



## **Integration Guide**

### **Adobe LiveCycle Document Security Server and nCipher Modules**

#### **Configuring an nCipher Hardware Security Module with:**

- Windows 2003 Enterprise Server



These installation instructions are intended to provide step-by-step instructions for installing nCipher software with third party software. These instructions do not cover all situations, and are intended as a supplement to the nCipher documentation provided with nCipher products. Disclaimer: nCipher Corporation Ltd disclaims all liabilities regarding third party products and only provides warranties with its own products as addressed in the nCipher Terms and Conditions for Sale. nCipher is a registered trademark of nCipher Corporation Limited. Any other trademarks referenced in this document are the property of the respective trademark owners. ©2006 nCipher Corporation Ltd, Cambridge, United Kingdom.

# CONTENTS

|   |          |
|---|----------|
| <b>1.0 Technical Overview .....</b>   | <b>3</b> |
| 1.1 Supported nCipher Functions .....   | 3        |
| <b>2.0 Requirements.....</b>  | <b>3</b> |
| <b>3.0 Installing and Configuring the Hardware and Software .....</b>             | <b>4</b> |
| 3.1 Installing the ALDS Server.....   | 4        |
| 3.2 Configuring the ALDS Server to use the nCipher Hardware Security Module ..... | 4        |

## 1.0 Technical Overview

The nCipher Hardware Security Module (HSM) and netHSM integrate with the Adobe LiveCycle Document Security (ALDS) server by using the industry standard interface PKCS#11 to conduct all cryptographic operations.

Functionality provided by ALDS:

- Document signing:
  - Batch processing to sign PDF documents on the server
  - Document recipients can validate authenticity and integrity
- Signature validation by the sever of signed PDFs from a user
- Document encryption, PDFs can be automatically encrypted for secure distribution

There are several benefits to using nCipher HSM or netHSM with the ALDS:

- FIPS 140-2 level 3 validated hardware
- Improved server performance by offloading the cryptographic processing
- Enables the secure storage of the signing and encryption keys
- Enables the management of the full key life cycle, including
  - Secure backup and archiving of the keys
  - Secure disaster recovery

### 1.1 Supported nCipher Functions

|   |  |  |  |
|---|--|--|--|
| <input type="checkbox"/> Soft Cards           | <input checked="" type="checkbox"/> Key Management | <input checked="" type="checkbox"/> Strict FIPS    | <input checked="" type="checkbox"/> Key Recovery |
| <input type="checkbox"/> Module Only Key      | <input checked="" type="checkbox"/> 1ofN Card Set  | <input checked="" type="checkbox"/> Key Generation | <input type="checkbox"/> Key Import              |
| <input checked="" type="checkbox"/> Fail Over | <input type="checkbox"/> Fall Back                 | <input type="checkbox"/> Load Balancing            | <input type="checkbox"/> Preload                 |

## 2.0 Requirements

It is important to familiarize yourself with the Adobe LiveCycle Document Security Server documentation and setup, and have the nCipher user documentation available.

Before attempting to install the software, consider the following aspects of HSM administration:

- Administrator Card Set (ACS) k-of-n and management of the card set
- Type of protection for application keys, module or Operator Card Set (OCS) protected
- Operator Card Set 1-of-n and management of the card set
- Any requirements for a FIPS 140-2 level 3 Security World
- Key attributes – key size, persistence, time out etc.
- The need for auditing key usage

For more information on any of the above points refer to the nCipher User Guide.

Integration between nCipher HSM and the Adobe LiveCycle Document Security server have been tested with the following combinations:

| Operating System                 | ALDS Version | nCipher Version | PCI support | Ethernet support |
|----------------------------------|--------------|-----------------|-------------|------------------|
| Windows 2003 Enterprise Server   | 7.0          | 10.16           | Yes         | Yes              |
| Windows 2003 R2 Standard Edition | 7.1          | 10.15           | Yes         | Yes              |

### 3.0 Installing and Configuring the Hardware and Software

The installation is performed in several steps:

- Install the nCipher hardware
- Install the nCipher software
- Creating the nCipher Security World
- Install and configure the ALDS Server

Information on the installation of hardware can be found in the Hardware Installation Guide. Similarly, information on software installation and security world creation can be found in the User Guides. These guides are available from [www.ncipher.com/documentation](http://www.ncipher.com/documentation). The installation procedures outlined in this document assume, for example purposes, that the user is installing an offline root Certificate System. It is assumed that a new root key is generated during installation rather than a software key being imported from an existing installation.

