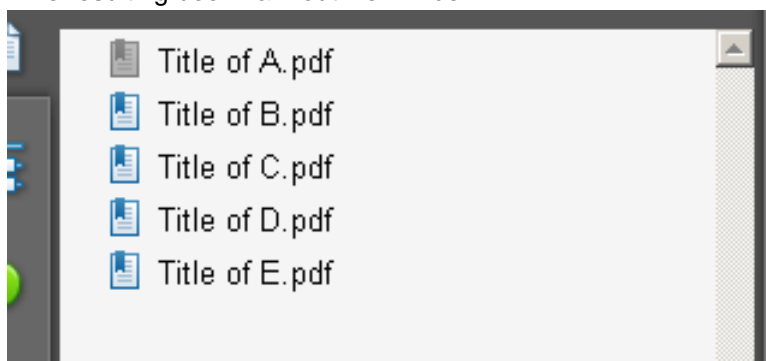
```
$merger = SetaPDF_Merger::factory();
$merger->setAddBookmarks(
    true, // Tells the API that it should create bookmarks
    true // The bookmarks outline should be visible when opening the resulting
document.
);
```

Create new bookmarks for added files

The [SetaPDF_Merger::addFile\(\)](#)-method accepts the \$bookmark parameter which is responsible for the bookmark handling at all. It has a default behavior if it is not set but bookmark creation is set to *true*. Let's see:

```
$merger = SetaPDF_Merger::factory();
$merger->setAddBookmarks(true, true);
$merger->addFile('A.pdf');
$merger->addFile('B.pdf');
$merger->addFile('C.pdf');
$merger->addFile('D.pdf');
$merger->addFile('E.pdf');
$merger->merge();
```

The resulting bookmark outline will be:



As you can see the API has created a bookmark entry for every addFile-call. The bookmark title it has

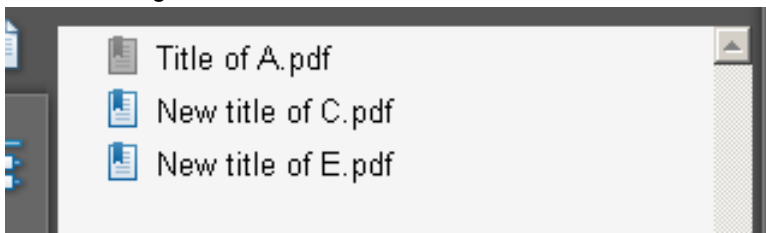
resolved from the title of the used PDF document. If you omit the title at any place the API first tries to resolve a title from the metadata of the pdf document. If the document have no title it'll use the filename instead.

Omit a bookmark entry and define own titles

In some cases a document or page range should not create a bookmark entry. This can be done by simply passing *false* to the \$bookmark-parameter. In the following script we'll give some entries new titles in the bookmark outline and will omit bookmark entries for others:

```
$merger->addFile('A.pdf');
$merger->addFile('B.pdf', false, false);
$merger->addFile('C.pdf', false, 'New title for C.pdf');
$merger->addFile('D.pdf', false, false);
$merger->addFile('E.pdf', false, 'New title for E.pdf');
```

The resulting bookmarks outline will be:



Styling and hierarchic structure

Until now we just used the \$bookmarks parameter in the "easy" mode. The parameter also accepts an array which allows you to style the bookmark entries and arrange them in a hierarchic structure. Let's see:

```
$a = $merger->addFile('A.pdf', false, array(
    'color' => array(255, 0, 0), // red
    'format' => SETAPDF_M_BOOKMARK_FORMAT_ITALIC // italic
));
$merger->addFile('B.pdf', false, false);
$c = $merger->addFile('C.pdf', false, array(
    'title' => 'New title for C.pdf', // the new title
    'color' => array(0, 255, 0), // green
    'format' => SETAPDF_M_BOOKMARK_FORMAT_BOLD,
    'parent' => $a
));
$merger->addFile('D.pdf', false, false);
$merger->addFile('E.pdf', false, array(
    'title' => 'New title for E.pdf',
    'color' => array(0, 0, 255), // blue
```

```
'format' => SETAPDF_M_BOOKMARK_FORMAT_ITALIC |
SETAPDF_M_BOOKMARK_FORMAT_BOLD, // bold and italic
'parent' => $c
));
```

The opened outline will look like this:



title

The new bookmark title. If omitted the behavior is the same as described above.

color

The color-key accepts an array of 3 values representing the RGB components. Values are between 0 and 255.

format

The format key defines if the entry should be displayed in italic and/or bold style. It accepts 2 defined constants:

```
SETAPDF_M_BOOKMARK_FORMAT_ITALIC = italic
SETAPDF_M_BOOKMARK_FORMAT_BOLD = bold
```

They can be combined using a bitwise AND (&) operation.

parent

The parent key accepts a unique id returned by [SetaPDF_Merger::addFile\(\)](#). By passing such id to this key will tell the API that the new entry will be arranged below the entry that created the id.

