# Root CA certificate renewal

**Update 22.10.2017:** updated use-case recommendations based on best practices.

**Update 27.06.2018:** added commands

---

In this article I will discuss about Root CA certificate renewal with new and existing key pair. At first we discuss about CA certificate renewal with existing key pair.

## Renewal with existing key pair

When you renew CA certificate with existing key pair, nothing important in certificate is changed. The certificate will contain the same public and private key. As the result all previously issued certificates will chain up to new CA cert without any changes. You just replace old CRT file in AIA download locations. In addition, new CA cert ValidFrom (NotBefore) field will contain the value when existing CA key pair was generated. For example, old CA cert has **ValidFrom** (**NotBefore**) = 08.10.2000 and **ValidTo** (**NotAfter**) 08.10.2010. When you renew CA cert with existing key pair new certificate will have following values: **ValidFrom** (**NotBefore**) 08.10.2000 and **ValidTo** (**NotAfter**) 08.10.2020. In other words this renewal just increases current CA certificate validity period. In addition new CA cert introduces one new extension: **Preious CA certificate hash** that will contains preious certificate Thumbprint extension value. And changes another extension: **CA Version**. Let's take a look to a CA Version extension.

**CA Version** extension allows to build correct chains in the case when particular CA has more than one certificate. This extension consist of two values: **CA Certificate Index** and **CA Key Index**. These values are separated by dot, for example: 0.0, 2.1, 3.3, etc. Each time you renew CA certificate (regardless with existing or new key pair), **CA Certificate Index** is increased by 1: 0.0, 1.0, 2.0, etc. Since the key pair remains the same, the **CA Key Index** value is not changed. In that case CA will maintain the same CRL's and clients will be able to chain previously (prior to CA cert renewal) and newly (after CA cert renewal) issued certificates up to new CA certificate. This is because all these client certificates was signed by the same CA signing key and both CA certs produces the same signature for the identical data.

> After CA cert renewal, new CA cert will not replace previous CA cert, but is another file and adds a certificate index in parenthesises in the file name. For example old cert has a name: **TestCA.crt**, and new cert will have the following name: **TestCA(1).crt**. The number and parenthesises are controlled by **<CertificateName>** variable in AIA location settings and the number always equals to certificate CA Version extension **CA Certificate Index** value (except when you setup new CA. In that case <CertificateName> is ignored).

As you see, this renewal is quite simple. However, this method is not suitable when one or more of the following cases occur:

- CA signing (existing CA key pair) is compromised;
- You have a program that requires a new signing key to be used with a new CA certificate;
- The current CRL file is too large and you want to move some revocation information to a new CRL file.

Run the following command on CA server to renew CA certificate and reuse existing key pair:

```
certutil -renewCert ReuseKeys
```

## Renewal with new key pair

As we have discussed previous scenario is Ok for most scenarios. However there might be a requirement to renew CA certificate with a new key pair. This renewal type is more complex. Since new key pair is generated many things in the CA cert are changed. For example new public key will produce different **Subject Key Identifier** (the hash of public key). When CA issues new certificate it places own certificate **Subject Key Identifier** value to a issued certificate **Authority Key Identifier** extension. Actually these extension comparison is used by certificate chaining engine (CCE). As the result previously issued certs will chain up to previous CA cert and newly issued certs will chain up to new CA cert respectively. In addition new CRL is generated. New CRL will contain only those revoked certificates that were signed using renewed CA cert (or signing key) and new CRL file will contain CRL suffix. For example, old CRL has **TestCA.crl** name and new CRL will have new name: **TestCA(1).crl**. This CRL suffix is controlled by **<CRLNameSuffix>** variable in CDP location settings and the number always equals to certificate CA Version extension CA Key Index value.

> Unlike CA Certificate Index value, the CA Key Index is not always increased by 1, but is set to CA Certificate Index value. For example, previous CA cert has CA Version extension as 2.0 and new CA cert CA Version extension will have the following value: 3.3.

To address this issue (when you use new root CA cert, but it is not deployed to all clients yet) Windows CA generates two cross-certificates. First cross-certificate is signed by previous CA signing key and certifies new CA certificate. Certification direction is determined by numbers in parenthesises. In our case one cross-cert will have **(0-1)** suffix. Here is an example how this works during certificate chain building:

1) **PreviousCACert**
    **LeafPreviousCertificate**

2) **PreviousCACert**
    **cross-cert(0-1)**
        **LeafNewCertificate**

As you see in these example, by using cross-certificate CCE will be able to construct certification paths for previously and newly (after CA cert renewal) issued certificates, so both paths chains up to only previous CA certificate (because new CA cert is not deployed yet). In order to chain both paths to a new CA certificate (when new CA cert is deployed and you are ready to remove old CA cert from clients) additional cross-certificate is generated. In that case new CA cert certifies previous CA cert (reverse direction). This direction is shown in the file name parenthesises: **(1-0)**. And here are examples how this works:

3) **NewCACert**
  **LeafNewCertificate**

4) **NewCACert**
  **cross-cert(1-0)**
    **LeafPreviousCertificate**

As you see by using these cross-certificates you can maintain only one root CA certificate with ability to build correct chains for any certificate issued by this CA (before and after CA cert renewal with new key pair).

In order to use these cross-certificates you must publish them in you Active Directory forest by running the following commands:

```
certutil –dspublish –f <CAName(0-1).crt> CrossCA
certutil –dspublish –f <CAName(1-0).crt> CrossCA
```

> Replace <CAName(#-#).crt> with actual file names.

Once you have deployed new CA certificate to clients (it MUST be published to Trusted Root CAs container on client computer) you may remove previous CA certificate from clients if they are not required to validate digital signatures.

If you are able to deploy new CA cert to all client computers very quickly, you may ignore these cross-certificates or instruct CA server to not generate them by running the following command:

```
certutil -setreg ca\CRLFlags +CRLF_DISABLE_ROOT_CROSS_CERTS
```

Run the following command on CA server to renew CA certificate and create new key pair:

```
certutil -renewCert
```

## Recommendations

Although key reuse is simple from maintenance overhead perspective, it can cause unpredictable and unfortunate flaws in certificate chain building, because existing certificates will now produce two separate chains: one chain through previous CA certificate and second through renewed CA certificate. Certificate chaining engine will have to do an extra work to select the best chain. Different cryptographic libraries have different metrics and rules used when selecting best chain. However these metrics are not compatible between libraries and behavior may be truly unpredicted. Also, there might be an issue within a single crypto library: KB329433, KB2639180 and KB2831004 are examples of such issues in Microsoft CryptoAPI certificate chaining engine.

As the result, modern best practices dictate that CA keys **SHALL NOT BE REUSED** to avaoid potential issues when wrong chain is selected. New key pair ensures only single certification path for each certificate and eliminates the issue with multiple chains and guarantees expected behavior.

---

This is 3 tier PKI hierarchy -- Root(offline) -> **Intermediate (offline) CA** -> Issuing (online) CAs

---