Standard autour des architectures basées sur le Protocole SOAP

I. INTRODUCTION :

1) Définition:

SOAP (***Simple Object Access Protocol***) est un protocole standardisé par W3C, et qui assure le transfert des messages et les appels de procédures RPC indépendamment du transport.

Il définit le cadre général pour l'échange de données structurées en XML.

Il permet d'échanger des structures de données complexes en XML avec les espaces de nom, et la spécification XML Schéma.

2) Bref historique:

- **Septembre 1999 : SOAP 0.9**
  - Spécifications par MicroSoft et DevelopMentor.
- **Décembre 1999 : SOAP 1.0**
  - Soumission des spécifications à l' IETF
  - Association de UserLand
- **Mai 2000 : SOAP 1.1**
  - Nombreuses associations : IBM, HP, Lotus, Compaq, Intel …
  - XIDL : rapprochement de Corba
- **Septembre 2000**
  - Groupe de travail W3C pour la standardisation de SOAP
  - Corba/Soap Interworking RFP => SCOAP
- **Avril 2007 : SOAP 1.2**
  - SOAP abandonne le modèle objet pour utiliser XML Infosets.

II. PRINCIPES FONDAMENTAUX DU PROTOCOLE SOAP :

Le principal objectif du protocole SOAP est de permettre la normalisation des échanges de données. Il définit un moyen uniforme d'échange de données encodées XML sous forme d'appels de procédures à distance RPC, en utilisant souvent le HTTP comme protocole de communication ainsi les protocoles SMTP, POP, et FTP dans certains cas.

Il assure l'interopérabilité entre composants tout en restant indépendant des plates-formes et des langages de programmation. Il repose sur deux standards :

- XML : pour la structuration des messages.
- http : pour le transport.

Il permet une interopérabilité avec divers environnements logiciels (.NET, Java/RMI, CORBA, COM/DCOM …) quel que soit leur plate-forme d'exécution.

Les paquets de données circulent sous format XML, mais tous types de données peuvent être transportés même si les données binaires peuvent faire l'objet d'un encodage spécifique.

Les données ayant un format binaire sont transportées sous forme de pièces jointes avec le message SOAP principal.

Les messages SOAP peuvent traverser les Proxy et les pare-feu.

Il permet l'échange de données que ce soit en mode synchrone (requête/réponse) ou asynchrone (accompagnement des processus) [1] [2].

1. Framework de messagerie :

La communication dans le protocole SOAP est similaire à celle entre services web, cependant celui-ci fait introduire des nouveaux concepts relatifs spécifiquement à la manière avec laquelle les messages sont manipulés (avec une implémentation technique de ces concepts dans le ***Framework***).

a. Nœud SOAP (Nodes):

Un nœud SOAP est l'infrastructure technique qui alimente les différents scénarios de communication entre services, ainsi un nœud SOAP peut être vu comme étant le processeur logique permettant de recevoir, transmettre, et exécuter un ensemble d'opérations sur les messages SOAP. Chaque nœud est identifié par un URI.

L'implémentation d'un nœud SOAP est fortement liée à la plateforme utilisée, tandis que leurs interfaces de communication doivent être standards pour leur permettre de communiquer dans différents environnements. Ainsi un nœud SOAP peut être classifié en SOAP server, SOAP listener ou bien SOAP router selon son rôle dans la communication.

Un nœud SOAP peut implémenter des fonctions spécifiques à lui-même sans qu'elles soient présentes sur d'autres nœuds.

b. Différents rôles joués par un nœud:

Selon le rôle d'un nœud SOAP dans une communication, celui-ci peut être vu comme:

- Expéditeur initial (Demandeur de service)(***Initial Sender***) ou (***Service Requestor***).
- Récepteur Final (Fournisseur de service) (***Ultimate Receiver***) ou (***Service Provider***).
- Nœud intermédiaire (***Intermediarry***)

2. Encodage et sérialisation standard pour les objets:

Le protocole SOAP impose une spécification d'encodage/sérialisation standard, qui doit être appliquée sur les données du message, que ce soit du côté client ou bien fournisseur. Cette tâche est assurée par le runtime SOAP implanté des deux côtés (client, fournisseur), et essentiellement utilisée avec le RPC [1].

Exemple: encodage d'un tableau d'entiers :

```
<SOAP-ENC:Array SOAP-ENC:ArrayType="xsd:int[3]">
         < SOAP-ENC:int> 8 </ SOAP-ENC:int>
         <SOAP-ENC:int> 8 </ SOAP-ENC:int>
         <SOAP-ENC:int> 8 </ SOAP-ENC:int>
</SOAP-ENC:Array>
```

3. Invocation de Service d'objets distants via SOAP RPC :

Le protocole SOAP permet d'invoquer un objet distant en communiquant les informations nécessaires à l'appel: nom de la méthode, et sa signature numérique (ses arguments) dans un message au format XML. La réponse à la requête est aussi renvoyée encapsulée dans un message au format XML. La librairie SOAP assure l'envoi du RPC à l'aide du protocole du transport choisi.

