Thales e-Security PDFlib PLOP DS Integration Guide

This document describes the integration of PDFlib PLOP DS with Thales e-Security HSM (Hardware Security Module) products.

Contact:
PDFlib GmbH
Franziska-Bilek-Weg 9
80339 München, Germany
phone +49 • 89 • 452 33 84-0
support@pdflib.com
www.pdflib.com

Change Log

Version	*Date*	*Editor*	*Notes*
0.1	*2017-01-22*	*Stephan Mühlstrasser*	*Initial draft*
0.2	*2017-03-01*	*Stephan Mühlstrasser*	*Applied changes from internal review*
1.0	*2017-03-15*	*Stephan Mühlstrasser*	*Applied changes from external review*
1.1	*2017-04-27*	*Thomas Merz*	*Mention ECC activation license for the HSM*
1.2	*2017-09-06*	*Thomas Merz*	*Deleted incorrect links*
1.3	*2018-08-28*	*Thomas Merz*	*Minor updates and improvements*

1 Introduction

1.1 Overview of PDFlib PLOP DS

PDFlib PLOP (PDF Linearization, Optimization, Protection) is a versatile tool for linearizing, optimizing, repairing, analyzing, encrypting and decrypting PDF documents. The extended version PLOP DS (Digital Signature) supports all features of PLOP plus the ability to apply digital signatures to PDF documents. PLOP DS supports the following signature standards:

- CMS-based PDF signatures according to ISO 32000-1
- Signatures for Long-Term Validation (LTV) according to ISO 32000-2
- PAdES *(PDF Advanced Electronic Signatures)* according to ETSI TS 102 778 part 2, 3 and 4, ETSI EN 319 142 and CAdES (ETSI TS 101 733)

For PAdES the following conformance levels are supported:

- Basic Signature (PAdES Level B-B)
- Signature with Time (PAdES Level B-T)
- Signature with Long-Term Validation Material (PAdES Level B-LT)
- Signature providing Long Term Availability and Integrity of Validation Material (PAdES B-LTA): required for qualified signatures according to eIDAS
- Basic Electronic Signature (PAdES E-BES) and Explicit Policy-based Electronic Signature (PAdES E-EPES) according to PAdES part 3

PLOP DS provides the following features regarding digital signature creation:

- Signatures according to the RSA and DSA algorithms as well as the Elliptic Curve Digital Signature Algorithm (ECDSA).
- Strong signature and hash functions.
- Embed the full certificate chain in the generated signatures, which means that signatures with certificates from a CA (Certificate Authority) on the Adobe Approved Trust List (AATL) or European Union Trust List (EUTL) can be validated in Acrobat and Adobe Reader without any configuration on the client side.
- Embed Online Certificate Status Protocol responses (OCSP according to RFC 2560 and RFC 6960) and Certificate Revocation Lists (CRL according to RFC 3280) as revocation information for Long-Term Validation (LTV)
- Retrieve a time-stamp from a trusted Time-Stamp Authority (TSA) according to RFC 3161, RFC 5816 and ETSI EN 319 422, and embed it in the generated signature. TSA details can be read from AATL certificates to create time-stamps without any configuration.

PLOP DS functionality can be used in the following ways:

- Via the PLOP DS command line tool *plop* (Unix/Linux) resp. *plop.exe* (Windows).
- As a library from various programming environments: C, C++, .NET, COM, Perl, PHP, Python, Ruby.

All of the features above except DSA signatures are available when combining PLOP DS with a Thales HSM. PLOP DS can use Thales HSMs via the standard PKCS#11 interface for creating digitally signed PDF documents.

The combination of PLOP DS with a Thales HSM provides the following advantages compared to a setup where PLOP DS is used with file-based digital identities:

- Centralized storage and management of the master key(s).

- Full life cycle management of the master key(s).
- Highest level of security assurance, the keys never leave the HSM as plain text.
- FIPS 140-2 level 3 validated hardware.
- Failover support.

1.2 Product Configuration

The integration between PLOP DS and the Thales HSMs uses the PKCS#11 interface. The integration has been successfully tested in the following setups:

Operating System	*nShield Security World Software Version*	*PLOP DS Version*	*nShield Solo/Solo+ support*	*nShield Connect/ Connect+ support*
Windows 10 64-bit	*12.10.00*	*5.2/5.3*	*not tested*	*yes*
Ubuntu Linux 14.04 64-bit	*12.00.00*	*5.2/5.3*	*not tested*	*yes*

The operating systems above represent only a small fraction of systems supported by PLOP DS. Since PLOP DS doesn't depend on OS specific features it works with other versions and other systems as well.

