.NET-Beispielclient für Edec

Kurzdokumentation

Kurz und Knapp

- GUI mit WPF
- Eingaben werden als Property-Settings gespeichert
- Ein fertiger Soap-Envelope wird eingelsen und mit HttpWebRequest verschickt
- Der Benutzer kann die Antwort des Servers kopieren oder speichern

Zertifikate

In C# kann ein Zertifikat ganz einfach erstellt werden. Man verwendet dafür den Namespace `System.Security.Cryptography.X509Certificates`. Es gibt verschiedene Möglichkeiten. Praktisch ist es, wenn man den Pfad zu dem Zertifikat hat. Ein Zertifikat-Objekt kann dann folgendermassen instanziert werden:

```
var path = @"C:\path\to\certificate.pem";
var certificateA = new X509Certificate2(path);
```

Ist das Zertifikat zusätzlich mit einem Passwort geschütz, kann es ebenfalls im Konstruktor mitgegeben werden:

```
var password = "A really strong password";
var certificateB = new X509Certificate2(path, password);
```

X509Certificate oder X509Certificate2

Wenn man den oben erwähnten Namespace einbindet, fällt auf, dass mit `X509Certificate` und `X509Certificate2` zwei sehr ähnliche Klassen existieren. Um es kurz zu machen: `X509Certificate2` ist die neuere Version, welche grundsätzlich zu empfehlen ist. Erstens wird der Umgang mit neuern Zertifikaten ermöglicht. Zweitens ist auch die Entwicklung damit ein wenig komfortabler.

Beispiel

```
// verringert Lesbarkeit
new X509CertificateCollection(new[]{ new X509Certificate() });

// kein Array benötigt
new X509Certificate2Collection(new X509Certificate2());
```

HttpWebRequest

Um in .NET Http-Requests ausführen zu können, brauchen wir ein `HttpWebRequest`-Objekt.

```
var httpWebRequest = WebRequest.CreateHttp("http://example.com/service");
```

Jetzt können präzisieren, wie die Request aussehen soll:

```
httpWebRequest.Method = "POST";
httpWebRequest.ContentType = "text/xml;charset=UTF-8";
httpWebRequest.KeepAlive = true;
httpWebRequest.ClientCertificates = new
X509Certificate2Collection(certificate);
```

Es gibt noch weitere Properties. Um eine Edec-Soap-Request auszuführen werden jedoch nicht alle benötigt.

Wir wollen etwas senden. Dazu verwenden wir den RequestStream:

```
var body = File.ReadAllBytes(@"C:\path\to\soap\envelope.xml");

httpWebRequest.ContentLength = body.LongLength;

using (var stream = httpWebRequest.GetRequestStream())
{
    stream.Write(body, 0, body.Length);
}
```

Schliesslich können wir die Antwort lesen:

```
// hier ohne Exception-Handling
using (var responseStream = httpWebRequest.GetResponse().GetResponseStream())
{
    using(var reader = new StreamReader(responseStream))
    {
        Console.WriteLine(reader.ReadToEnd());
    }
}
```

Achtung

Die Methode `GetResponse()` kann eine `WebException` auslösen. Auf die Response kann man dann trotzdem zugreifen: `webException.Response`.