Titre : Présentation des Web Services Enhancements 2.0

Niveau : Moyen

Introduction

Le framework .Net fournit nativement de nombreuses classes permettant de développer des applications qui consomment et produisent des WebServices. Microsoft propose, avec les Web Service Enhancements, un add-on au framework .Net et à Visual Studio .Net. Il permet au développeur de construire des WebServices sécurisés basés sur les derniers standards et spécifications en vigueur. Les WSE 2.0 étendent les fonctionnalités de sécurité, de routage, et d'intégration de pièces jointes dans des messages SOAP de la version 1.0. Ils permettent aux développeurs de construire des applications basées sur les dernières spécifications publiées par Microsoft et différents acteurs de l'industrie (IBM, Verisign, SAP…) telles que WS-Security, WS-Policy, WS-SecurityPolicy, WS-Trust, WS-SecureConversation et WS-Addressing.

Dans cet article, nous abordons l'architecture WSE, nous présentons quelles sont les nouveautés de ce nouveau modèle objet et sa mise en œuvre. Nous étudierons le niveau de compatibilité assuré par rapport à la version 1.0; enfin nous terminerons par quelques exemples de mise en œuvre de WS-Security.

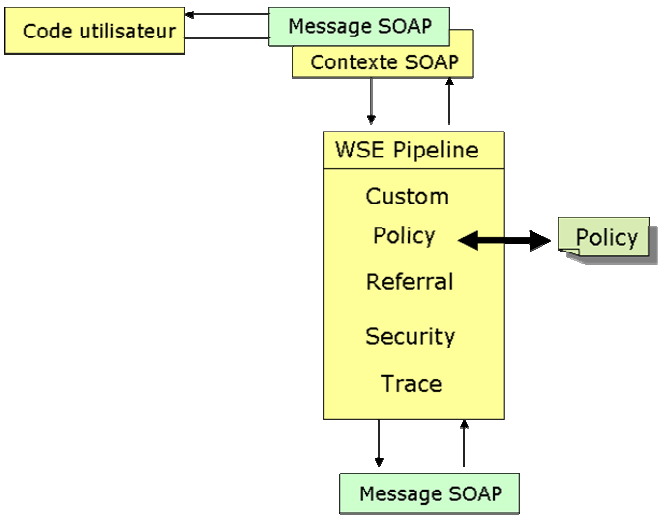
Commençons par étudier l'architecture des WebServices Enhancements.

L'architecture des Web Services Enhancements 2.0

SOAP propose un modèle standard de description de messages contenus dans un document XML. Un message SOAP est constitué d'une enveloppe contenant le corps d'un message et d'une entête pour le décrire. L'entête (<soap:Header>) est utilisée pour transmettre des informations sur la manière dont le message doit être traité : par exemple des informations de cryptage, d'authentification.