#### 3.1 Installing the ALDS Server

Before installing the ALDS, make sure that j2sdk version 1.4.2.x and j2eesdk version 1\_4\_xx are installed and the environment variables are properly set.

1. Install ALDS by executing the jar file “LiveCycleDocumentSecurityInstall.jar” that starts the installation wizard.
2. Create a new keystore when prompted in the installation wizard. Name the keystore, use an appropriate alias and password for the keystore.

#### 3.2 Configuring the ALDS Server to use the nCipher Hardware Security Module

To configure the nCipher PKCS#11 interface to ALDS server follow the below steps:

1. Use the nCipher command line utility, `generatekey`, to generate both the key and the Certificate Signing Request (CSR) using the nCipher module. The command output below shows the user entries in ***bold italics***. Note the entries that are specific to the certificate request.

```
c:\nfast\bin>generatekey pkcs11 certreq=yes enter

protect: Protected by? (module, token) [token] > enter
cardset: Operator cardset to protect key? (isa ocs, pkcs11adobe) [pkcs11adobe]> enter
recovery: Key recovery? (yes/no) [yes] > enter
size: Key size? (bits, minimum 1024) [1024] > enter
OPTIONAL: pubexp: Public exponent for RSA key (in hex)? []>enter
embedsavefile: Filename to write key to? []> key-file-name
plainname: Key name? [] > key-file-name
x509country: Country code? [] > us
x509province: State or province? [] > ma
x509locality: City or locality? [] > woburn
x509org: Organisation? [] > adobe test
x509orgunit: Organisation unit? [] > test
x509dnscommon: Domain name? [] > adobe.test.com
```

```

x509email: Email address? [] > adss@test.com
nvrnm: Store blob in NVRAM (will require administrator cardset)? (yes/no) [no]> enter
key generation parameters:
operation      Operation to perform      generate
application    Application                embed
protect        Protected by              token
slot           Slot to read cards from   0
cardset        Operator cardset to protect key pkcs11adobe
recovery       Key recovery              yes
verify         Verify security of key   yes
type           Key type                  RSA
size           Key size                  1024
pubexp         Public exponent for RSA key (in hex)
embedsavefile  Filename to write key to  adss
plainname      Key name                  adss
x509country    Country code              us
x509province   State or province        ma
x509locality   City or locality         woburn
x509org        Organisation              adobe test
x509orgunit    Organisation unit        test
x509dnscommon  Domain name              adobe.test.com
x509email      Email address            adss@test.com
nvrnm          Store blob in NVRAM (will require administrator cardset) no
Please enter the passphrase for card 1:
Enter-OCS-passphrase-here
Key successfully generated.
Path to key: C:\nfast\kmdata\local\key_pkcs11_uc1e52352c15fa0108fd1656e5de347b6b57032cb3-34a41e3abfab1cd7eeb15ec2ba0e37403b34de81

```

2. After key generation key details are displayed as below:

```
"Path to key: C:\nfast\kmdata\local\key_pkcs11_uc1e52352c15fa0108fd1656e5de347b6b57032cb3-34a41e3abfab1cd7eeb15ec2ba0e37403b34de81"
```

The part beginning with “uc...” is the key-id that will be used later (in step 4) for importing the resulting certificate. This ensures that the certificate is associated with the right private key.

3. The PKCS#11 '<key-file-name>\_req' CSR file created in “C:\nfast\bin” may now be submitted to the Certificate Authority of choice for approval. When the signed certificate is received back, proceed to the next step.
4. Import the resulting certificate file (in Base64 format) with `ckcerttool` utility as below:

```
ckcerttool -c <ocs-name> -f <cert-file> -k <key-id> -L <key-alias-name>
```

The <key-id> parameter above refers to the key-id in step 2.

