******

DG JUSTICE

Framework Contract No DI/06769-00

Specific Contract No 12960

**Digital Signature Service for DG Justice**

**Configuration, Installation and Operation Manual - Public**



|  |  |
| --- | --- |
| Subject | Configuration, Installation and Operation Manual - Public |
| Version | 1.03 |
| Release Date | 12/03/2015 |
| Filename | DSS4eJustice-CIO-Configuration Installation Operation-Public-v1.03.docx |
| Document Reference | DSS4eJustice-CIOP |

Document History

|  |  |  |  |
| --- | --- | --- | --- |
| Document History | | | |
| Version | Release Date | Author | Description |
| 0.01 | 12/05/2014 | Peter Macfarlane | Initial version created for QTM4 of Specific contract No 7712 under Framework Contract No DI/06769-00 |
| 1.00 | 13/06/2014 | Peter Macfarlane | QTM4 of Specific contract No 7712 under Framework Contract No DI/06769-00  Update according to customer comments |
| 1.01 | 17/06/2014 | Peter Macfarlane | QTM4 of Specific contract No 7712 under Framework Contract No DI/06769-00  Update according to customer comments – mention JDK for build, correct description for enabling additional configuration |
| 1.02 | 25/02/2015 | Peter Macfarlane | QTM4 of SC12960 |
| 1.03 | 12/03/2015 | Peter Macfarlane | QTM4 of SC12960  Implement review comments |

|  |  |  |  |
| --- | --- | --- | --- |
| Reviews | | | |
| Version | Release Date | Reviewer | Position |
|  |  |  |  |

Table Of Contents

[1 Introduction 6](#_Toc413924065)

[1.1 Terminology 6](#_Toc413924066)

[1.2 Purpose of the Document 6](#_Toc413924067)

[1.3 Scope of the Document 6](#_Toc413924068)

[1.4 Intended Audience 6](#_Toc413924069)

[1.5 Structure of the Document 6](#_Toc413924070)

[2 Configuration 7](#_Toc413924071)

[2.1 Card Repository 7](#_Toc413924072)

[2.1.1 Card Profiles 8](#_Toc413924073)

[2.1.2 Unknown Card Behaviour 8](#_Toc413924074)

[2.1.3 Signing Methods 9](#_Toc413924075)

[2.2 Server Seal 10](#_Toc413924076)

[2.3 Validation Configuration 10](#_Toc413924077)

[2.4 Logging 11](#_Toc413924078)

[2.5 HTTP(S) Proxy 12](#_Toc413924079)

[3 Build 13](#_Toc413924080)

[3.1 Install Dependencies 13](#_Toc413924081)

[3.2 Build Digital Signature Applet 13](#_Toc413924082)

[4 Operation 14](#_Toc413924083)

[4.1 Statistics Logging 14](#_Toc413924084)

[4.1.1 Examples of Log Entries 14](#_Toc413924085)

[4.2 Applet Log 15](#_Toc413924086)

[4.3 External Access Log 15](#_Toc413924087)

[4.4 Adding a new Card 15](#_Toc413924088)

[5 Demonstration Application 16](#_Toc413924089)

[5.1 Installing 16](#_Toc413924090)

[5.2 Starting and Stopping the Demonstration Application 16](#_Toc413924091)

[5.3 Open the Application 16](#_Toc413924092)

[5.4 Additional Configuration 16](#_Toc413924093)

[5.4.1 Mocca Support 16](#_Toc413924094)

[5.4.2 Improved Smart Card Support on Windows 16](#_Toc413924095)

[6 Annex 17](#_Toc413924096)

[6.1 Example of libraryPath wildcard searches 17](#_Toc413924097)

List of Tables

[Table 1: Reference Documents 5](#_Toc413924098)

[Table 2: Applicable Documents 5](#_Toc413924099)

List of Figures

**No table of figures entries found.**

Reference and Applicable Documents

This section contains the lists of all reference and applicable documents. When referring to any of the documents below, the bracketed reference will be used in the text, such as [[SAD](#R01)].