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <Digest>B839D234A3F87</Digest>
  </soap:Header>
  <soap:Body>
    <StockReport>
      <Symbol>MSFT</Symbol>
      <Price>74.56</Price>
    </StockReport>
  </soap:Body>
</soap:Envelope>
```

WSE est constitué d'un jeu de classes qui implémentent de nouveaux protocoles (WS-Security …) et d'un jeu de filtres hébergés par ASP .Net qui interceptent les messages SOAP entrants et sortants. Ces filtres interprètent ou génèrent les en-têtes permettant de prendre en charge les fonctionnalités requises. Le schéma ci-dessous illustre le pipeline WSE et présente les différents types de filtres.



Prenons le cas d'un fichier de configuration Policy défini. A l'exécution le runtime WSE vérifie que tous les messages entrants et sortants respectent les pré-requis définis dans le fichier Policy. Si ce n'est pas le cas alors une exception SOAP est levée (Soap Fault). Par défaut un certain nombre de paramètres sont prédéfinis : par exemple le corps du message doit être signé avec un certificat X.509.

Les nouveautés Web Services Enhancements 2.0

Les Web Services Enhancements 2.0 présentent de nombreuses nouveautés :

- Une intégration améliorée des outils de développement à la suite Visual Studio .Net.

- Le support de nouvelles spécifications Web Services :
    - **WS-Policy**
      WS-Policy permet de décrire pour un Web Services les pré-requis et paramétrages nécessaires pour un Web Service. Par exemple, définir les pré-requis en matière de sécurité tels que l'encryptage et la signature numérique basée sur un certificat X.509.
    - **WS-SecureConversation**
      WS-SecureConversation est basée sur les spécifications WS-Trust et WS-Security et permet de définir un contexte de sécurité partagé entre plusieurs applications. Il est ainsi possible que deux applications échangent de multiples messages sans systématiquement transmettre les credentials (informations de sécurité) dans chaque message ; au final, les performances devraient être améliorées.
    - **WS-Trust**
      WS-Trust décrit une structure destinée à gérer, à établir et à évaluer des relations de confiance permettant aux services Web d'interopérer d'une manière sûre. WS-Trust doit autoriser un service Web à communiquer indépendamment des protocoles de sécurisation employés.
    - **WS-Adressing**
      WS-Adressing permet de mettre en œuvre des protocoles de transport autres que http, comme TCP par exemple. Vous pouvez donc désormais développer votre propre application de peer-to-peer.
    - **WS-Attachment**
      WS-Attachment utilise la spécification DIME (Direct Internet Message Encapsulation) pour l'envoi et la réception de messages SOAP avec pièces jointes supplémentaires, telles que des fichiers binaires, des fragments XML et même d'autres messages SOAP.