Les différentes étapes d'invocation des objets distants sont [1] [3] :

- Le programme client SOAP crée le document XML contenant les informations nécessaires à l'invocation d'un objet distant, et le transmet au protocole HTTP (requête POST).
- Le message est transmis au serveur via une connexion HTTP.
- Le serveur reçoit le message et fait appel à l'objet concerné.
- L'objet effectue le traitement demandé et renvoie les résultats au serveur SOAP.
- Le serveur SOAP encapsule les résultats et les envoie dans un document XML au client via le protocole HTTP.
- Le client SOAP décode le message et envoie les résultats au demandeur initial.

Exemple: appel du service GetLastTradePrice:

```
Post /StockQuote HTTP/1.1
Host : www.stockquoteserver.com
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAPAction: "http://example.org/2001/06/quotes"
<env:Envelope xmlns:env="http://www.w3.org/2001/06/soap-envelope">
     <env:Body>
          <m: GetLastTradePrice
          Env:encodingStyle=="http://www.w3.org/2001/06/soap-
          encoding"
          Xmlns:m="http://example.org/2001/06/quotes">
               <symbol>DIS</symbol>
          </m: GetLastTradePrice>
     </env:Body>
</env:Envelope
```

4. Messages SOAP :

Le framework de messagerie SOAP requiert que les messages SOAP soient composés d'une enveloppe (***Envelope***) contenant un entête (***Header***) et un corps (***Body***).

Exemple : (Squelette d'un message SOAP)

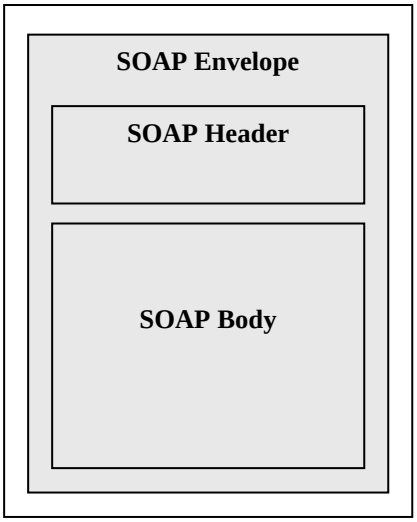
```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    ...
  </env:Header>
  <env:Body>
    ...
  </env:Body>
</env:Envelope>
```

La partie entête du message qui sert à véhiculer les méta-données du message est une partie optionnelle. Elle est formée d'un ensemble de blocs appelés "entrées entête", dont chacun peut contenir plusieurs attributs.

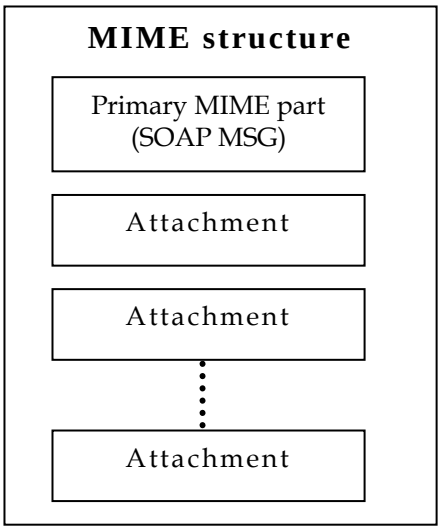
La partie Body est la partie principale du message, elle contient les données applicatives destinées au récepteur final.

Cependant un message SOAP peut avoir des fichiers attachés au fichier principal, ainsi tout fichier qui n'est pas au format xml doit apparaître dans la partie attachement. Ainsi tous les fichiers attachés en pièces jointes doivent être référencés par des URL dans le message principal.

La spécification SOAP 1.1 avec attachements a été soumise au W3C comme base du XMLP : http://www.w3.org/TR/SOAP-attachments. Elle utilise le MIME "Multipart/Related" comme container pour l'enveloppe SOAP et tous les attachements complémentaires. [1]



Message SOAP simple



Message SOAP avec pièces jointes

5. Acheminement des messages :

A l'arrivée d'un message chez un nœud SOAP, celui-ci doit [2][3][4]:

1. Vérifier la structure du message: enveloppe, entête, et corps du message.
2. Déterminer les entrées d'entêtes destinées à lui-même, et l'ensemble des rôles à en exécuter.
3. Sélectionner parmi ces entrées celles qui sont mandataires.
4. Si une de ces entrées n'est pas comprise ou bien qu'il n'a pas la capacité d'en traiter, il envoie un message d'erreur à l'expéditeur initial.
5. Traiter toutes les entrées mandataires, ainsi que celles qui sont optionnelles et dont il a la capacité d'en traiter.
6. Traiter le corps du message s'il s'agit du récepteur final.
7. Retransmettre le message au nœud suivant s'il s'agit d'un nœud intermédiaire.

a) Traitement de l'entête:

Une entrée entête SOAP peut contenir un attribut d'information appelé "***role***", qui sert à déterminer quel nœud doit exécuter ce rôle. Si la valeur "none" est affectée à cet attribut, aucun

nœud ne doit traiter cette entrée. Cette situation est généralement utilisée pour transmettre des informations requises pour le traitement d'autres entrées [2].

b) Compréhension des entrées d'entête:

L'utilisation de l'attribut ***mustUnderstand = "true"*** permet de rendre la compréhension et le traitement d'une entrée par un nœud SOAP spécifique obligatoire. Ainsi si le nœud concerné n'a pas compris ou bien n'est pas capable de traiter cette entrée, l'acheminement du message est arrêté et un message d'erreur est envoyé à l'expéditeur initial.