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Reference Documents | | | | |
| Ref. | Title | Reference | Version | Date |
| GLO | Digital Signature Service for DG Justice - Glossary | DSS4eJustice-GLO | 1.01 | 31/12/2013 |
| SAD | Digital Signature Service for DG Justice - Technical Analysis - Software Architecture Document | DSS4eJustice-SADP | 1.03 | 12/03/2015 |

Table 1: Reference Documents

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Applicable Documents | | | | |
| Ref. | Title | Reference | Version | Date |
| A01 | Framework Contract No DI/06769-00 | N/A | N/A | 10/09/2010 |
| A02 | Specific contract No 12960 | N/A | N/A | 28/03/2014 |

Table 2: Applicable Documents

# Introduction

## Terminology

The “product name” of the software is the Digital Signature Applet. This consists of a Java applet that runs in a user’s browser. This applet is supported by software that runs on the server. For clarity, we refer to the “signing service” when referring to the software in general, and “signing applet” or “applet” when referring to the applet component. We also use the term “enterprise application” by which we mean the overall application in which the signing service and signing applet are deployed.

## Purpose of the Document

The purpose of this document is to provide information about the configuration, installation and operation of the Signing Service.

## Scope of the Document

The scope of the document is restricted to the configuration, installation and operation of the Signing Service. These aspects are explained within the specific context of the Digital Signature Applet demonstration application.

The document is applicable to version 0.14 of the Digital Signature Applet.

## Intended Audience

The document is addressed to:

* Any technical person having an interest in using the signing service.

## Structure of the Document

The document is organised as follows:

Chapter **1** **Introduction** is this introduction;

Chapter **2** **Configuration** describes configuration of the software;

Chapter **3** **Build** describes how to build the software;

Chapter **4** **Operation** describes some operational aspects of the software;

Chapter **5** **Demonstration Application** describes an application that demonstrates the capabilities of the software.

# Configuration

This chapter describes the configuration of the Signing Service.

## Card Repository

The Signing Service relies on a data store of known smart card profiles, which is currently implemented as an XML file. From time to time it may be necessary to edit the data store, for example to add a new supported card. The data store also allows configuration of the behaviour of the signing service in case an unknown card is used.

The XML file should ideally be edited in a validating XML editor (for example Oxygen, Notepad++, Eclipse) configured to use the XML schema definition (XSD) provided in the [SAD] to avoid simple errors in the file. The following box contains an example of the XML file.

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<SigningContextRepository xmlns:dss=*"eu:europa:ejusticeportal:dynforms:signing:profile:v1"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns=*"eu:europa:ejusticeportal:dynforms:signing:profile:v1"*>

<CardProfiles>

<CardProfile>

<ATR>3B 7D 94 00 00 80 31 80 65 B0 83 02 04 7E 83 00 90 00</ATR>

<Description>LuxTrust</Description>

<URL>http://www.luxtrust.lu/fr/simple/189</URL>

<APIContext>

<API>MSCAPI</API>

<OS>windows</OS>

<arch>either</arch>

</APIContext>

<APIContext>

<API>PKCS11</API>

<OS>linux</OS>

<arch>either</arch>

<libraryPath>/usr/lib/ClassicClient/libgclib.so</libraryPath>

</APIContext>

<DigestAlgorithm>SHA1</DigestAlgorithm>

</CardProfile>

</CardProfiles>

<Defaults>

</Defaults>

</SigningContextRepository>

<SigningMethods>

</SigningMethods>

In the current implementation, the card repository XML file is loaded on each access but parsed only when there is a change. In order to ensure that the modification to the XML file was correctly done, the signing service should be used. XML validation errors will be logged in the application log, however the service will continue to function if the XML is well formed.

### Card Profiles

The card profiles are defined using the *CardProfile* XML element.