1.3 Supported Thales nShield Functionality

1.3.1 nShield Functions

Function	*Support*
Key Generation	*No*[1]
Key Management	*No*[1]
Key Import	*No*[1]
Key Recovery	*No*[1]
1-of-N Operator Card Set	*Yes*
K-of-N Operator Card Set	*Yes*
Softcards	*Yes*
Module-only Key	*Yes*
Strict FIPS Support	*No*[2]
Load Sharing	*Yes*
Fail Over	*Yes*
Digital Signatures	*Yes*

1. PLOP DS only makes use of keys and certificates stored in the HSM. They have to be managed with Thales or third-party tools.
2. While the Thales nShield hardware is FIPS-validated, PLOP DS is not.

1.3.2 Digital Signature Features

- RSA signatures
- ECDSA signatures with named NIST-15 and Brainpool curves (requires an ECC activation license for the HSM)

1.3.3 Hash Features

PLOP DS can be instructed to offload the hash computation for digital signatures to the HSM.

1.4 This Guide

This guide documents how to use PLOP DS for creating digitally signed PDF documents with certificates and keys that are readily configured in the Thales HSM. PLOP DS itself provides no functionality for creating and managing certificates and keys. How to create and manage certificates in the context of a Certificate Authority (CA) is outside of the scope of this guide. This guide documents how a self-signed certificate can be created for testing purposes with tools provided by the Thales nShield Security World Software.

1.5 More information

For more information about how to use the various digital signature features of PLOP DS refer to the »PLOP and PLOP DS« manual that is included in every PLOP DS package.

»Chapter 9: Application interfaces«, »nCipher PKCS #11 library« in the nShield Connect User Guides for Unix and Windows contains general documentation about the PKCS#11 library of the Thales HSM.

2 Procedures

2.1 Overview

To integrate PLOP DS with a Thales HSM perform the following steps:

- Install PLOP DS
- Install the HSM
- Install the Thales nShield Security World Software and Configure the HSM
- Generate keys and create certificates
- Create digitally signed PDF documents with PLOP DS

2.2 Install PLOP DS

To install PLOP DS:

- Download the appropriate PLOP DS package for your operating system and programming environment from *www.pdflib.com/download/plop-and-plop-ds/* (note: all PLOP DS packages contain the PLOP DS command line tool).
- On Windows run the MSI installer.
- On Unix/Linux extract the compressed tar file.

If the PLOP DS command line tool is to be used, no further preparation is necessary for creating digitally signed PDF documents. If any of the PLOP DS libraries is to be used, a corresponding program that uses the PLOP DS API must be created and potentially compiled.

2.3 Install the HSM

Install the HSM using the instructions in the documentation for the HSM. We recommend that you install the HSM before configuring the nShield software.

2.4 Install the Thales nShield Security World Software and Configure the HSM

To install the nShield support software and configure the HSM:

- Install the latest version of the support software and create a security world as described in the User Guide for the HSM. We recommend that you uninstall any existing nShield software before installing the new software.
- Create or edit the *cknfastrc* file, and depending on how you want to protect the master encryption key, set one of the following environment variables:

OCS or softcard key protection:	*CKNFAST_LOADSHARING=1*
Module-only key protection:	*CKNFAST_FAKE_ACCELERATOR_LOGIN=1*

  The *cknfastrc* file is usually located in the */opt/nfast* directory on Unix/Linux platforms, and in *C:\Program Files (x86)\nCipher\nfast* on Windows platforms.
  For more information, see the PKCS #11 library environment variables in the User Guide for the HSM.
- Initialize a security world.

► For OCS protection, create a *1 of N* card set, following the instructions in the User Guide for the HSM.
Ensure that your Operator Card or softcard pass phrase has a minimum of eight alphanumeric characters. You must create a softcard for softcard protection; see the User Guide for the HSM for more information.

2.5 Generate Keys and Create Certificates

PLOP DS does not have any features for creating and managing keys and certificates, so the setup of keys and certificates that are signed by a Certificate Authority must be performed with third-party tools.

For testing purposes an RSA certificate with a corresponding private key stored in the HSM can be created with the following steps (commands are identical for Unix/Linux and Windows):

► Generate a key in the HSM together with a corresponding self-signed certificate:

```
$ generatekey -g pkcs11 plainname=rsa2048-testcert selfcert=yes certreq=no type=RSA
size=2048 pubexp="" embedsavefile=rsa2048-testcert.pem x509country=DE
x509province=Bavaria x509locality=Munich x509org="PDFlib GmbH" x509orgunit=""
x509dnscommon="" x509email="testcert@test.com" nvram=no digest=sha256
key generation parameters:
 operation      Operation to perform               generate
 application    Application                        pkcs11
 verify         Verify security of key             yes
 type           Key type                           RSA
 size           Key size                           2048
 pubexp         Public exponent for RSA key (hex)
 embedsavefile  Filename to write key to           rsa2048-testcert.pem
 plainname      Key name                           rsa2048-testcert
 x509country    Country code                       DE
 x509province   State or province                  Bavaria
 x509locality   City or locality                   Munich
 x509org        Organisation                       PDFlib GmbH
 x509orgunit    Organisation unit
 x509dnscommon  Domain name
 x509email      Email address testcert@test.com
 nvram          Blob in NVRAM (needs ACS)          no
 digest         Digest to sign cert req with       sha256
Key successfully generated.
Path to key:
/opt/nfast/kmdata/local/key_pkcs11_ua0da0758eddc491f8cae00a853a339bbf128eb68a
```