- Le support de Security Token pour Kerberos
  WSE supporte l'utilisation de jetons de sécurité basés le mécanisme d'authentification Kerberos. Ainsi il est possible sur Windows XP et Windows 2003 d'utiliser des tickets Kerberos pour signer de façon numérique et encrypter des messages SOAP.

- La signature et le cryptage d'entête SOAP personnalisés

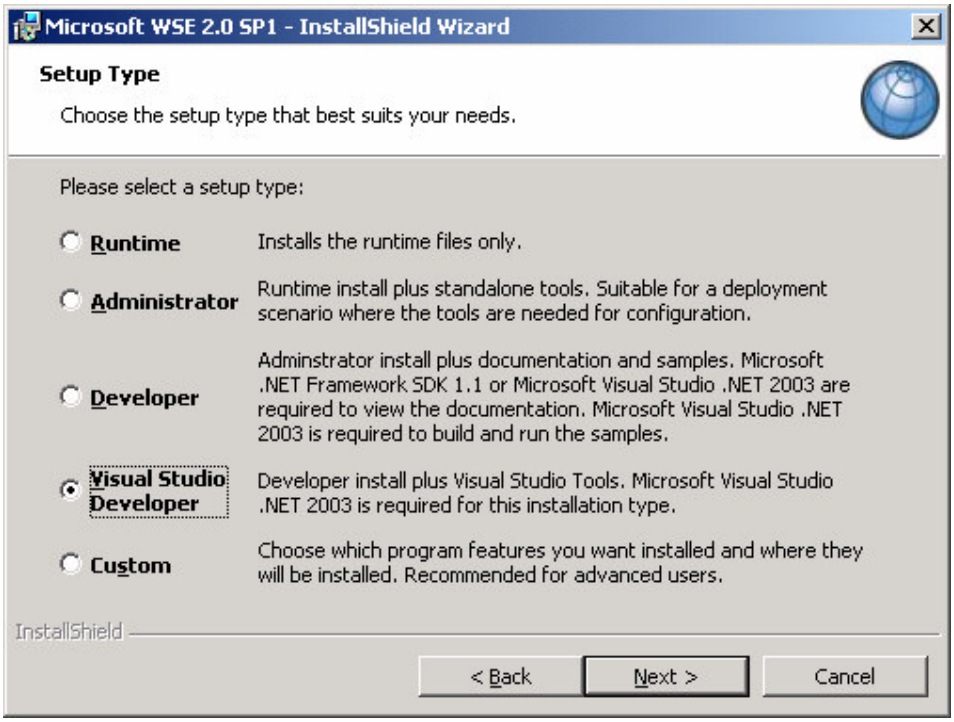
- La possibilité d'autoriser l'accès à des WebServices en se basant sur des rôles et des comptes Windows.

Installation de Web Services Enhancements 2.0

Le package d'installation de WSE 2.0 SP1 (Microsoft WSE 2.0 SP1.msi) est téléchargeable depuis le Microsoft Download Center à l'adresse suivante :

http://msdn.microsoft.com/webservices/building/wse/default.aspx

Le menu d'installation propose plusieurs types de mises à jour en fonction des profils utilisateurs. Les développeurs choisiront l'option « Visual Studio Developer » qui installe le runtime, les exemples, la documentation, les outils et intègre WSE 2.0 dans l'environnement de développement via des boîtes de paramétrage.



Plusieurs outils sont proposés avec WSE 2.0 :

- WseWsdl2.exe : Il permet de générer un proxy client à partir d'un fichier WSDL (Web Services Description Language)
- WseSettings 2.O Tool : Il permet d'activer et de paramétrer les fonctionnalités WSE à partir de l'environnement de développement Visual Studio .Net
- WseCertificate2.exe : Il permet de visualiser les certificats présents sur un ordinateur.

La compatibilité entre WSE 1.0 et WSE 2.0

Bien qu'un degré de compatibilité ascendante soit prévue entre les versions 1 et 2 de WSE, WSE2 .0 implémente des spécifications des Web Services mises à jour mais aussi une version standard OASIS de WS-Security. Les développeurs devront donc retravailler leur code afin d'intégrer le nouveau modèle objet WSE 2.0.

Il est possible d'exécuter des applications utilisant ces deux versions côte à côte sur une même machine, en effet les assemblys et namespaces sont différents. Cela permet de conserver un code existant développé avec les WSE 1.0 tout en intégrant de nouveaux services s'appuyant sur les fonctionnalités de WSE 2.0.

Remarque : Un service Web ne peut utiliser les deux SoapExtension (WSE 1.0 et WSE 2.0) en même temps.

Voici quelques unes des différences d'implémentations entre ces deux versions :

- Les assemblys et namespaces sont différentes : Microsoft.Web.Services pour WSE 1.0 et Microsoft.Web.Services2 pour WSE 2.0.
- Des messages SOAP transmis entre des applications implémentant WSE 1.0 et WSE 2.0 aboutiront à un message d'erreur SOAP (SOAP fault)
- La récupération d'un objet SoapContext se fait désormais via la propriété Current des classes RequestSoapContext et ResponseSoapContext et non plus en utilisant la classe HttpSoapContext.