A new CardProfile APIContext should not be added to the repository until it has been tested (or, at least the CardProfile APIContext should be tested soon after it has been added). An incorrect Card Profile APIContext will most often result in a failure of the service for that card, whereas if the Card Profile APIContext is simply not present the user may still be able to use the Signing Service.

The ATR in general uniquely identifies a smart card type. However, note that the ATR may in some cases also depend on the operating system (A simple way to obtain the ATR for a card in your possession is to use the Signing Service and look in the applet log (refer to section 4.2). The ATR is written in the log file.

The Description element provides a human readable description of a card in order to identify it in the repository. This is not exposed in the Signing Service user interface.

The URL contains a URL that could give the user assistance with the smart card. This URL is exposed in the Signing Service user interface.

The DigestAlgorithm is the strongest algorithm that is known to work with the smart card. Note that these can be overridden at the APIContext level to allow different algorithms to be used in certain environments.

#### APIContext

The APIContext describes an operating system environment for the card. This is uniquely defined by OS (windows, linux or macos), arch (the bitness, either 32, 64, or either) and the minimum version of the JRE.

The API is the technical mechanism used by the Signing Service to access the card – this can be MSCAPI, NEWSCAPI, PKCS11, or MOCCA. \*MSCAPI is only applicable for Windows OS.

The cards supporting MOCCA are all listed by default in the Card Repository and should only be changed if the supporting API is changed.

The PKCS11 API can be applicable to Windows, Linux or MacOS operating systems. For this API, there must be at least one libraryPath element which gives the default location of the library supporting PKCS11. The libraryPath supports asterisks (\*) as wild cards. If more than one libraryPath is present, the signing service will attempt to find the library by searching the paths in the order they are listed. Similarly, if a libraryPath contains a wildcard and more than one file is matched, the matched files are tried in the order they are discovered. Generally, errors encountered on opening invalid files are not detectable to the user unless they provoke a fatal error in the Java virtual machine. Section 6.1 provides examples of libraryPath search strings.

As stated in section 2.1.1, it is possible to override the DigestAlgorithm defined for the card. An example where this was used is in the Estonian identity card on Linux. The card supports SHA512/RSA on Windows (MSCAPI) and on MacOS (PKCS11), however on Linux only the MOCCA API is supported, with the weaker SHA1 digest algorithm.

### Unknown Card Behaviour

The signing service is also able to deal with unknown cards (i.e.: not listed in the CardProfiles), for which the hash algorithm is not known.

Two strategies have been implemented to try to discover a working hash algorithm, namely *LIST* and *RANDOM*.

When configured to use the *LIST* strategy, the signing service will try to sign using the digest algorithms listed in the Defaults element, in the order they are listed, until signing succeeds.

<Defaults>

<Strategy>LIST</Strategy>

<DigestAlgorithm>SHA512</DigestAlgorithm>

<DigestAlgorithm>SHA256</DigestAlgorithm>

<DigestAlgorithm>SHA1</DigestAlgorithm>

</Defaults>

When configured to work with the RANDOM strategy, the signing service will try to sign with a randomly selected digest algorithm, until signing succeeds, or it reaches the specified maximum of attempts, at which point it will make a final attempt to sign with the default algorithm.

<Defaults>

<Strategy>RANDOM</Strategy>

<MaxTries>4</MaxTries>

<DefaultDigAlgorithm>SHA1</DefaultDigAlgorithm>

<DigestAlgorithm>MD2</DigestAlgorithm>

<DigestAlgorithm>MD5</DigestAlgorithm>

<DigestAlgorithm>SHA512</DigestAlgorithm>

<DigestAlgorithm>SHA256</DigestAlgorithm>

</Defaults>

### Signing Methods

Each supported signing method is configured in the XML file. Each configured method appears as a separate entry in the list of signing methods presented to the user. The following box gives as an example the configuration for the Smart Card signing method.

