# Key Manager

Table of Contents



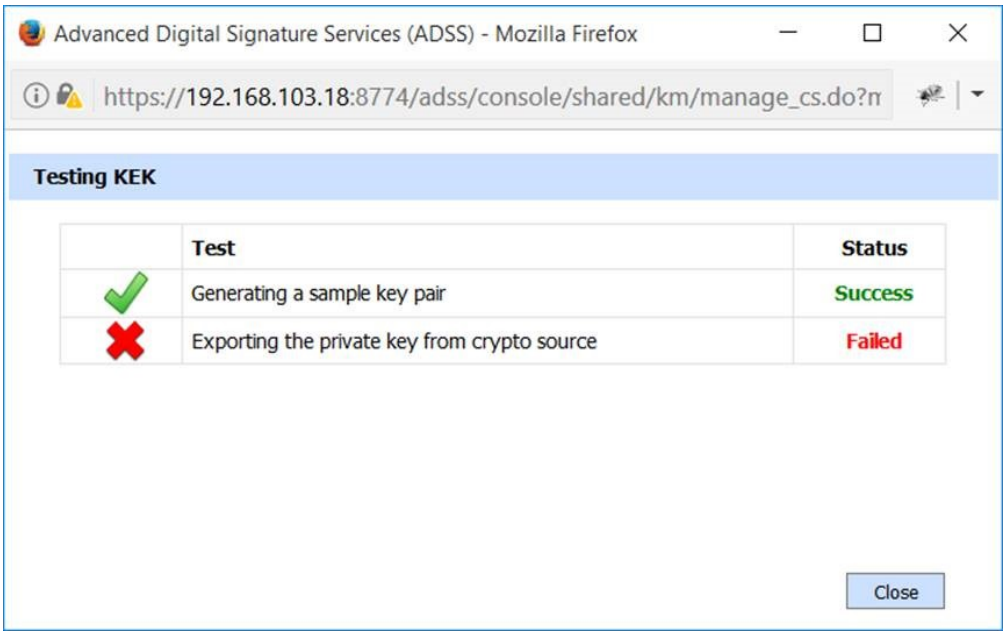
## How to configure ADSS Server to use a Key Encryption Key (KEK)?

Key Encryption Key (KEK) is a symmetric key that is used to encrypt the keys generated within the crypto device. The encrypted keys are saved outside the crypto device (HSM), and are then imported into the crypto device again when required (e.g. signing operation). After this, the encrypted keys are discarded. To use KEK in ADSS Server:

1. Configure an HSM crypto profile within ADSS Key Manager and select the "Enable Key Wrapping for Document Signing Keys" option for this HSM.
    a. Select a KEK from the already available KEKs in HSM. The system automatically tests and verifies whether the HSM supports key wrapping/ unwrapping or not.
    b. Use the HSM provided utilities and tools, to create or backup the KEK. See details to learn more about configuring a crypto profile in ADSS Server.
2. Create a "Certification Profile" in ADSS Certification Service, and select the crypto profile for which you have selected the "Enable Key Wrapping for Document Signing Keys" option in step 1.
3. Send a request to ADSS Certification Service to generate a new certificate, by using the profile Id of the same "Certification Profile" created in step (2). Also specify the unique "Certificate Alias" in the request by which the encrypted private key and certificate can be identified at the time of signing. This works as follows:
    a. ADSS Certification Service requests the HSM to generate a key pair according to the "Certification Profile" settings.
    b. ADSS Certification Service requests the HSM to wrap (encrypt) and export the private key using the KEK configured in crypto profile.
    c. ADSS Certification Service stores the exported encrypted private key in database against the "Certificate Alias" provided in the request.
4. Request the ADSS Signing Service to sign a document using the encrypted private key. This works as follows:
    a. Specify the "Certificate Alias" for the encrypted private key with which the document needs to be signed.
    b. The ADSS Signing Service loads the encrypted private key from the database and imports the signing key into HSM. While importing the key, it also specifies the "Certificate Alias" for the private key and the KEK to be used to unwrap (decrypt) the private key within the HSM.
    c. The ADSS Signing Service produces the final hash of the document and requests the HSM to sign it using the private key identifying it by the Certificate Alias.
5. Once the document hash is signed, the private key is deleted from the HSM. The key is not deleted from database so that it can be used later to sign other documents.