**Note:** The `ckcerttool` utility that comes with the standard release has a bug and an updated `ckcerttool` utility is needed to import the certificate successfully. Please contact [support@ncipher.com](mailto:support@ncipher.com) for the updated patch.

5. The ALDS configuration file must now be edited to include specific information about the nCipher PKCS#11 provider and the signed certificate. In the <ALDS\_Root>\trust directory, where ALDS\_Root refers to the ALDS installation directory, copy the certificate obtained (in DER format) to the certificates subdirectory.
6. Edit `trust.xml` and add a record to the `trustAnchors` section for the certificate:

```
<trustAnchors>
```

```
<cerrecord cerFile="nCipherSelfSignCert.cer" TrustedFor="" />
</trustAnchors>
```

7. Add a record for the signing certificate under the credentials section in the trust.xml file:

```
<credentials>
<hsmrecord alias="nCipherSelfCert" slot="492971158" dllpath="C:\nfast\toolkits\pkcs11\cknfast.dll"
sha1="dcdd7a5a11337a095f5d234a7ecc55b62ac4c30e"/>
</credentials>
```

- The alias is arbitrary, it is used by the signing code to identify which key to use for signing.
- The slot value is the internal slot number for the nCipher PKCS#11 library and must be 492971158.
- The sha1 value is the thumbprint of the signing certificate. (Open the certificate in Windows, the thumbprint is listed on the Details tab)

### Sign and verify the trust.xml file

The private key is used for signing, and the public key is for validation (or verification). Each time the content of the trust.xml is modified, the file should be resigned.

To sign the trust.xml file:

1. At the command prompt, navigate to the <ALDS\_Root>\trust directory, where SignTool is located.
2. Type the following text to run SignTool in sign mode:

```
java -jar SignTool.jar -sign -trustfile trust.xml -signaturefile
trust.sig -keyalias credential -kspath keystore.jks -keypass password
```

1. To verify the trust.xml file type the following command:

```
java -jar SignTool.jar -verify keystore -trustfile trust.xml
-signaturefile trust.sig -keyalias credential -kspath keystore.jks
```

Note that the alias and password must match those used when the keystore was created during the installation process.

### Updating the Trust Manager module

1. Run <ALDS\_Root>\configtool\common\configtool.bat to start the configuration tool.
2. In the configuration tool, open <ALDS\_Root>\deploy\adobe-TrustManager.bar and highlight it in the list.
3. On the Properties tab, confirm that the 'Path to Public Key...' refers to the keystore.jks created during installation and that the alias matches the one provided during installation.
4. On the Resources tab, click on each of trust/trust.xml, trust/trust.sig and trust/keystore and click the import button below to import the corresponding file from the <ALDS\_Root>\trust directory.
5. Under Collections on the right panel, highlight Certificates, and then use the import button below to import the <ALDS\_Root>\trust\certificates directory.
6. Save the file and exit.
7. Copy adobe-TrustManager.bar from <ALDS\_Root>\deploy folder to <ALDS\_Root>\jboss-3.2.5\server\all\deploy folder and restart the Adobe Document Security service.

## DSE200 Timestamp Integration

1. A working DSE200 (Document Sealing Engine) must be correctly configured to accept timestamping requests.
2. It is necessary to configure trust.xml to enable timestamping during the signing process. Edit trust.xml to modify the existing credential record to include a TimestampURL reference:

```
<credentials>  
<hsmrecord alias="<alias>" slot="492971158" dllpath="c:\nfast\toolkits\pkcs11\cknfast.dll"  
sha1="<thumbprint-signing-cert>" TimestampURL<url>/>  
</credentials>
```

3. Follow the information provided in: "Sign and verify the trust.xml file" and "Updating the Trust Manager module".
4. As opposed to stating the following under signature properties:

"Signature data/time are from the clock on the signer's computer"

It should now state (it is necessary to assign the timestamp service as a Trusted Identity):

"Signature is timestamped"

Viewing Date / Time of the signature should provide a date and name of the appropriate Timestamp Authority.

The Adobe LiveCycle Document Security server configuration with nCipher HSM and DSE200 is now complete and ready for use.