<!-- Smart card-->

<SigningMethod>

<needsJavaScript>true</needsJavaScript>

<needsJava>true</needsJava>

<code>sc</code>

<jspPage>dynform\_dss\_applet.jsp</jspPage>

</SigningMethod>

The following parameters are defined:

* needsJavaScript is used to disable the choice of the method when JavaScript is not enabled in the browser.
* needsJava can be used to warn the user that Java is required to use the method.
* code is a unique code for the method. It is used to get translations/descriptive text in a generic manner;
* jspPage is the JSP that is associated with the signing method.

## Server Seal

Each PDF document provided by the Signing Service to the signing service user may be signed by the signing service[[1]](#footnote-2).It is possible to disable this behaviour in the configuration file (refer to section 2.3).

The signing certificate is stored in a PKCS12 file. The location of this certificate file is given to the signing service by the “dss\_applet\_server\_seal\_cert” system property. Should it be necessary to change the signature applied to the PDF documents, this certificate must be replaced. The signing service must be re-deployed for the change to take effect. The password of the PKCS12 file is stored in a file that must contain only one line; the password must be disguised for casual readers by base 64 encoding. The location of the password file is given to the signing service by the “dss\_applet\_server\_seal\_pwd” system property.

Note that the signing service validates the origin of uploaded PDF documents by referencing this certificate. Depending on the validation configuration (refer to section 2.3), documents provided by the signing service before a change of certificate may prompt a validation warning or error if they are uploaded after the change of certificate.

The certificate used to seal the PDF is also used to secure the communication between the applet and the signing service. The server uses the private key to sign each outgoing communication, while the DSS Applet has the public key to verify the signature

## Validation Configuration

The validation of the uploaded, signed PDF document includes five steps. It is the responsibility of the enterprise application to provide the validation configuration to the signing service, via the DocumentValidationConfig interface. A default implementation is DefaultDocumentValidationConfig, which takes the configuration from the dss\_validation.properties from the classpath. The contents of this file are reproduced below, and are self-explanatory. T

##The URL for the root of the trusted lists

LOTL\_URL=https://ec.europa.eu/information\_society/policy/esignature/trusted-list/tl-mp.xml

##How often the trusted list cache in the Signing Service is refreshed, in seconds

LOTL\_REFRESH\_PERIOD=3600

##Test if we should check the signatures of each certificate in the trusted list.

CHECK\_LOTL\_SIGNATURE=false

##The properties controlling the validation of the digital signature.

##Possible values are

## E (Exception - prevents upload)

## W (Warning - does not prevent upload)

## D (Disabled - the validation is not done)

##Check if the PDF originated on the enterprise application

ORIGIN=D

##Check if the PDF belongs to the current workflow

WORKFLOW=D

##Check if the PDF was tampered with

TAMPERED=D

##Check if the signing certificate is trusted

TRUSTED=D

##Check if the signing certificate is revoked

REVOKED=D

##Check that the user has signed the PDF

SIGNED=E

##check that the signature format is OK (this means that the signature format has the right

##pieces of information that allow us to validate.

SIGNATURE\_FORMAT=D

##Check if the signature has expired

EXPIRED=W

##Seal method to be used.

##Possible values are:

## C (sealPDFCustom)

## S (PAdES seal method)

## N (no seal)

## It is strongly recommended to set this to N

SEAL\_METHOD=N

## Logging

The Signing Service server-side components use slf4j logging where logger names correspond to the simple class name in which the logger is declared. All classes (and the corresponding loggers) belong in the eu.europa.ejusticeportal.dss.controller package.

In addition to these debug/error loggers, there are some special loggers:

* eu.europa.ejusticeportal.dss.controller.statistics.sign
  + Logs attempts to sign a document
* eu.europa.ejusticeportal.dss.controller.statistics.repo
  + Logs access to the card repository
* eu.europa.ejusticeportal.dss.controller.statistics.newprofile
  + Logs information that might be sufficient to make a new card profile
* eu.europa.ejusticeportal.dss.controller.statistics.error
  + Records errors reported by the applet
* eu.europa.ejusticeportal.dss.controller.statistics.applet
  + Records the applet log (when the user agrees to upload it)
* eu.europa.ejusticeportal.dss.controller.CommonsHttpLoaderFactory.external
  + Logs access by the signing controller to external URLs

## HTTP(S) Proxy

The following box contains the configuration parameters for the HTTP(S) proxy that can be enabled for the Signing Service. When enabled, the proxy will be used to contact the trusted lists and certificate revocation lists servers during digital signature validation.