### Special handling for Thales nCipher HSM

When applying a KEK generated from the nCipher HSM via ADSS Server Console > Key Manager > [nCipher-Crypto-Source], if you get the error "Exporting the private key from crypto source" function fails - as shown below;




Check the CKNFASTRC config file (default path is C:\Program Files (x86)\nCipher\nfast) of the HSM for following line:

```
CKNFAST_NO_UNWRAP=1
```

This essentially means that the HSM will not allow to unwrap the key resulting in the private key export failure. You will need to change this attribute value to 0; allowing the HSM to unwrap the key and for ADSS Server to set the KEK.

```
CKNFAST_NO_UNWRAP=0
```

## Can I attach more than one HSM to an ADSS Server?

Multiple HSMs, USB tokens and smartcards can be attached and configured as allowed by the device driver software. ADSS Server expects a hardware crypto profile for each specific HSM (unless the software driver provides load balanced access to cloned HSMs). PKCS#11 and Windows CAPI/CNG drivers are supported, although Windows CAPI/CNG HSMs have the restriction that ADSS Server cannot generate new keys within them, Java is unable to tell CAPI/CNG which CSP to use when generating a key so a keygen request is only ever processed in the default software store.
This link discusses how a new hardware crypto profile can be created

## Can keys and certificates that already exist in an HSM or within the Windows CAPI/CNG store be used by ADSS Server?

ADSS Server does allow certificates that exist in an HSM or Windows CAPI/CNG store to be imported into ADSS Server. Only the certificates are imported and as they are imported they must be assigned a certificate purpose. This link describes how to import existing certificates:
http://manuals.ascertia.com/ADSS-Admin-Guide/default.aspx?pageid=Importing_existing_keys

## How to replace an existing HSM with a new one?

If a business need arrives to replace the configured HSM with a new one, then the following aspects should be considered. It is recommended to read the complete article first before you go with one of the recommended options.

**Retain Old Keys:**

It is generally required and recommended to first replicate the existing keys and certificates available in the current HSM into the new one. Use the HSM vendor utilities to achieve this. Refer to the relevant manuals from the HSM vendor to find instructions for this or ask for the technical help from the HSM vendor. If the key wrapping is being used then it is very important that the back-up and restore must include the KEK as well as any other key material.

Now replace the current HSM with the new one as per HSM vendor instructions. While doing this, if the new HSM is also from the same vendor and can use the existing drivers, then simply go to ADSS Server Key Manager > Crypto Source and perform a test connection with the HSM to make sure it is connectable. The steps 3 and 4 can be ignored in this case.

If the new HSM is from a different vendor or if it should use a newer version of the HSM drivers, then first install the new HSM drivers on the ADSS Server machine to be able to communicate with the new HSM.

If the new HSM is from a different vendor, then you will also need to create a new Crypto Source on ADSS Server Key Manager > Crypto Source. The following additional steps will also be needed:

1. If there were some existing infrastructure keys (service keys) on the old HSM e.g. verification response signing key, OCSP response signing key or timestamp response signing key, issuing CA keys etc., then it will be first required to temporarily switch these keys with dummy keys created in the software keystore (ADSS Server database) to be able to remove the existing keys from the ADSS Server Key Manager > Service Keys
2. Now from the Key Manager > Service Keys, delete the relevant service keys that exist in the old Crypto Source.
3. Once the new crypto profile is created, edit this profile and click on the Import Existing Keys button
4. From the list of the keys available to import from the new HSM, select all the relevant keys with their associated key purposes and import these keys
5. Now, you can go back and replace the dummy service keys created in step a with the newly imported keys and restart the ADSS Server from Server Manager module

**Do not Retain Old Keys:**

This is not a recommended approach but if it is a business need, then simply switch any service keys with dummy ones created in the software keystore (ADSS Server database) to be able to remove the existing keys from the ADSS Server Key Manager > Service Keys

Once the keys are removed, detach the current HSM and attach the new one. If the new HSM is of the same brand and can use the existing drivers, you should be done here. Simply go to ADSS Server Key Manager > Crypto Source and perform a test connection with the HSM to make sure it is connectable.