Ceci est très utile pour éviter une mauvaise compréhension ou un mauvais traitement des messages. Par exemple on peut imaginer un module qui crypte et retire le corps et insère un en-tête contenant une somme de contrôle et une indication du mécanisme de cryptage utilisé. La spécification d'un tel module indiquerait que l'algorithme de décryptage du côté récepteur doit être appliqué avant tout autre module reposant sur le contenu du corps.

c) Réacheminement des messages par les nœuds intermédiaires:

Un nœud SOAP intermédiaire doit obligatoirement retransmettre le message reçu au nœud suivant. Cependant il doit:

- Supprimer toutes les entrées entête traitées, ainsi que celles ignorées et dont elles ne contiennent pas l'attribut "***relay = true*** ".
- Garder toutes les entrées non traitées qui contiennent l'attribut "***relay = true***", ainsi que toute entrée destinée aux autres nœuds.
- Ne prendre un attribut ***relay*** en considération que dans le cas où ***mustUnderstand*** est absent ou bien égal à ***"false"***.

La retransmission des blocs entête doit obéir aussi aux spécifications des caractéristiques SOAP en cours d'utilisation. La spécification de chacune de ces caractéristiques doit décrire la sémantique imposée, en incluant les règles décrivant la construction du message réacheminé. Ces règles pourraient décrire où placer des blocs d'en-tête SOAP insérés ou réinsérés. Les blocs d'en-tête SOAP insérés peuvent être indistincts d'un ou plusieurs blocs d'en-tête retirés [3].

6. Nœud intermédiaire actif :

En plus du traitement effectué par des intermédiaires de réacheminement, les intermédiaires actifs entreprennent du traitement supplémentaire pouvant modifier le message sortant selon des manières non décrites dans le message entrant. C'est-à-dire qu'ils peuvent entreprendre un traitement non décrit par un bloc d'en-tête SOAP du message arrivé. L'ensemble potentiel de services fournis par un intermédiaire actif inclut (liste non exhaustive) : des services de sécurité, des services d'annotation et des services de manipulation de contenu.

Il se peut que les résultats de tels traitements actifs aient un impact sur l'interprétation des messages par les nœuds suivants. Par exemple, lors de la génération d'un message sortant, un intermédiaire actif peut avoir retiré et chiffré des ou tous les blocs d'en-tête SOAP trouvés dans le message entrant.

Il est fortement recommandé de décrire les caractéristiques fournies par les intermédiaires actifs de manière à permettre la détection de telles modifications par les nœuds SOAP affectés dans le chemin du message [3].

7. Règles d'encodage:

Les règles d'encodage SOAP expriment les types de données définis pour l'application en XML. Elles constituent un ensemble unique de règles d'encodage défini par la spécification SOAP. Cet encodage est fondé sur les schémas XML W3C. Il existe deux types de données [1] :

a) Les types de données simples:

Soap utilise tous les types simples définis comme ***Built-in data-type*** dans la partie ***XML Schema Part2***. L'espace de noms associé aux types de données est le schéma XML "*http://www.w3.org/1999/XMLSchema*". Il fournit les types courants comme: ***string, integer, float, date***, etc [4].

1. **Chaînes de caractères:**

   Ce type est différent du type string utilisé dans la plus part des langages de programmation, dans certain cas il peut interdire l'utilisation de certains caractères. Une chaîne peut être codée comme valeur à référence unique ou à références multiples.

2. **Enumérations:**

   Une liste de valeurs distinctes appartenant au même type de base. L'exemple suivant définit un ensemble de régions géographiques:

```
<simpleType name= "Region" base= "xsd:string">
    <enumeration value="Nord"/>
    <enumeration value="Sud"/>
    <enumeration value="Est"/>
    <enumeration value="Ouest"/>
</simpleType>
```

3. **Tableaux d'octets:**

   Codé comme valeur à référence unique ou à référence multiples. Les règles concernant les tableaux sont similaires à celles concernant les chaînes de caractères.

b) Les types composés :

A l'instar de certains langages de programmation, l'encodage SOAP définit les types composés suivants [4] :

- ***Struct*** (enregistrement) qui désigne une valeur composée dans laquelle lr nom d'accesseur seul permet de différencier les valeurs de membres, chaque accesseur ayant un nom unique.
- ***Array*** qui désigne un tableau dont les éléments sont de même type.

1. **Struct:**

La structure Livre	Le schéma correspondant

`<e:Livre>`
`<auteur>Hubert Kadima</auteur>`
`<preface> Texte preface </preface>`
`<intro>ce livre est pratique </intro>`
`</e:livre>`
	`<element name="Livre">`
	`<complex Type>`
	`<element name="auteur" type="xsd.string"/>`
	`<element name="auteur" type="xsd.string"/>`
	`<element name="auteur" type="xsd.string"/>`
	`</complex Type>`
	`</element>`

2. **Array:**

Ils ont un type particulier marqué par leur attribut ***xsi:type*** qui est ***SOAP-ENC:Array***. Les éléments qui possèdent l'attribut ***xsi:type*** sont déclarés en tant que tableau d'encodage SOAP. Le

type des membres du tableau est déclaré au moyen d'un autre attribut ***SOAP-ENC: arrayType***. Cet attribut indique le type de la taille du tableau.

```
<nombres                                    xsi:type=SOAP-ENC:Array"SOAP-
ENC:arrayType="xsd:integer[5]">
     <item>10</item>
     <item>20</item>
     <item>30</item>
     <item>40</item>
     <item>50</item>
</nombres>
```

8. Liaison du SOAP avec les protocoles du transport:

a) Modes de transport de message SOAP:

Les messages sont véhiculés soit en mode RPC ou bien en mode messagerie.