It is the responsibility of the enterprise application to provide the proxy configuration to the signing service via the HttpProxyConfig interface. A default implementation DefaultHttpProxyConfig takes the information from a properties file in the class path (file dss\_http\_proxy.properties).

##This file configures internet proxy for signature validation by the dss service.

HttpEnabled=FALSE

HttpHost=

HttpPassword=

HttpPort=

HttpUser=

HttpsEnabled=FALSE

HttpsHost=

HttpsPassword=

HttpsPort=

HttpsUser=

# Build

The Digital Signature Applet is provided as a maven project.

## Install Dependencies

The project has dependencies on the DSS project[[2]](#footnote-3) that is currently not available in a public maven repository. The jar and pom files are included in the folder dss-dependencies. Install each of these files in your local maven repository by executing the command[[3]](#footnote-4):

mvn install:install-file -Dfile=<path-to-file> -DpomFile=<path-to-pomfile>

as well as

mvn install:install-file -Dfile=<path-to-file> -DgroupId=eu.europa.ec.joinup.sd-dss -DartifactId=<artifactId> -Dversion=4.0.4 -Dpackaging=pom

for the two pom files sd-dss-4.3.0.pom and sd-dss-app-4.3.0.pom.

## Build Digital Signature Applet

The parent digital-signature-applet pom.xml file contains parameters that give the location and access credentials of a certificate store that will be used to sign the applet jars during the build. These should be adjusted for the local environment. Note that a self-signed certificate suitable only for testing is provided in the certificate folder of the distribution.

Then, the project can be built by running

mvn clean install -Psign-jars -Dmaven.test.skip

Note that JDK version 1.7 or 1.6 must be used to build the application, and building on Windows requires the 32 bit version.

# Operation

## Statistics Logging

The Signing Service collects usage statistics, in particular client configurations where there was no suitable Card Profile in the repository (refer to section 2.1) and client configurations where signing was possible with additional input from the user.

Collected information should inform the creation of new Card Profiles in the repository therefore it is important to keep the log files at least until they have been analysed. The location of the log files is defined in the logging configuration file of the enterprise application.

### Examples of Log Entries

#### Repo Log

The following box contains an example of a “repo” log entry. Such an entry is created whenever the card repository is queried for a specific ATR and, if no card was detected, whenever the applet user chooses to sign with a smart card.

SC 37F46F88FBD9D32C2387E3F18CD532E9 [Mar 14, 2014 3:22:27 PM] arch=32 os=windows jre=1.705 ATR=3B DD 96 FF 81 B1 FE 45 1F 03 80 31 B0 52 02 03 64 04 1B B4 22 81 05 18 found=true fingerprint=Fingerprint [userAgent=mozilla/5.0 (windows nt 6.1; wow64) applewebkit/537.36 (khtml, like gecko) chrome/33.0.1750.146 safari/537.36, navPlatform=win32, jreVendor=Oracle Corporation, jreVersion=1.7.0\_51, osVersion=6.1, arch=x86, os=windows 7, cardDetectionAvailable=true, cardProfiles=[CardProfile [api=MOCCA, digestAlgo=null, libraryPath=null, cardDescription=null, url=null, atr=3B DD 96 FF 81 B1 FE 45 1F 03 80 31 B0 52 02 03 64 04 1B B4 22 81 05 18, eventFilters=null, terminalIndex=0, synthetic=false]]]

From left to right, the log entry components are: fixed text (SC); the unique session identifier; the date/time of the log entry; the architecture calculated from the fingerprint; the operating system calculated from the fingerprint; the JRE version of the applet; the ATR used to query the card store (“null” if no card was detected); true if a card was found in the store for the ATR/OS/architecture, false otherwise; the fingerprint.