If the new HSM is from a different vendor or if it should use a newer version of the HSM drivers, then first install the new HSM drivers on the ADSS Server machine to be able to communicate with the new HSM.

If the new HSM is from a different vendor, then you will also need to create a new Crypto Source on ADSS Server Key Manager > Crypto Source

Now create new keys and associated certificates on the new crypto profile and configure it in relevant services or Manage CAs module accordingly

Restart the ADSS Server from Server Manager module

It is highly recommended **NEVER** to detach an HSM without removing the associated keys from the ADSS Server Key Manager > Service Keys as this results in losing the references of these keys on the HSM. If you still get to this situation by mistake or an unwanted accident, then follow these additional instructions to remove the stale keys from the ADSS Server Key Manager > Service Keys:

Launch ADSS Server console

Navigate to location: **Global Settings > Advanced Settings**

From the **Property Type** drop down select the option **Console**

Search for the property **ENABLE_PKCS11_KEY_DELETION** and change the value of this property to **FALSE**

**Save** the changes

**Restart** the ADSS Server console instance from Windows NT Services panel or UNIX daemon to have the changes take into effect

Go to Key Manager and delete the unwanted keys which were generated in the old HSM

**Follow** the **steps 2-6** to set the value of **ENABLE_PKCS11_KEY_DELETION** back to **TRUE**

Launch ADSS Server console

Navigate to location: **Global Settings > Advanced Settings**

From the **Property Type** drop down select the option **Console**

Search for the property **ENABLE_PKCS11_KEY_DELETION** and change the value of this property to **FALSE**

**Save** the changes

**Restart** the ADSS Server console instance from Windows NT Services panel or UNIX daemon to have the changes take into effect

Go to Key Manager and delete the unwanted keys which were generated in the old HSM

**Follow** the **steps 2-6** to set the value of **ENABLE_PKCS11_KEY_DELETION** back to **TRUE**

## What is the relationship between options to manipulate Certificate Templates in the Key Manager, Manage CAs and Certification Service modules?

Certificate Templates are used to define which certification extensions should be included in the certificates being issued under different certificate purposes. Certificates can be requested within Key Manager, Manage CAs and Certification Service modules and hence the templates can be edited from any of these modules. Updated Certificate Templates are stored centrally in ADSS Server and are available to all three service modules.

## What are Certificate Groups and when should these be used?

Certificate groups are valuable when one or more keys may be unavailable to an instance of ADSS Server. Such a situation may occur when PCIe HSMs, USB HSMs, tokens or smartcards are used, e.g. one token is attached to one server and another token attached to a second server.

When ADSS Server runs on each server it will identify the first certificate AND key that is available to it and use this for all requests on this server instance. A key located on the other server will therefore never be chosen. Click here for more details.

## ADSS Server connection is lost with Utimaco after sometime of inactivity

If you are using the Utimaco simulator and you observe that the ADSS Server loses the connection with Utimaco then follow these instructions for a fix:

1. Search for file **cs_pkcs11_R2.cfg** in **Program Files** (e.g. C:\Program Files\Utimaco\CryptoServer\Lib\cs_pkcs11_R2.cfg) and open it in edit mode
2. Locate the property KeepAlive and set its value to true (**KeepAlive = true**)
3. Save the file and restart the Utimaco Simulator for the changes to take effect
4. Now **restart** the ADSS Server from **Server Manager** so it can establish the connection with HSM.