- Le mécanisme RPC SOAP est utilisé pour les communications au sein de l'entreprise, en partant de l'hypothèse que le couplage fort entre composants du système n'est pas une contrainte.
- La communication par messages est utilisée entre deux entités métiers différentes.

XP (XML Protocol) est une nouvelle initiative du W3C pour tenter de combler les faiblesses du SOAP et ajouter des nouvelles fonctionnalités.

b) Utilisation de SOAP avec RPC:

Le protocole SOAP encapsule et permet d'échanger les appels RPC en utilisant la flexibilité du XML. Pour effectuer une invocation de méthode distante les informations suivantes sont nécessaires:

- URI de l'objet cible.
- Nom de la méthode.
- Signature de la méthode (facultative)
- Paramètres de la méthode
- Données de l'entête (facultatives)

Les appels et réponses de méthodes RPC sont transmis encapsulés dans l'élément Body du message.

Des informations complémentaires liées au codage d'une requête de méthode mais ne faisant pas partie de la signature formelle de la méthode peuvent être exprimés dans le codage RPC. Dans ce cas elles doivent être exprimées comme sous-éléments de l'élément Header du message [3][4].

c) Utilisation du SOAP avec http en mode messagerie:

Une caractéristique SOAP est une extension de la structure SOAP pour les messages. Bien que SOAP ne pose aucune contrainte sur la portée de telles caractéristiques, on trouve parmi les exemples "fiabilité", "sécurité", "corrélation", "routage" et des séquences d'échange de messages (Message Exchange Patterns, MEPs) telles que interactions requêtes/réponse, messages unidirectionnels et conversation d'égal à égal

d) Liaison sur des protocoles applicatifs spécifiques:

Des protocoles sous-jacents pourraient être conçus pour un but particulier ou un profil d'application. Les liaisons de SOAP sur de tels protocoles pourraient utiliser la même

identification de points d'entrée (par exemple numéro de port TCP) que le protocole sous-jacent, de manière à réutiliser l'infrastructure existante associée à ce protocole.

Cependant, l'utilisation de ports connus par SOAP peut engager des manipulations supplémentaires non intentionnelles par des intermédiaires et implémentations sous-jacentes. Par exemple, HTTP est communément compris comme un protocole de "navigation sur le Web" et les administrateurs réseau pourraient y mettre certaines restrictions ou interposer des services comme du filtrage, de la modification du contenu, du routage, etc. Souvent, ces services sont interposés en utilisant le numéro de port comme heuristique.

Par conséquent, les définitions de liaison sur des protocoles sous-jacents avec un port par défaut bien connu ou des profils d'application devraient documenter les interactions potentielles avec des infrastructures couramment déployées sur ces ports ou en conformité avec des profils d'application par défaut. Les définitions de liaison devraient également illustrer l'utilisation de liaison sur un autre port que celui par défaut comme moyen d'éviter des interactions non voulues avec de tels services.

9. Sécurité du SOAP:

La structure de messages SOAP ne fournit pas directement de mécanismes pour traiter le contrôle d'accès, la confidentialité, l'intégrité et la non répudiation. De tels mécanismes peuvent être fournis sous forme d'extensions SOAP grâce au modèle d'extensibilité.

Le concepteur et l'implémenteur du protocole SOAP doivent prendre en considération différents types de risques qui peuvent être produits [3][5]:

a) Risque dû au contenu du message (nœud SOAP) :

SOAP peut transporter des données définies par l'application dans le bloc d'en-tête ou bien dans le contenu du corps SOAP. Le traitement de ces données peut éventuellement inclure de traiter des effets de bord comme les changements d'état, la journalisation des informations ou la génération de messages supplémentaires.

Les considérations de sécurité ne se limitent pas uniquement à la partie XML du message, mais aussi les fichiers attachés, et les données passées comme paramètres (par exemple des chaînes de requêtes sous forme d'URI).

Ainsi il est fortement recommandé, pour tout scénario de déploiement, que:

- Seuls des blocs d'en-tête soigneusement spécifiés (qui peuvent entraîner des effets de bord bien compris), soient traités par un nœud SOAP.
- Aucune entrée dans le corps du message qui peut avoir des conséquences graves sur le nœud récepteur n'est traitée par un nœud SOAP.
- Toute partie attachée au message principal, ou donnée passée comme paramètre doit être analysée avant d'être traitée.

b) Risque dû au transport (nœuds intermédiaires):

SOAP fournit intrinsèquement un modèle de traitement qui peut impliquer un passage de message SOAP par de multiples nœuds SOAP. Les intermédiaires SOAP sont par définition des tiers et représentent une opportunité pour des attaques de tiers.

Des failles de sécurité sur des systèmes exécutant des intermédiaires SOAP peuvent aboutir à de sérieux problèmes de sécurité et de confidentialité. Un intermédiaire SOAP atteint ou un intermédiaire implémenté ou configuré sans attention aux considérations de sécurité et de confidentialité pourrait être utilisé pour un large étendu d'attaques potentielles.