```
using Microsoft.Web.Services2;   // Référence à la nouvelle assembly WSE 2.0
public class MaClasse
{
 [WebMethod]
  public string MaMethode()
  {
    SoapContext reqCtx = RequestSoapContext.Current;
    SoapContext resCtx = ResponseSoapContext.Current;
  }
  •••
}
```

- Suppression du namespace Microsoft.Web.Services.Security. IdecryptionKeyProvider au profit du namespace Microsoft.Web.Services.ISecurityTokenManager
- Il n'est plus nécessaire de définir une classe implémentant l'interface IpasswordProvider pour authentifier des utilisateurs Windows. Désormais WSE supporte l'authentification Windows d'un UserNameToken.
- La classe IdecryptionKeyProvider devient obsolète.

WS-Security

WS-Security permet à une application de sécuriser ses Web Services et :
- D'identifier un utilisateur qui demande l'exécution d'un service Web (authentification)
- De vérifier le rôle de l'utilisateur et les droits qui lui sont attribués (autorisation)
- De s'assurer qu'un message n'a pas été corrompu durant le transport (signature)
- De s'assurer que seul le destinataire d'un message peut le lire (encryptage)

WS-Security utilise des jetons de sécurité pour identifier un utilisateur. Les informations permettant la mise en œuvre de la sécurité sont stockées directement dans les entêtes SOAP.

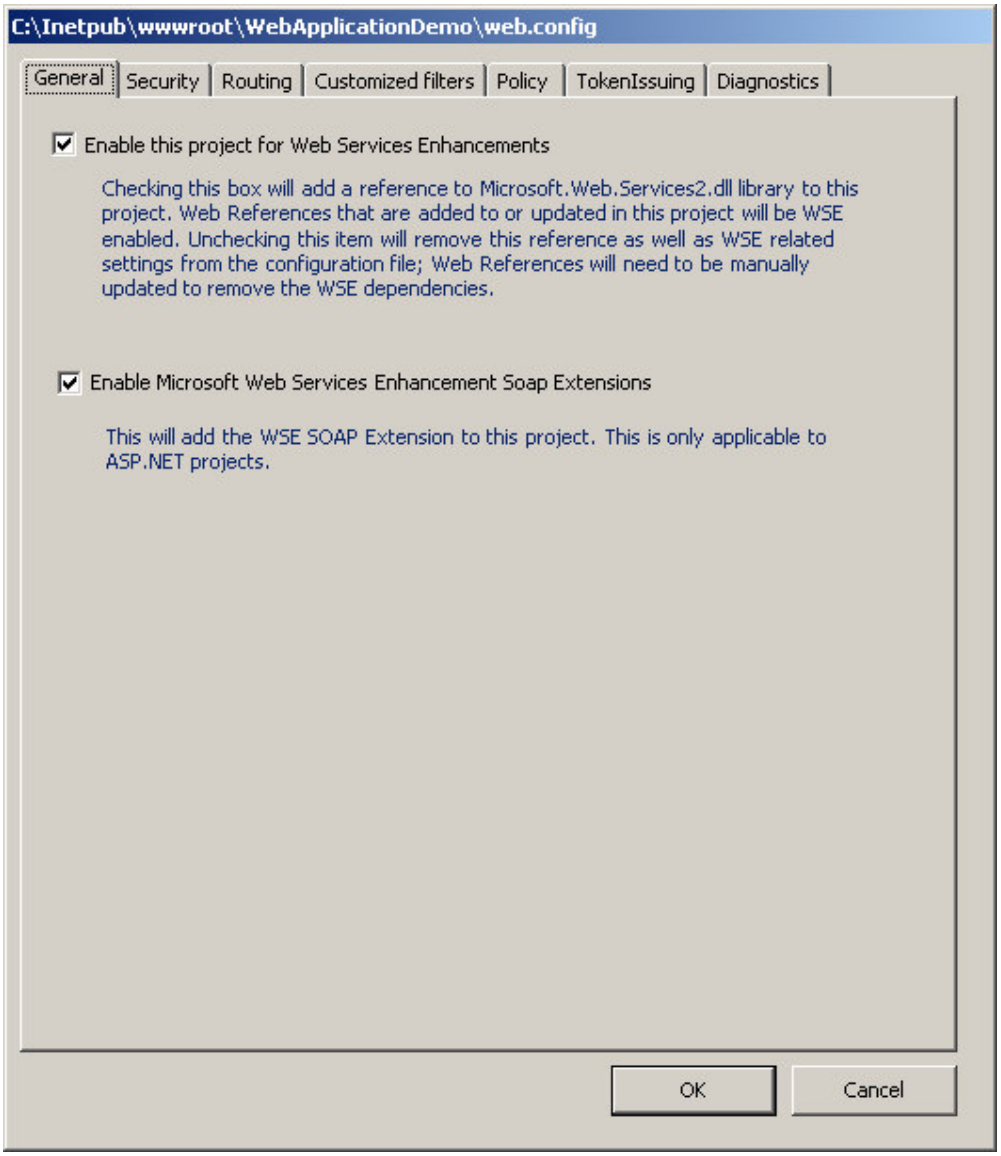
L'authentification est gérée soit à partir de jetons ***UserNameToken*** pour représenter un utilisateur/mot de passe soit à partir de jetons binaires (***BinarySecurityToken***) comme les tickets Kerberos ou les certificats X.509.

Sécurisation d'un service Web avec un UserNameToken

Regardons maintenant comment authentifier un utilisateur à partir d'une identité constituée d'un nom d'utilisateur et d'un mot de passe.

Tout d'abord, il est nécessaire de créer un WebService et une application cliente. Ces deux applications doivent être paramétrées afin de supporter les extensions WSE2.0. Il suffit de sélectionner l'option WSE2.0 Settings de chaque projet et de cocher dans l'onglet « General » l'activation des WSE pour le projet ainsi que l'ajout des extensions SOAP (option applicable uniquement pour les applications ASP .Net). Ces options sont directement rajoutées dans le fichier Web.Config.

