

# private Key aus P12 Datei

---

*Source:*

[http://www.tech-archive.net/Archive/German/Entwicklung/microsoft.public.de\\_german.entwickler.dotnet.csharp/2008](http://www.tech-archive.net/Archive/German/Entwicklung/microsoft.public.de_german.entwickler.dotnet.csharp/2008)

---

- *From:* "Andre Rust" <[andre.rust@xxxxxxxx](mailto:andre.rust@xxxxxxxx)>
  - *Date:* Wed, 8 Oct 2008 08:10:13 +0200
- 

Hallo,  
ich brauche Hilfe zu folgendem.

Ich habe eine P12-Datei, die ich mit X509 auslese. Wie kann ich den privaten Schlüssel bekommen, so das ich danach eine Signierung machen kann. Ich bin schon die ganze Zeit am testen, aber dieer Schlüssel läßt sich nicht auslesen. In einer PEM-Datei kann ich den sehen. Nur lesen klappt auch hier nicht. Der Private Schlüssel sollte als XML gespeichert sein.

Danke.

André R.

Hier der Code den ich benutze:

HIER AUSLESEN:

```
private void GetAllKeys()
{
    X509Certificate2 x509 = new X509Certificate2();
    X509KeyStorageFlags KeyStorageFlags = new X509KeyStorageFlags();

    if (mReferenceData.SignatureFile.Length < 1)
    {
        mKeyClass.x509_Error = "File name not found";
        return;
    }

    try
    {
        rawData = ReadFile(SignatureFile);

        x509.Import(rawData, mReferenceData.Password,
            KeyStorageFlags);

        mKeyClass.x509_Subject = x509.Subject;
        mKeyClass.x509_Issuer = x509.Issuer;
        mKeyClass.x509_Version = x509.Version;
        mKeyClass.x509_NotBefore = x509.NotBefore;
```

## private Key aus P12 Datei

```
mKeyClass.x509_NotAfter = x509.NotAfter;
mKeyClass.x509_Thumbprint = x509.Thumbprint;
mKeyClass.x509_SerialNumber = x509.SerialNumber;
mKeyClass.x509_PublicKey_Oid_FriendlyName =
x509.PublicKey.Oid.FriendlyName;
mKeyClass.x509_PublicKey_EncodedKeyValue_Format =
x509.PublicKey.EncodedKeyValue.Format(true);
mKeyClass.x509_RawData_Length = x509.RawData.Length;
mKeyClass.x509_s = x509.ToString(true);
mKeyClass.x509_PublicKey_Key_Xml =
x509.PublicKey.Key.ToXmlString(false);
mKeyClass.x509_HasPrivateKey = x509.HasPrivateKey;

if (x509.HasPrivateKey)
{
mKeyClass.x509_PrivateKey =
x509.PrivateKey.KeyExchangeAlgorithm;
mKeyClass.x509_Error = "";
}
else
{
mKeyClass.x509_Error = "No private key found. Please
convert with the tool";
}

X509Store store = new X509Store();
store.Open(OpenFlags.MaxAllowed);
store.Add(x509);
store.Close();
}
catch (Exception ExG)
{
mKeyClass.x509_Error = ExG.Message.ToString() + " (ExG)";
}
}
}
```

HIER SIGNIEREN:

```
public string Sign(string TextToSign, string PrivateKey)
{
byte[] valueToHash = null;
byte[] signedValue = null;
string ErrorText = "";

try
{
mSign = new Sign(TextToSign, PrivateKey);

RSACryptoServiceProvider rsaCryptoServiceProvider = new
RSACryptoServiceProvider();
```

private Key aus P12 Datei

## private Key aus P12 Datei

```
RSAPKCS1SignatureFormatter rsaFormatter = new
RSAPKCS1SignatureFormatter(rsaCryptoServiceProvider);
RSA RSA = RSA.Create();
ASCIIEncoding Encoding = new ASCIIEncoding();
SHA1Managed SHA1 = new SHA1Managed();
rsaFormatter.SetHashAlgorithm("SHA1");

rsaCryptoServiceProvider.ImportCspBlob(ConvertStringHexadecimalToByteArray(mSign.PrivateKey.ToString()));
rsaFormatter.SetKey(RSA);

valueToHash = Encoding.GetBytes(PrivateKey);
signedValue =
rsaFormatter.CreateSignature(SHA1.ComputeHash(valueToHash));
}
catch (Exception sEx)
{
    ErrorText = sEx.Message.ToString() + " (sEx)";
}
return Convert.ToBase64String(signedValue);
}
```