Des risques peuvent être aussi survenus dus à la portée des mécanismes de sécurité fournis par le protocole sous-jacent qui peut ne pas couvrir le chemin complet du message.

Il est important aussi de noter qu'il n'est pas obligatoire dans SOAP que tous les bonds entre nœuds participants utilisent le même protocole sous-jacent, et même si c'était le cas, l'utilisation du même intermédiaire SOAP est susceptible de dépasser la portée de la sécurité du niveau transport.

c) Risque dû aux spécifications (protocole de transport):

Les auteurs de spécification de liaison doivent décrire en détails les implications de sécurité dans le cas où des recommandations ou obligations ne seraient pas suivies, car la plupart des implémenteurs n'auront pas le bénéfice de l'expérience et des discussions qui auront produit la spécification.

Une spécification de liaison peut ne pas se préoccuper de fournir de contre-mesures pour tous les aspects des risques de sécurité inhérents. Les auteurs de spécification de liaison doivent identifier tout risque qui pourrait subsister et indiquer où des contre-mesures supplémentaires seraient nécessaires au-delà de celles fournies dans la spécification de la liaison.

Les auteurs de spécifications de liaisons doivent être conscients que les modules d'extension de SOAP exprimés sous forme de blocs d'en-tête SOAP pourraient affecter le protocole sous-jacent de manière imprévue. Un message SOAP transporté sur une liaison sur un protocole particulière pourrait aboutir à des caractéristiques en apparence contradictoires. Un exemple de ceci serait un message SOAP transporté sur HTTP, utilisant le mécanisme d'authentification basique de HTTP en combinaison avec un mécanisme d'authentification basique de SOAP. Il est fortement recommandé qu'une liaison de spécification décrive toute interaction de ce genre entre des extensions et les protocoles sous-jacents.

III. IMPLEMENTATION SOAP:

Une multitude d'implémentations du protocole SOAP et Toolkits existent aujourd'hui, qui correspondent aux différents environnements et langages de programmation (Java, C++, C, Perl, PHP, Python..) et qui peuvent répondre aux différents besoins de l'utilisateur.

Ces outils peuvent implémenter de différentes fonctionnalités, au-delà des fonctionnalités de base communes:

- Processus fondamental de création, déploiement et utilisation des services web.
- Comprendre et réaliser tous les traitements concernant les styles d'encodage et la translation des types natifs de données dans XML et vice versa.
- Encodage, décodage et routage des messages SOAP à travers le protocole de transport.

Cependant les fonctionnalités spécifiques sont diverses, elles concernent surtout:

- L'utilisation du protocole de transport: certains outils implémentent leurs propres serveurs HTTP, d'autres s'attendent à être intégrés comme partie d'un serveur web particulier. Certains d'autres sont pluggable permettant de choisir entre protocoles (SOAP:Lite choix entre FTP, HTTP, SMTP, POP3, TCP, Jabber,..).
- Les fonctionnalités de sécurité sont généralement implémentées à grandeurs différentes d'un outil à un autre.
- Appel d'objet à distance avec différents niveaux d'automatisation.
- Utilisation et manipulation des données SQL.

IV. STRUCTURE DES MESSAGES SOAP :

La structure d'un message SOAP diffère selon les types d'informations transportées :

Si ces informations sont tous textuelles, le format de message simple suffit, par contre s'il y a des informations nécessitant des formats de données binaires spécifique (image, vidéo, son, pdf,..) le format simple en suffit plus, et l'ensemble de ces informations doivent être encapsulées dans un format MIME est indispensable.

1. Message simple :

Un message SOAP simple est message contenant uniquement du texte (XML). Il est composé de :

a) L'enveloppe SOAP (*Envelope*) :

C'est l'élément récipient (conteneur) qui englobe tous les autres éléments du message. Sa déclaration est strictement obligatoire puisque c'est elle qui détermine le début et la fin de la partie SOAP dans une requête [3][4][5].

Le mot-clé ***Envelope*** (qui doit suivre le nom local de l'enveloppe) est suivi d'un espace de nom contenant un URI qui pointe vers un schéma XML. La valeur de cet URI (schéma XML) varie selon la version du message SOAP (voir l'exemple 1).

A la suite de cet URI l'attribut ***encodingStyle*** peut être utilisé pour indiquer au destinataire du message SOAP quel format de sérialisation a été utilisé pour coder un élément donné et ses descendants.

Cet attribut peut apparaître sur n'importe quel élément. Les éléments descendants peuvent remplacer la valeur d'un attribut ***encodingStyle*** spécifié sur leur ancêtre.

Remarque: un message SOAP peut commencer par une déclaration XML (voir exemple 2).