#### Sign Log

The following box contains an example of a “sign” log entry. Such an entry is created whenever the applet user attempts to sign the PDF document.

SG 37F46F88FBD9D32C2387E3F18CD532E9 [Mar 14, 2014 3:22:45 PM] arch=32 os=windows jre=1.705 newProfile=false signed=true method=sc atr=3B DD 96 FF 81 B1 FE 45 1F 03 80 31 B0 52 02 03 64 04 1B B4 22 81 05 18 api=MOCCA digAlgo=SHA1 sigAlgo=ECDSA keyUsage=digitalSignature nonRepudiation issuer=null path=null cardCount=1 error=null extensions= etsi\_qcs\_QcCompliance, recommended=true

From left to right, the log entry components are: fixed text (SG); the unique session identifier; the date/time of the log entry; the architecture calculated from the fingerprint; the operating system calculated from the fingerprint; the JRE version of the applet; true if the card is not in the card store and there could be enough information to make a new entry in the card store; true if signature was successful, false otherwise; the signature method (smart card, P12, other); the ATR used to sign (if applicable and if known); the API (MSCAPI, PKCS11, MOCCA, PKCS12); the digest algorithm; the signature algorithm; the key usage flags (space separated if there is more than one); the issuer of the card (if provided by the applet user); the path to the PKCS11 library (if provided by the applet user); the number of detected cards; the error code (if there was an error), the certificate extensions, true if the applet recommended the certificate.

## Applet Log

The Signing Service applet logs events and exception conditions on the applet console. The console can be enabled from the Control Panel (windows), Settings (MacOS) or the appropriate tool in Linux. The “Show Console” and “Enable logging” option in the advanced properties tab must be selected.

## External Access Log

The Signing Service will log each access to an external URL in a dedicated log file. The location of the log file is defined in the logging configuration file of the application.

## Adding a new Card

In order to add support for a new card, a new CardProfile element must be added to the CardRepository (refer to section 2.1). The easiest way is to copy an existing element and adapt it to the new card. The information required can be obtained from the Sign Log after the card has been tested with the system.

At the same time the help page should be updated. Currently, the list of supported cards is maintained in the TestedCards\_en.jsp file. Copy an existing entry and adapt it to the new card. If there is a helpful URL with more information about the card, this should also be included in the CardRepository so that the user can get specific help for his/her card.

Changes to the card repository are picked up immediately, but if you update the help page you will have to redeploy the application.

# Demonstration Application

The Digital Signature Applet has a demonstration application for Windows consisting of a Tomcat instance containing, in the webapps folder, a war file.

## Installing

Unzip the package and copy the digital-signature-applet-webapp-full.zip file to the installation folder. Unzip this file to that folder.

## Starting and Stopping the Demonstration Application

Start the demonstration application by double-clicking the bin/Webapp-Startup.bat file.

Stop the demonstration application by double-clicking the bin/ Webapp-Shutdown.bat file.

Note that Tomcat is configured to run on the default 8080 port.

## Open the Application

To open the application, double-click the DigitalSignatureAppletWeb link in the installation folder, or directly enter the address <http://localhost:8080/digital-signature-applet-webapp/choosefile.html> in your browser.

## Additional Configuration

Some jar files are provided under special licenses - refer to the digital-signature-applet-licences.zip file in the package.

### Mocca Support

Some cards in the repository are configured to use the MOCCA API. In order to add support for this, copy the following jars from the digital-signature-applet-extras.zip file to the root of the digital-signature-applet-webapp.war file:

* iaik\_jce\_me4se-\*-signed.jar
* smcc-\*-signed.jar jars
* ej-portal-dss-mocca\*.jar
* sscd-mocca-adapter\*.jar

Then re-deploy the war.