```
<configSections>
   <section name="microsoft.web.services2"
type="Microsoft.Web.Services2.Configuration.WebServicesConfiguration, Microsoft.Web.Services2,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" />
 </configSections>
…
 <webServices>
    <soapExtensionTypes>
      <add type="Microsoft.Web.Services2.WebServicesExtension, Microsoft.Web.Services2, Version=2.0.0.0,
Culture=neutral, PublicKeyToken=31bf3856ad364e35" priority="1" group="0" />
    </soapExtensionTypes>
  </webServices>
```



Pour utiliser les fonctionnalités WSE coté client, il est nécessaire de référencer les assemblys correspondant à WSE. Nous instancions ensuite notre classe proxy qui porte le nom de « Service1Wse » au lieu de « Service1 ». Cette nouvelle classe hérite de la classe Microsoft.Web.Services2.WebServicesClientProtocol qui supporte les jetons de sécurité.

Nous créons ensuite un jeton de sécurité en lui précisant le nom et le mot de passe de l'identité à transmettre au WebService. Enfin, nous rajoutons le jeton de sécurité à la collection des jetons de sécurité du contexte soap. Il sera lu par le service web afin de déterminer l'identité de l'application cliente.

```
using Microsoft.Web.Services2.Security;
using Microsoft.Web.Services2.Security.Tokens;

…

ServiceDemo.Service1Wse ws= new ServiceDemo.Service1Wse();

UsernameToken Token = null;

Token = new UsernameToken(@"peyrusse","motdepasse",PasswordOption.SendPlainText);
ws.RequestSoapContext.Security.Tokens.Add(Token);
ws.WsTest();
```

Nous constatons que l'entête SOAP transmise à été modifiée et intègre un UserNameToken (le nom d'utilisateur, le mot de passe) et la date de création. Le mot de passe apparaît en clair, et peut être caché en utilisant le paramètre PasswordOption.SendHashed pour l'encryter.

```
<wsse:Security soap:mustUnderstand="1">

    <wsse:UsernameToken xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd" wsu:Id="SecurityToken-c726d91d-61fe-4e90-b834-358c078fd3ea">
```