Copying existing bookmarks

The SetaPDF-Merger API allows you to import existing bookmark outlines of the concatenated pdf documents.

You can control this behavior with the copyBookmarks-key in the \$bookmark parameter array:

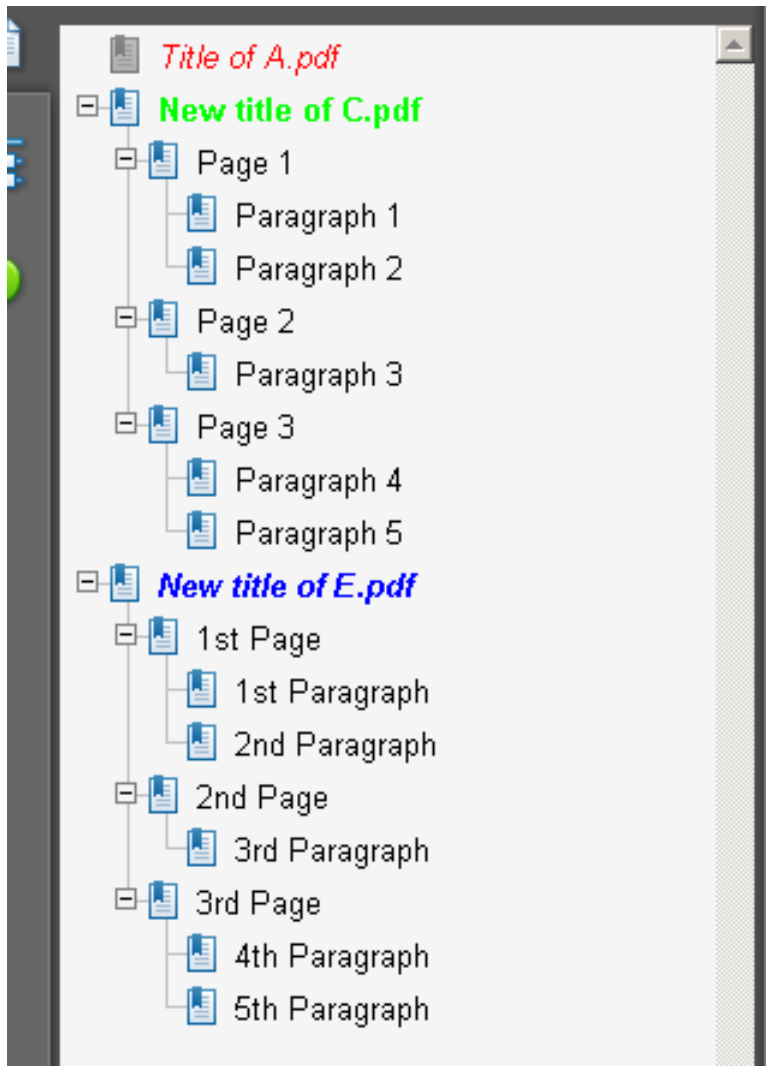
```
// don't copy bookmarks
$merger->addFile('A.pdf', false, array(
    'color' => array(255, 0, 0), // red
    'format' => SETAPDF_M_BOOKMARK_FORMAT_ITALIC // italic
));
// copy bookmarks as child nodes
$merger->addFile('C.pdf', false, array(
```

```

    'title' => 'New title for C.pdf', // the new title
    'color' => array(0, 255, 0), // green
    'format' => SETAPDF_M_BOOKMARK_FORMAT_BOLD,
    'copyBookmarks' => SETAPDF_M_COPY_BOOKMARKS_AS_CHILD
));
// copy bookmarks as child nodes
$merger->addFile('E.pdf', false, array(
    'title' => 'New title for E.pdf',
    'color' => array(0, 0, 255), // blue
    'format' => SETAPDF_M_BOOKMARK_FORMAT_ITALIC |
SETAPDF_M_BOOKMARK_FORMAT_BOLD, // bold and italic
    'copyBookmarks' => SETAPDF_M_COPY_BOOKMARKS_AS_CHILD
));