The certificate is stored in the file *rsa2048-testcert_selfcert.pem* and can be inspected with the *openssl x509* command (output truncated):

```
$ openssl x509 -in rsa2048-testcert_selfcert.pem -noout -text
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 14918394638516426431 (0xcf08c8e1f7dd16bf)
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=DE, ST=Bavaria, L=Munich, O=PDFlib GmbH/
emailAddress=testcert@test.com
```

```
        Validity
            Not Before: Feb 21 13:42:19 2017 GMT
            Not After : Mar 23 13:42:19 2017 GMT
        Subject: C=DE, ST=Bavaria, L=Munich, O=PDFlib GmbH/
emailAddress=testcert@test.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
00:9e:4b:b2:e0:dc:bb:25:a2:76:e3:e8:75:4f:06:
…
```

- Register the certificate in the PKCS#11 environment using the key identifier that was reported as a result of the *generatekey* command in the previous step:

```
$ ckcerttool -n -f rsa2048-testcert_selfcert.pem -k
ua0da0758eddc491f8cae00a853a339bbf128eb68a -L rsa2048-testcert

Certificate found, processing…
2017-02-21 14:59:10 [15988]: pkcs11-sam: 000008cb Warning: Key type CKK_DES3
2017-02-21 14:59:10 [15988]: pkcs11-sam: 000008cb Warning: Allow imported secret key
to be considered secure because CKNFAST_OVERRIDE_SECURITY_ASSURANCES=import set
Certificate successfully imported.
Run cklist to view your certificate object.
OK
```

This test certificate will be referenced with the label *rsa2048-testcert* when creating a signature with PLOP DS.

Note that this certificate does not have any key usage flags set and therefore it will by default not be accepted by PLOP DS for creating PDF signatures. This test certificate can only be used with PLOP DS when specifying the undocumented option *validate= none*.

2.6 Create digitally signed PDF documents with PLOP DS

2.6.1 Overview

For creating a digitally signed PDF document using PLOP DS with a HSM, the following information is required:

- The path to the Thales PKCS#11 library.
- The PKCS#11 label of the certificate in the HSM's PKCS#11 subsystem.
- The names of the input and output documents.
- Optionally additional parameters for PLOP DS signature features.

The PKCS#11 library of the nShield Security World Software is typically installed under the following paths:

- Unix/Linux: */opt/nfast/toolkits/pkcs11/libcknfast.so*
- Windows 32-bit: *C:\Program Files (x86)\nCipher\nfast\toolkits\pkcs11\cknfast-32.dll*
- Windows 64-bit: *C:\Program Files (x86)\nCipher\nfast\toolkits\pkcs11\cknfast-64.dll*

Note that while on Unix/Linux only one variant of the PKCS#11 library is installed that matches the bitness of the operation system, on Windows 64-bit both 32-bit and 64-bit

DLLs are provided. On Windows 64-bit use the appropriate 32-bit or 64-bit DLL that matches the used 32-bit or 64-bit PLOP DS version.

2.6.2 Signature Creation with the PLOP DS Command-Line Tool

Unix/Linux example for creating a digitally signed PDF document *rsa2048-testcert-signed.pdf* from an input document *document.pdf* using the test certificate labeled *rsa2048-testcert* that was created with the steps described in »Generate Keys and Create Certificates«:

```
$ plop --signopt "engine=pkcs#11 digitalid={label=rsa2048-testcert filename=/opt/nfast/
toolkits/pkcs11/libcknfast.so} validate=none" -o rsa2048-testcert-signed.pdf document.pdf
```

The same signature created with PLOP DS 32-bit on a Windows machine:

```
C:\Users\nFast> plop --signopt  "engine=pkcs#11 digitalid={label=rsa2048-testcert
filename={C:\Program Files (x86)\nCipher\nfast\toolkits\pkcs11\cknfast-32.dll}}
validate=none" -o rsa2048-testcert-signed.pdf document.pdf
```

For both examples it is assumed that the PLOP DS command line tool can be found via the PATH environment variable. Note that both commands are intended to be entered as a single line.