```
        <wsse:Username>peyrusse</wsse:Username>
        <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#PasswordText">motdepasse</wsse:Password>
        <wsse:Nonce>8u3w/SHU1o4LAdgjh3d1KQ==</wsse:Nonce>
        <wsu:Created>2004-09-08T14:52:11Z</wsu:Created>
      </wsse:UsernameToken>
    </wsse:Security>
```

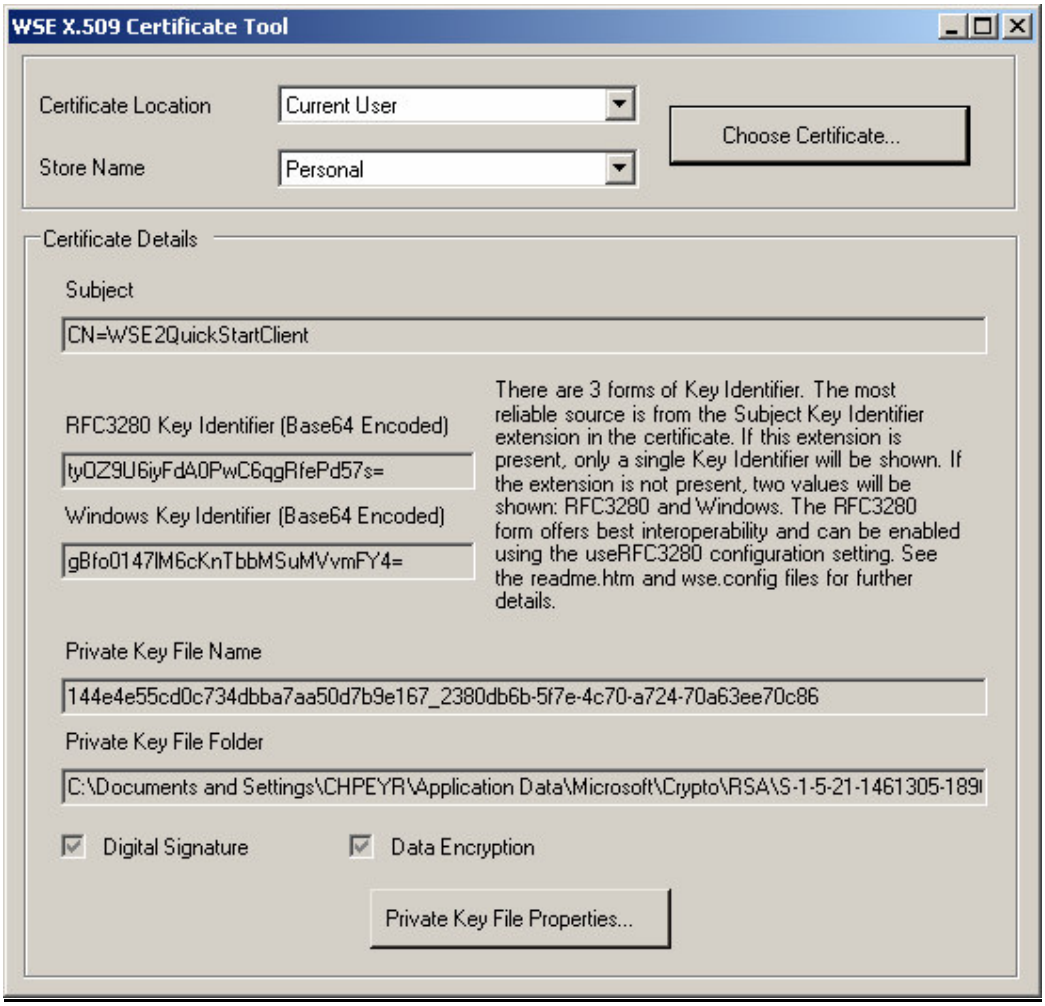
Pour vérifier l'identité de l'utilisateur à l'origine de la demande, le service web récupère le contexte Soap courant. Il parcours la liste des jetons de sécurité afin de rechercher ceux de type UserNameToken et vérifie que le nom d'utilisateur transmis est bien celui du jeton. Si l'utilisateur est bien authentifié à partir de comptes Windows, alors WSE renseigne la propriété Principal de l'objet UserNameToken avec son nom. Nous utilisons cette propriété pour vérifier avec la méthode IsInRole si l'utilisateur appartient au groupe « Admin » de la machine locale.

```
SoapContext ctx  = RequestSoapContext.Current;

foreach (SecurityToken tok in ctx.Security.Tokens)
{
        if (tok is UsernameToken)
        {
                UsernameToken user = (UsernameToken)tok;

                if (user.Username == @"peyrusse")
                {
                  if (user.Principal.IsInRole(System.Net.Dns.GetHostName() + @"\Admin"))
                  return "rôle administrateur";
                }
        }
        if (tok is X509SecurityToken)
        {
                …
        }

}
```

Sécurisation d'un service Web avec un BinarySecurityToken

Le mécanisme mis en place précédemment ne permet pas de s'assurer de la non corruption du message durant le transport. Il est alors nécessaire de signer notre message SOAP avec une signature numérique telle qu'un certificat X.509.

Vous pouvez utiliser l'outil fourni avec WSE, permettant de visualiser les certificats disponibles sur un machine (wsecertificate2.Exe). Pour notre test, nous utiliserons un certificat de test installé par WSE. Un certificat est identifié par une clé codée en base 64, dans notre cas la clé est "gBfo0147lM6cKnTbbMSuMVvmFY4=".



Nous définissons une méthode qui retourne un jeton de type X509SecurityToken. Cette méthode récupère un objet X509CertificateStore qui contient un ensemble de certificats puis recherche le certificat identifié par le paramètre KeyId. Enfin il retourne un jeton constitué à partir de ce certificat.