```

The result:

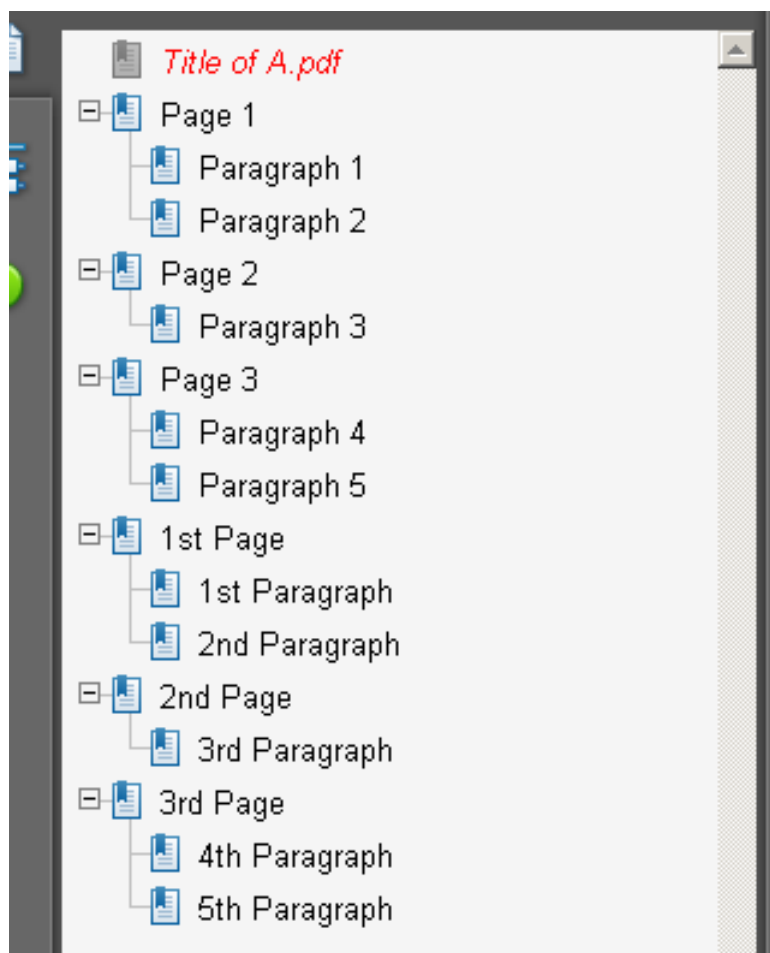


The existing outlines are imported and were appended as childs of the new bookmark entries. This behavior was defined by passing the constant `SETAPDF_M_COPY_BOOKMARKS_AS_CHILD` to the `copyBookmarks`-key.

To omit the creation of such parent nodes you can use the constant `SETAPDF_M_COPY_BOOKMARKS_TO_SAME_LEVEL`, which will only import the existing outline. All other keys, like "title", "color" and "format" were not considered:

```
// don't copy bookmarks
$merger->addFile('A.pdf', false, array(
    'color' => array(255, 0, 0), // red
    'format' => SETAPDF_M_BOOKMARK_FORMAT_ITALIC // italic
));
// copy bookmarks to the same/current level
$merger->addFile('C.pdf', false, array(
    'copyBookmarks' => SETAPDF_M_COPY_BOOKMARKS_TO_SAME_LEVEL
));
// copy bookmarks to the same/current level
$merger->addFile('E.pdf', false, array(
    'copyBookmarks' => SETAPDF_M_COPY_BOOKMARKS_TO_SAME_LEVEL
));
```

The result will look like this:



In this case the outlines were imported to the root node. For sure you also can use this kind of import into your own build hierarchic structure.

SetaPDF-Merger API - Dynamic content and PDF forms

The SetaPDF-Merger API is able to concatenate PDF documents with dynamic content like PDF forms (AcroForms) or any other annotation.

Named Objects

To avoid naming problems all named objects or form field names were renamed. They were simply suffixed with an underscore (for form fields) or a dot (for named objects) and a number (the number of the current processed document).

Internal Links

The API also takes care of internal links.

Internal links will only be written if the document is imported as an entire document with all pages (the `$pages-parameter` of `SetaPDF_Merger::addFile()` has to be set to `false`)

PDF forms (AcroForms)

As already stated form field names will be suffixed with an underscore and a number.

The API is not able to merge pages with form elements from one document a few times! Currently the API didn't recognize that issue, so it's your turn to avoid this!