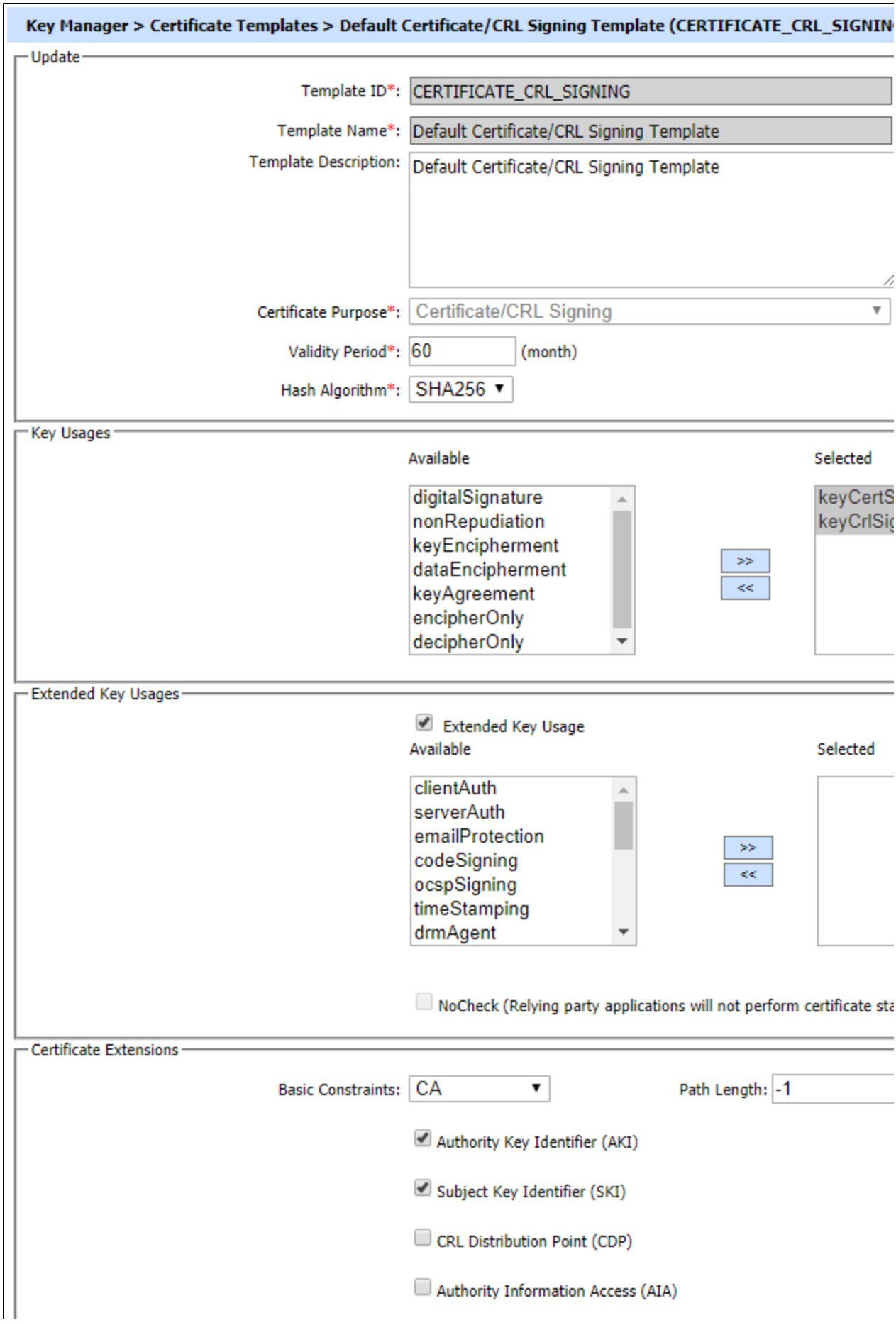
You will not face the disconnection issue again.

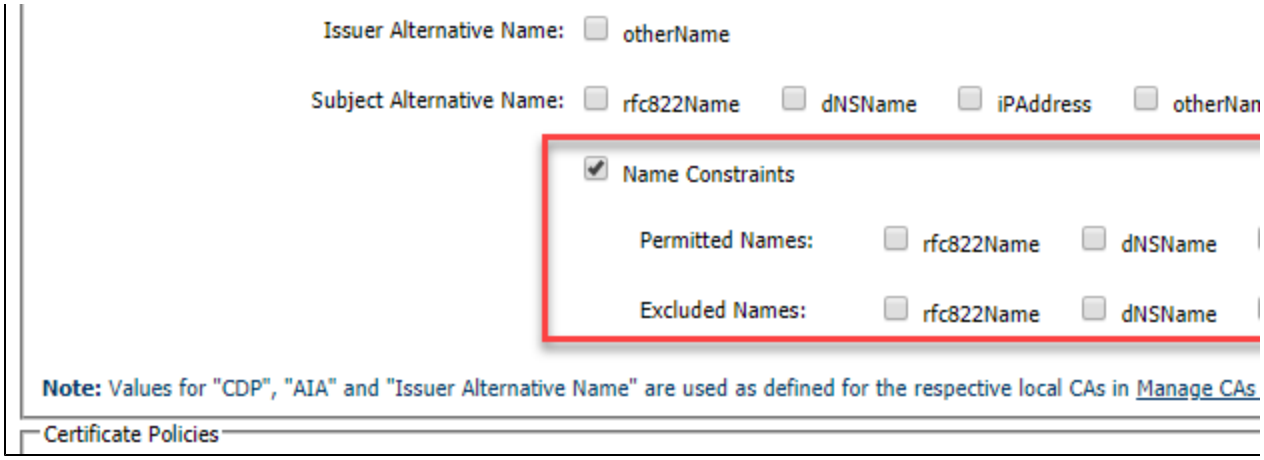
## How to enable Name Constraints extension to a CA certificate?