The option *validate=none* is necessary because the test certificate doesn't have the required key usage flags that are necessary for creating digital signatures. This option should not be used in a production environment.

2.6.3 Signature Creation using the PLOP DS Library

Apart from the command line tool PLOP DS provides an API that can be used from various programming languages. Each PLOP DS package contains various example programs for the different areas of functionality. The following programs provide blueprints for signing PDF documents using the PLOP DS API:

- *sign:* Sign a single PDF document.
- *multisign:* Perform all preparations for signing and then sign multiple documents in a row with the same digital id

The *multisign* example is specifically useful for signing with a HSM because it demonstrates how to sign multiple PDF documents in the context of a single PKCS#11 session. This is discussed in more detail in »Performance Considerations«.

Note that it is necessary to modify the source code of these example programs before they can be used to create signatures in a specific environment. As a rule of thumb the same command line options that are passed to the PLOP DS command line tool with the *–signopt* option have to be passed to the *PLOP_prepare_signature()* API method.

2.6.4 Hash Computation on the HSM

PLOP DS usually computes the hash for the digital signature in software. The PLOP DS signature option *externalhash=true* instructs PLOP DS to use the PKCS#11 interface to compute the hash. This can be used to have the hash computed on the HSM.

3 Performance Considerations

3.1 Overview

While the PLOP DS command line tool provides an easy out-of-the-box way to digitally sign PDF documents with a HSM, it is not suitable for reaching the maximum performance that is possible. If maximum performance is important, then direct use of the PLOP DS API in a custom program is the better choice.

3.2 PKCS#11 Session Management

The basic sequence of PLOP DS API calls for creating a single digitally signed PDF document is the following:

- *PLOP_prepare_signature( ):* Specify digital id and signature parameters, and validate certificates.
- *PLOP_open_document( ):* Obtain a handle to the input document to be signed.
- *PLOP_create_document( ):* Create a signed output document from the input document.
- *PLOP_close_document( ):* Close the input document.

In the context of signing with an HSM this means that when calling *PLOP_prepare_signature( )* the PKCS#11 session for the HSM is opened. Other potentially time-consuming tasks like certificate validation are also performed in this step.

After a signature context has been prepared, it can be used to call *PLOP_open_document( ), PLOP_create_document( )* and *PLOP_close_document( )* multiple times to sign multiple input documents with the same certificate. This improves performance as the PKCS#11 session setup and other signature preparation tasks need only be performed once for many output documents.

Note that a prepared signature context may expire, for example because the validation information for a certificate expired. In this case *PLOP_create_document( )* returns an error code. As a consequence a new signature context must be set up by calling *PLOP_prepare_signature( )* again before creating more output documents with *PLOP_create_document( )*. The *multisign* sample program demonstrates how to do this.

3.3 Multithreading and Multitasking

It is not possible to reach the maximum signature performance of the HSMs by running the signing procedure in a single process in a single task. In this scenario it is not possible to fully exploit the signing capabilities of the HSM performance-wise. Therefore it is necessary to parallelize the signing procedure, either by using multithreading or by multitasking, or both.

Note that in the case of ECDSA signatures a performance improvement through parallelization is only possible for the named curves that are documented in the nShield Connect User Guide as having »offload acceleration«.

3.4 Conclusion

For reaching the maximum signature performance with the combination PLOP DS and Thales HSM stick to the following guidelines:

- Use a custom program that makes use of the PLOP DS API.
- Minimize calls to *PLOP_prepare_signature( )* and reuse the signature context as long as possible.
- Parallelize signing operations with multithreading and/or multitasking.
- In the case of ECDSA signatures use named curves that are documented to have off-load acceleration.

4 Troubleshooting

4.1 PLOP DS Logging

PLOP DS has extensive logging capabilities that can used for troubleshooting problems. Chapter 8.9 »Logging« in the PLOP DS Manual provides information about how to create log files for PLOP DS operations.

4.2 Thales HSM PKCS#11 Logging

The Thales HSM PKCS#11 library also has extensive logging features. Appendix D: »Logging, debugging, and diagnostics«, »Logging and debugging information for PKCS #11« in the nShield Connect User Guide for Unix resp. chapter Appendix F »Logging, debugging, and diagnostics«, »Logging and debugging information for PKCS #11« in the nShield Connect User Guide for Windows contains information about how to produce log files at the PKCS#11 API level.

A Internet Resources

A.1 Thales Internet Resources

Web site:	*www.thalesesecurity.com*
Support:	*www.thalesesecurity.com/support*
International sales offices:	*www.thalesesecurity.com/about-us/contact-us*

A.2 PDFlib GmbH Internet Resources

Web site:	*www.pdflib.com*
PLOP DS Web site:	*www.pdflib.com/products/plop-ds/features/*
Support:	*www.pdflib.com/licensing-support/support/*