```
private X509SecurityToken GetX509Token(string keyId, string storeId)
{
        X509CertificateStore store = X509CertificateStore.  LocalMachineStore (storeId);
        store.OpenRead();
        X509CertificateCollection certs =
        store.FindCertificateByKeyIdentifier(Convert.FromBase64String(keyId));
        store.Close();
        return new X509SecurityToken(((X509Certificate)certs[0]));
}
```

Nous référençons l'assembly permettant de manipuler des certificats X509.

```
using Microsoft.Web.Services2.Security.X509;
```

Nous utilisons la classe MessageSignature afin de générer une signature numérique à partir de notre jeton de sécurité X509, puis nous nous en servons pour signer notre message SOAP.

```
X509SecurityToken X509Token = GetX509Token("gBfo0147lM6cKnTbbMSuMVvmFY4=",
X509CertificateStore.MyStore);

ws.RequestSoapContext.Security.Tokens.Add(X509Token);
ws.RequestSoapContext.Security.Elements.Add(new MessageSignature(X509Token));
```

Nous constatons, en exécutant le code qu'un élément BinarySecurityToken a été ajouté dans l'entête SOAP de notre message. L'attribut « ValueType » indique que le jeton est de type X509v3, l'attribut « EncodingType » précise la méthode d'encodage, en l'occurrence « Base64Binary ».

```
<wsse:BinarySecurityToken ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3" EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" wsu:Id="SecurityToken-c822f48c-e95f-4066-86d9-5d38dbb2cea6">MIIBxDCCAW6gAwIBAgIQYpjr4FOk3IFNSd3lJj6ItzANBgkqhkiG9w0BAQQFADAWMRQwEgYDVQQDEwtSb290IEFnZW5jeTAeFw0wMzA3MDgxODQ4MTBaFw0zOTEyMzEyMzU5NTlaMB8xHTAbBgNVBAMTFFdTRTJRdWlja1N0YXJ0U2VydmVyMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDLkqglArInukRlDwXbcjN3zxfsjeaLd+IvfyD5o35pUjpTkPwPXmApScr8UVQxB5JDRSVlMz1lUQ6CBLFLGIAQOpbPKn2oul3VmKAf9nRQf9PLU+biWozZXkhebIya43D75r5+5NUq1RbQiCC4qIobRqUdg6adujBY333wJy4YgwIDAQABo0swSTBHBgNVHQEEQDA+gBAS5AktBh0dTwCNYSHcFmRjoRgwFjEUMBIGA1UEAxMLUm9vdCBBZ2VuY3mCEAY3bACqAGSKEc+41KpcNfQwDQYJKoZIhvcNAQEEBQADQQAGSGNKz1gZqbXN8JYl0PQM7ngkHfW1mQ88NRYADmoHw5A/rUZDHAPs5HLSn3i5iXlRwT91v3SU6iuaAid+Mwyq
</wsse:BinarySecurityToken>
```

Il existe d'autres moyens de sécuriser notre service Web: par exemple, crypter le contenu du message afin que seul son destinataire puisse le lire ; nous ne traiterons pas cette partie dans cet article.

Conclusion

Les Web Services Enhancements 2.0 implémentent de nouvelles spécifications dans des domaines cruciaux pour les services Web : la sécurité, la fiabilité de la messagerie et l'envoi de pièces jointes. Ils sont construits sur les standards existants tels que XML, SOAP et WSDL et évolueront afin de rester à jour avec les dernières avancées des Services Web. A priori, une version majeure WSE 3.0 est prévue avant la sortie d'Indigo.