The Name Constraints extension is used in CA certificates and it specifies the constraints that apply on Subject DN and Subject Alternative Names of subsequent certificates in the certificate path. If a CA certificate contains this extension, all the subsequent certificates can have only the permitted names in Subject DN and Subject Alternative Names and certificates that don't comply with name constraints set by CA would not be generated by ADSS Server. Follow these steps to add Name Constraints extension to CA certificates in ADSS Server:

### 1- Enable Name Constraints extension in Certificate Template

The Name Constraints extension can be added to CA certificates (not self-signed) only, so first step is to enable this extension in the relevant Certificate Template. Create a new certificate template with purpose **Certificate/CRL Signing** or update an existing one with the same purpose. Enable the Name Constraints extension in the template as shown in the image below:


Key Manager > Certificate Templates > Default Certificate/CRL Signing Template (CERTIFICATE_CRL_SIGNIN
Update
Template ID*: CERTIFICATE_CRL_SIGNING
Template Name*: Default Certificate/CRL Signing Template
Template Description: Default Certificate/CRL Signing Template
Certificate Purpose*: Certificate/CRL Signing
Validity Period*: 60 (month)
Hash Algorithm*: SHA256
Key Usages
Available
Selected
digitalSignature
nonRepudiation
keyEncipherment
dataEncipherment
keyAgreement
encipherOnly
decipherOnly
keyCertS
keyCrlSig
>>
<<
Extended Key Usages
Extended Key Usage
Available
Selected
clientAuth
serverAuth
emailProtection
codeSigning
ocspSigning
timeStamping
drmAgent
>>
<<
NoCheck (Relying party applications will not perform certificate sta
Certificate Extensions
Basic Constraints: CA
Path Length: -1
Authority Key Identifier (AKI)
Subject Key Identifier (SKI)
CRL Distribution Point (CDP)
Authority Information Access (AIA)




The Name Constraints consist of two types of lists:

- Permitted Names
- Excluded Names

**Permitted Names:**

This list contains the names that will be permitted in subsequent certificates. The following name types can be added in the list:

- Rfc822Name
- dNSName
- iPAddress
- Directory Name

Only those name forms can be added to Name Constraints extension while generating a CA certificate that will be enabled in the template.