### Improved Smart Card Support on Windows

When using smart cards on Windows, you may be prompted to insert a smartcard even when your card is inserted. This can be due to a previous use of a smart card that has left the certificate entry in the Windows certificate store. For better handling:

1. Copy the ej-portal-dss-mscapi\*.jar files from digital-signature-applet-extras.zip file to the root of the digital-signature-applet-webapp.war file.
2. Edit the SigningRepo.xml file in the WEB-INF/classes folder of this war file by replacing all occurrences of MSCAPI with NEWMSCAPI.
3. Redeploy the war.

# Annex

## Example of libraryPath wildcard searches

This section provides some examples of libraryPath wildcard searches for the following folder/file structure:

C:\USERS\MACFARPE\DSS\_APPLET\_TEST

\---aaa

| pkcs11.so

|

\---bb bb

\---cc ccc

| pkcs11.so

|

\---ddd dd

library.dll

---------------------------------------------------

Search: C:/Users/macfarpe/dss\_applet\_test/aaa/bb bb/cc ccc/ddd dd/library.dll

Find:

C:\Users\macfarpe\dss\_applet\_test\aaa\bb bb\cc ccc\ddd dd\library.dll

---------------------------------------------------

---------------------------------------------------

Search: C:/Users/macfarpe/dss\_applet\_test/aaa/bb bb/cc ccc/ddd dd/lib\*.dll

Find:

C:\Users\macfarpe\dss\_applet\_test\aaa\bb bb\cc ccc\ddd dd\library.dll

---------------------------------------------------

---------------------------------------------------

Search: C:/Users/macfarpe/dss\_applet\_test/aaa/bb bb/cc ccc/ddd dd/lib\*y.\*dl\*

Find:

C:\Users\macfarpe\dss\_applet\_test\aaa\bb bb\cc ccc\ddd dd\library.dll

---------------------------------------------------

---------------------------------------------------

Search: C:/Users/macfarpe/dss\_applet\_test/aaa/bb bb/cc ccc/ddd\*d/library.dll

Find:

C:\Users\macfarpe\dss\_applet\_test\aaa\bb bb\cc ccc\ddd dd\library.dll

---------------------------------------------------

---------------------------------------------------

Search: C:/Users/macfarpe/dss\_applet\_test/a\*a/bb bb/cc\*/d\*dd\*d/library.dll

Find:

C:\Users\macfarpe\dss\_applet\_test\aaa\bb bb\cc ccc\ddd dd\library.dll

---------------------------------------------------

---------------------------------------------------

Search: C:/Users/macfarpe/dss\_applet\_test/\*/\*/\*/\*/library.dll

Find:

C:\Users\macfarpe\dss\_applet\_test\aaa\bb bb\cc ccc\ddd dd\library.dll

---------------------------------------------------

---------------------------------------------------

Search: C:/Users/macfarpe/dss\_applet\_test/\*/\*/\*/\*/\*.\*ll

Find:

C:\Users\macfarpe\dss\_applet\_test\aaa\bb bb\cc ccc\ddd dd\library.dll

---------------------------------------------------

---------------------------------------------------

Search: C:/Users/macfarpe/dss\_applet\_test/library.dll

Find:

---------------------------------------------------

---------------------------------------------------

Search: C:/Users/macfarpe/dss\_applet\_test/\*\*/pkcs11.so

Find:

C:\Users\macfarpe\dss\_applet\_test\aaa\bb bb\cc ccc\pkcs11.so

C:\Users\macfarpe\dss\_applet\_test\aaa\pkcs11.so

---------------------------------------------------

*End of Document*

1. Note that the signature is not a standard signature – it is an encrypted hash of the PDF added as a PDF attachment. [↑](#footnote-ref-2)
2. <https://joinup.ec.europa.eu/asset/sd-dss/description> [↑](#footnote-ref-3)
3. <http://maven.apache.org/guides/mini/guide-3rd-party-jars-local.html> [↑](#footnote-ref-4)