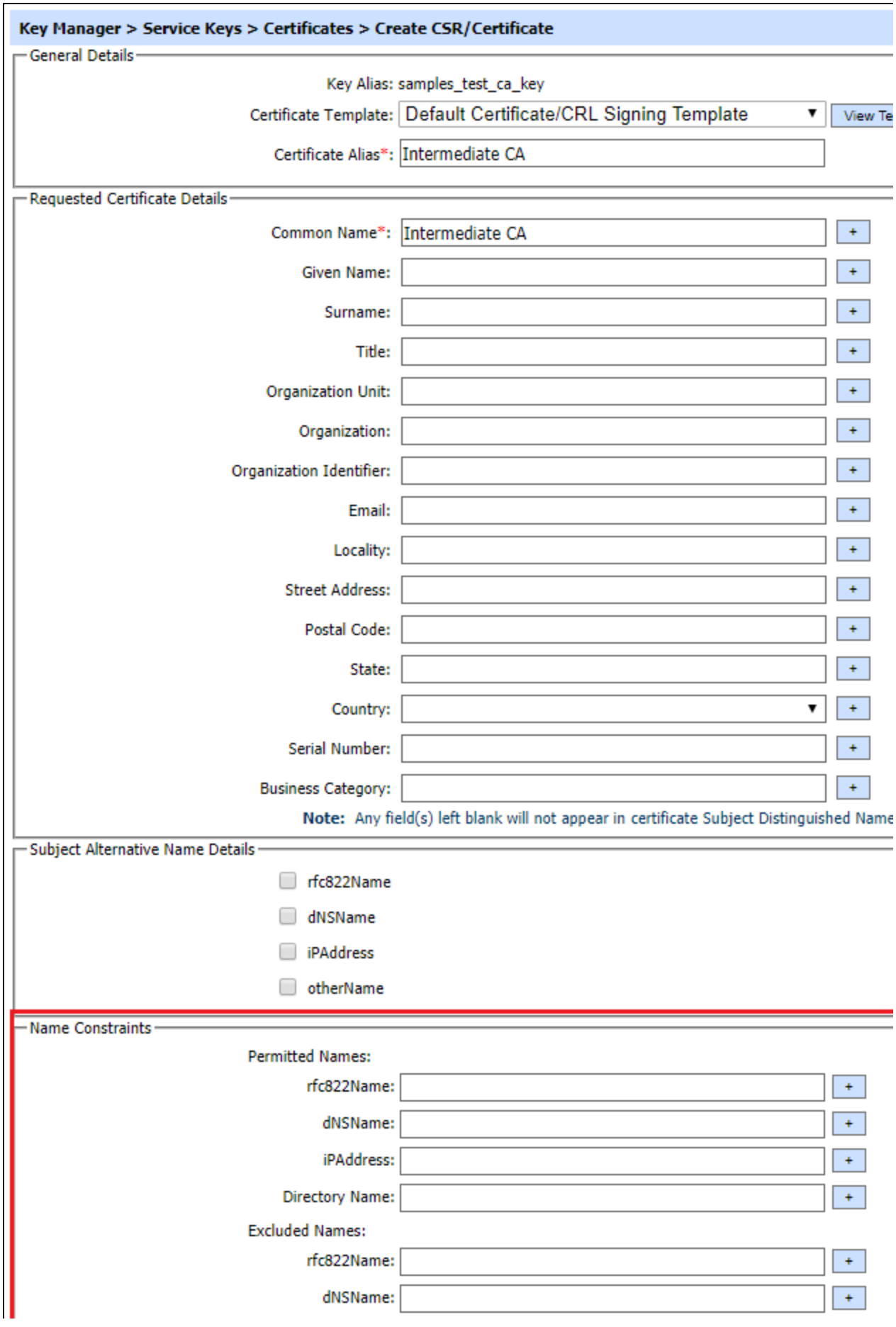
**Excluded Names:**

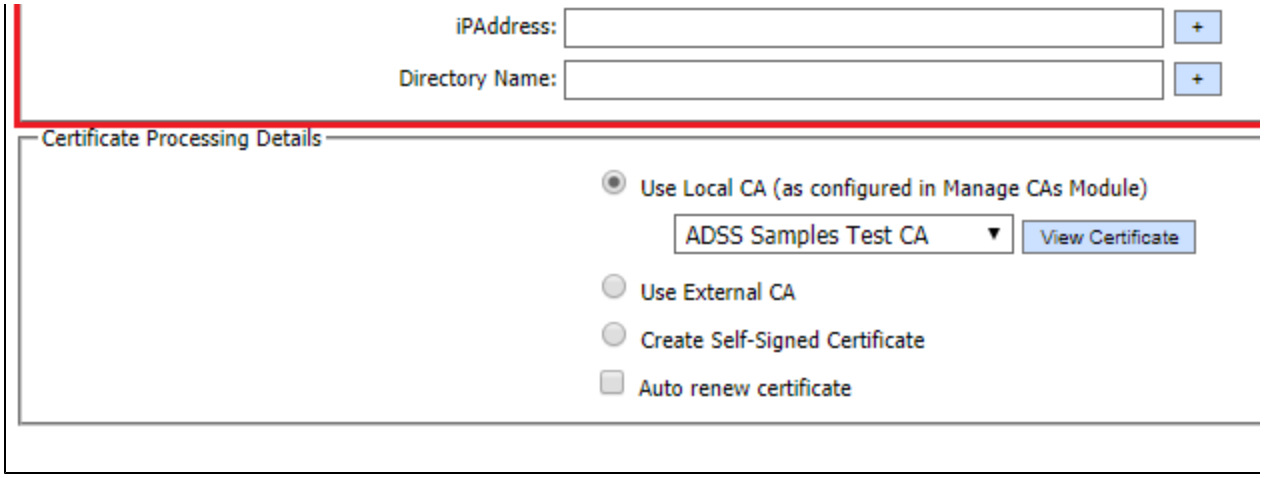
This list contains the names that CA wants to prohibit and cannot appear in subsequent certificates. The name forms selected in template for excluded names will be added to the Name Constraints extension while generating the CA certificate.

**2- Create a CA certificate with a Name Constraints extension**

Once the Name Constraints extension is enabled in the Certificate Template, the next step is to generate a CA certificate that will contain the Name Constraints extension. A subordinate CA certificate can be generated either via sending a request to ADSS Certification Service or on ADSS Console using Key Manager module. However, the option to add the Name Constraints extension to a CA certificate is only provided via Key Manager module on ADSS Console.

a. Launch ADSS Server Console and navigate to Key Manager > Service Keys
b. Select a key-pair with purpose **Certificate/CRL Signing** and view its certificates by clicking on **Certificates** button.
c. On next screen, click on **Create CSR/Certificate** button to create the certificate. The following screen will be displayed:


Key Manager > Service Keys > Certificates > Create CSR/Certificate
General Details
Key Alias: samples_test_ca_key
Certificate Template: Default Certificate/CRL Signing Template
View Te
Certificate Alias*: Intermediate CA
Requested Certificate Details
Common Name*: Intermediate CA
Given Name:
Surname:
Title:
Organization Unit:
Organization:
Organization Identifier:
Email:
Locality:
Street Address:
Postal Code:
State:
Country:
Serial Number:
Business Category:
Note: Any field(s) left blank will not appear in certificate Subject Distinguished Name
Subject Alternative Name Details
rfc822Name
dNSName
iPAddress
otherName
Name Constraints
Permitted Names:
rfc822Name:
dNSName:
iPAddress:
Directory Name:
Excluded Names:
rfc822Name:
dNSName:




In the image above, there is a highlighted section **Name Constraints** that contains fields and controls to add the Permitted and Excluded Names in the Name Constraints extension. Note that this section will be only visible if Name Constraints extension will be enabled in the selected Certificate Template. Only those name types can be added here that will be allowed in template e.g. if only dNSName is allowed in template for permitted and excluded names then fields related to dNSName will appear on the screen only. Also note that the Name Constraints section will not be visible while generating self-signed certificates as this extension is only added to the subordinate CAs. Different name forms can be provided here for permitted and excluded names. After providing the required names in the permitted or excluded lists, create the certificate and the Name Constraints extension will be added to certificate.

**3- Issuing Certificates from a CA containing Name Constraints extension:**

While issuing certificates (either via Certification Service or Console) from a CA that contains Name Constraints extension in its certificate, the ADSS will make sure the target certificate contains names in Subject DN or Subject Alternative Name according to the permitted and excluded names mentioned in the Name Constraints extension of the CA certificate.

For example, if Name Constraints extension contains a dNSName e.g. **alpha.com** in permitted list. The sub-sequent certificates can have only **www.alpha.com** as dNSName in the Subject Alternative Name extension. For subdomains, if **.alpha.com** is set in the permitted list, the sub-sequent certificates can contain **www.sub.alpha.com** or **www.any.alpha.com**.

For excluded names, if **bravo.com** is set in the Excluded Names list of CA certificate, the subsequent CAs cannot have this domain e.g. **www.bravo.com** in their Subject Alternative Name extension. Same rule applies to restrict sub-domain i.e. if **.bravo.com** is set in the Excluded Names list, sub-domains like **www.sub.bravo.com** or **www.adv.bravo.com** would not be allowed.

**Note:** Name Constraints rules only apply to a target certificate when the certificate contains a particular name type that is also mentioned in the Name Constraints extension e.g. if a **dNSName** is added to permitted or excluded names in CA certificate but target certificate does not contain any dNSName in its Subject Alternative Name extension, then the rules for dNSName would not be applied to target certificate and certificate will be issued without any issues.