1. Valeur des URI selon les versions :

Version SOAP		Valeur de l'URI utilisé
1.1	**Enveloppe**	http://schemas.xmlsoap.org/soap/envelope/
	Encodage	http://schemas.xmlsoap.org/soap/encoding/
1.2	**Enveloppe**	http://www.w3.org/2003/05/soap-envelope/
	Encodage	http://www.w3.org/2003/05/soap-encoding/

Exemple 1 :

Déclaration de l'enveloppe d'un message SOAP écrit dans la version 1.1 et contenant un attribut ***encodingStyle*** :

```
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    ...
</SOAP-ENV:Envelope>
```

Exemple 2 :

Déclaration de l'enveloppe d'un message SOAP écrit dans la version 1.2 précédée d'une ligne XML (qui indique la version du langage XML utilisée) :

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    ...
</env:Envelope>
```

b) L'entête du message (*Header*) :

C'est une partie optionnelle (souvent) qui sert comme mécanisme d'extension du protocole SOAP. Elle est composée d'une ou plusieurs entrées (Header entry/ Header Block), chacune d'elles peut être destinée à un ou plusieurs nœuds SOAP (nœuds intermédiaires et/ou récepteur final du message).

Chaque entrée peut contenir des attributs, parmi ces attributs il y en a quatre qui ont une importance dans le traitement du message: *encodingStyle, role (actor)*, *mustUnderstand et relay :*

1. Attribut « *role* » :

L'attribut ***role*** (***actor*** pour la version 1.1) est utilisé pour désigner le destinataire concerné par le traitement de cette entrée. Un destinataire peut être: (un émetteur, un récepteur final ou bien aucun ou plusieurs intermédiaires).

Exemple 3 :

```
<SOAP-ENV:Header>
  <m:local xmlns:m =''http://www.w3edfRetD.edu/local/''
     soap:actor= http://www.w3edfRetD.edu/appli>
     <m:language> dk </m:language>
  </m:local>
</soap:Header>
```

Cet attribut peut prendre 3 valeurs particulières:

Valeur	Nom	Description
next	"http://www.w3.org/2003/05/soap-envelope/role/next"	Tous les nœuds intermédiaires y compris le récepteur final
none	"http://www.w3.org/2003/05/soap-envelope/role/none"	Aucun nœud n'est concerné
UltimateReceiver	"http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver"	Récepteur final uniquement

Dans le cas d'absence de cet attribut, uniquement le récepteur final est concerné.

Exemple 4 :

```
<env:Header>
    <p:oneBlock                              xmlns:p="http://example.com"
    env:role="http://example.com/Log"  env:mustUnderstand="true">
            ...
    </p:oneBlock>
```

```
    <q:anotherBlock                          xmlns:q="http://example.com"
    env:role="http://www.w3.org/2003/05/soap-envelope/ role/next">
            ...
    </q:anotherBlock>
    <r:aThirdBlock xmlns:r="http://example.com"> ...
    </r:aThirdBlock>
</env:Header>
```

2. L'attribut « mustUnderstand » :

Cet attribut est utilisé pour déterminer si la compréhension de l'entrée par le nœud concerné est obligatoire ou non.

Exemple 5 :

```
<soap:Header>
     <ns1:preferences soap:mustUnderstand='1'>
          <genre>Noir</genre>
          <format>WMV</format>
     </ns1:preferences>
     <ns2:priority>
          <level>low</level>
     </ns2:priority>
</soap:Header>
```

Dans cet exemple l'entrée « preferences » est obligatoire (attribut ***soap:mustUnderstand*** à "1."). Le service traitant ce message doit reconnaître ce header et le traiter correctement. Si ce header ne peut être traité correctement parce que le récepteur n'en a pas la capacité, alors un message d'erreur SOAP (***SOAP fault message***) doit être renvoyé.

Exemple 2.5:

```
<SOAP-ENV:Header>
    <t:Transaction xmlns:t="some-URI" SOAP-ENV:mustUnderstand="0">
        5
    </t:Transaction>
</SOAP-ENV:Header>
```

Exemple2.6:

```
<soap:Header>
    <m:User xmlns:m="http://www.exemple.com/rights/"
      soap:actor="http://www.exemple.com/rights/RightsManager">
        Toto
    </m:User>
    <m:Session xmlns:m="http://www.exemple.com/session/"
      soap:mustUnderstand="1">12AE3C
    </m:Session>
    <m:Language xmlns:m="http://www.exemple.com/lang/"
        soap:actor="http://schemas.xmlsoap.org/soap/next"
        soap:mustUnderstand="0"> FR
    </m:Language>
</soap:Header>
```

Elément USER → destination = Nœud: *RightManager*
Elément Session → destination = Nœud Final

Elément Lang → destination = Nœud prochain.

3. L'attribut « encodingStyle » :

Il a le même rôle que celui détaillé dans l'enveloppe.

4. L'attribut « relay » :

Il détermine si une entrée dans l'entête d'un message SOAP arrivé chez un intermédiaire doit être réacheminée ou non lorsque le traitement de cette entrée est impossible par le nœud actuel.

Cet attribut ne prend effet que sur les entrées non obligatoires destinées au nœud intermédiaire actuel.

Exemple 2.7:

```
<m:Language xmlns:m="http://www.example.com/lang/"
soap:role="http://www.w3.org/2003/05/soap-envelope/role/next"
soap:mustUnderstand="0"
soap:relay="true">
     FR
</m:Language>
```

5. Résumé du comportement du réacheminement :

Valeur de l'attribut « role »		Comportement du nœud	
Valeur abrégée	**Nœud concerné par le traitement ?**	**Compris & traité ?**	**Réacheminé ?**
next	Oui	Oui	Non, sauf si réinséré
		Non	Non, sauf si relay ="true"
Défini par l'utilisateur	Oui	Oui	Non, sauf si réinséré
		Non	Non, sauf si relay ="true"
	Non	-	Oui
ultimateReceiver	Oui	Oui	-
		Non	-
	Non	-	Oui
none	Non	-	Oui

c) Le corps du message (*Body*) :

C'est le bloc qui contient le corps du message. Il doit absolument être présent de manière unique dans chaque message et être contenu dans le bloc ***Envelope***. Les entrées de cette partie sont destinées toutes au récepteur final du message.

Chaque entrée peut contenir l'attribut ***encodingStyle*** défini plus haut.
Le protocole SOAP n'exige l'utilisation d'aucune entrée prédéfinie dans le corps du message, sauf que pour le corps d'un message d'erreur qui ne doit contenir que l'entrée « ***fault*** » qui sert à reporter et expliquer la nature d'erreur rencontrée (voir en bas).

Exemple 3.1:

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header> ...
  </env:Header>
  <env:Body>
    <x:Book xmlns:x="http://www.examples.ws/">
      <x:Title>
      Service-Oriented Architecture
      A Field Guide to Integrating XML and Web services
      </x:Title>
    </x:Book>
  </env:Body>
</env:Envelope>
```

2. Les messages d'erreur [3][4][5] :

Les messages d'erreur sont des messages SOAP envoyés suite à une erreur dans le traitement d'un message de requête, ils servent à expliquer le type d'erreur à l'expéditeur du message requête.

Le corps d'un message d'erreur est constitué d'une entrée unique appelée l'entrée "Fault". Cette entrée sert à définir le type du message d'erreur, et à donner plus de détails sur cette erreur.

Selon le type d'erreur signalée, l'entête du message peut exister ou non. S'il existe il doit contenir des entrées spécifiques définies par le protocole SOAP.

a) Structure de l'entrée Fault:

Une entrée ***fault*** peut contenir les éléments suivants (2 au minimum) dans l'ordre :

1. L'élément « Code » :

C'est un élément obligatoire, constitué d'un nom local, suivi d'un URI de type "http://www.w3.org/2003/05/soap-envelope", et ensuite d'un ou de deux éléments fils: « Value » (obligatoire) et « Subcode » (optionnel).

- ***Value:*** constituée d'un nom local ([local name] of Value) et d'un URI de type "http://www.w3.org/2003/05/soap-envelope". Sa valeur peut être une des valeurs prédéfinies de type ***xs:QName*** qui donne le code d'erreur comme défini par SOAP, ou bien un code personnalisé donné par une application.
- ***Subcode:*** formé de la même manière que ***code*** s'il existe.

2. L'élément « Reason » :

C'est un élément obligatoire aussi, qui contient une chaîne de caractères explication l'erreur destinée à être comprise par l'utilisateur. SOAP 1.2 supporte de nombreuses raisons avec un support multilingue. Elle est formée d'un nom local ([local name] of Reason) suivi d'un espace de nom de type décrit précédemment, et d'un ou de plusieurs éléments textes (***text element***) qui ont des valeurs différentes dans leurs attributs ***xml:lang***.

- ***Elément texte:*** formé d'un nom local du texte ([local name] of Text) suivi d'un URI du genre "http://www.w3.org/2003/05/soap-envelope", et d'un attribut d'information mandataire avec un nom local ([local name] of lang) et un URI du genre "http://www.w3.org/XML/1998/namespace", et ensuite d'un nombre quelconque d'éléments caractères. Chaque élément texte est de type ***env:reasontext***.

3. L'élément « Node » :

Requis pour tous les nœuds SOAP, excepté l'ultime destinataire. Il contient un URI décrivant le nœud ayant entraîné l'erreur. Il est formé d'un nom local ([local name] of Node) suivi d'un espace de nom "http://www.w3.org/2003/05/soap-envelope". Il est de type : xs:anyURI.

4. L'élément « Role » :

Requis pour tous les nœuds SOAP, excepté pour l'ultime destinataire. C'est un URI décrivant la source de l'erreur (le rôle du nœud opérant sur le message pendant l'apparition de l'erreur). Formé d'un nom local ([local name] of Role), suivi d'un espace de nom = "http://www.w3.org/2003/05/soap-envelope". Son type est de *xs:anyURI*.

5. L'élément « Detail » :

Requis si le corps de l'erreur SOAP ne peut pas être traité (erreurs spécifiques aux applications). Doit fournir des éléments sur le corps des éléments SOAP ayant échoué. Il est formé d'un nom local ([local name] of Detail), d'un espace de nom de "http://www.w3.org/2003/05/soap-envelope" et d'un ensemble d'attribut s et d'éléments d'information fils.

Version 1.2		Version 1.1
Code	**Value**	**Faultcode**
	Subcode	
Reason		**Faultstring**
Node		---
Detail		Detail
Role		Faultactor

Exemple 3.2:

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
<s:Body>
<s:Fault>
<faultcode xmlns="">s:Client</faultcode>
<faultstring xml:lang="fr-FR" xmlns="">
Une opération invalide s'est produite.
</faultstring>
</s:Fault>
</s:Body>
</s:Envelope>
```

b) Codes d'erreurs:

Il existe 5 groupes de code d'erreur (4 dans la version 1.1) [3][4]:

Code d'erreur		Désignation
Version 1.2	**Version 1.1**	
VersionMismatch	VersionMismatch	Erreur dans l'enveloppe (espace de nom ou nom local)
MustUnderstand	MustUnderstand	Elément dans l'en-tête n'est pas compris

		ou ne peut être traité
DataEncodingUnknown	--	Encodage d'un élément dans l'entête ou dans le corps non supporté par le nœud traitant
Sender	Client	Message incorrecte (structure) ou bien ne contient pas assez d'informations (informations d'authentification ou de payement oublies)
Receiver	Server	Erreur dans le traitement par le nœud (exemple temps dépassé)

Exemple 3.3:

```
<env:Body>
  <env:Fault>
    <env:Code>
      <env:Value>
        env:VersionMismatch
      </env:Value>
    </env:Code>
    <env:Reason>
      <env:Text xml:lang="en">
        Versions do not match
      </env:Text>
    </env:Reason>
  </env:Fault>
</env:Body>
```

1. Erreur « VersionMismatch » :

Quand un nœud génère une erreur avec ***value*** de ***code*** ="***VersionMismatch***" il devrait fournir obligatoirement une entrée d'entête appelée "***Upgrade***" dans le message d'erreur généré qui contient une liste ordonnée d'espaces de noms représentant les versions supportées.

Un élément ***Upgrade*** est constitué d'un nom local *([local name] of Upgrade*), un espace de nom = "*http://www.w3.org/2003/05/soap-envelope*" et un ou plusieurs éléments "***SupportedEnvelope***".

Un élément ***Upgrade*** ne doit jamais avoir d'attribut ***encodingStyle***.

Un élément ***SupportedEnvelope*** est constitué d'un nom local *([local name] of SupportedEnvelope*), un espace de nom = "*http://www.w3.org/2003/05/soap-envelope*" et d'un attribut "***qname***" de type *xs:Qname* qui doit avoir comme valeur un espace de nom de l'enveloppe supporté par ce noeud.

Exemple 3.4:

```
<?xml version="1.0" ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    xmlns:xml="http://www.w3.org/XML/1998/namespace">
 <env:Header>
  <env:Upgrade>
```

```
    <SupportedEnvelope qname="ns1:Envelope"
       xmlns:ns1="http://www.w3.org/2003/05/soap-envelope">
    <SupportedEnvelope qname="ns2:Envelope"
xmlns:ns2="http://schemas.xmlsoap.org/soap/envelope/">
  </env:Upgrade>
 </env:Header>
 <env:Body>
  <env:Fault>
<env:Code><env:Value>env:VersionMismatch</env:Value></env:Code>
    <env:Reason><env:Text xml:lang="en">Version Mismatch</env:Text>
    </env:Reason>
   </env:Fault>
 </env:Body>
</env:Envelope>
```

2. Erreur « mustUnderstand » :

Lorsqu'un nœud SOAP génère une faute avec une valeur ***Value*** du ***Code*** fixée à "***env:MustUnderstand***", il devrait fournir un bloc d'en-tête ***NotUnderstood*** dans le message de faute généré. Le bloc d'en-tête ***NotUnderstood*** (***Misunderstood*** dans la version 1.1), comme décrit plus bas, détaille les noms qualifiés XML des blocs d'en-tête particuliers non compris.

Un nœud SOAP pourrait générer une faute SOAP dès qu'un ou plusieurs blocs d'en-tête SOAP n'ont pas été compris dans un message SOAP.

Chaque entrée d'en-tête ***NotUnderstood*** est constituée d'un nom local ([local name] ***NotUnderstood***), espace de nom de "http://www.w3.org/2003/05/soap-envelope" et d'un attribut ***qname*** de type *xs:qname* et qui à comme valeur le nom qualifié XML d'un bloc d'en-tête que le nœud n'a pas réussi à comprendre.

Un élément ***NotUnderstood*** ne doit jamais avoir d'attribut ***encodingStyle***.

Exemple3.5 (message non compris)

```
<?xml version="1.0" ?>
<env:Envelope xmlns:env='http://www.w3.org/2003/05/soap-envelope'>
  <env:Header>
    <abc:Extension1 xmlns:abc='http://example.org/2001/06/ext'
       env:mustUnderstand='true'>
    <def:Extension2 xmlns:def='http://example.com/stuff'
       env:mustUnderstand='true' >
  </env:Header>
  <env:Body>
    . . .
  </env:Body>
</env:Envelope>
```

Si le récepteur final du message ne comprend pas les deux éléments d'en-tête *abc:Extension1* et *def:Extension2* il génère le message d'erreur dans l'exemple suivant:

Exemple3.6 : (message d'erreur)

```
<?xml version="1.0" ?>
<env:Envelope xmlns:env='http://www.w3.org/2003/05/soap-envelope'
```

```
        xmlns:xml='http://www.w3.org/XML/1998/namespace'>
 <env:Header>
  <env:NotUnderstood qname='abc:Extension1'
                     xmlns:abc='http://example.org/2001/06/ext' >
  <env:NotUnderstood qname='def:Extension2'
                     xmlns:def='http://example.com/stuff' >
 </env:Header>
 <env:Body>
  <env:Fault>
   <env:Code><env:Value>env:MustUnderstand</env:Value></env:Code>
   <env:Reason>
    <env:Text xml:lang='en'>One or more mandatory
                SOAP header blocks not understood
    </env:Text>
   </env:Reason>
  </env:Fault>
 </env:Body>
</env:Envelope>
```

c) Transition de version de SOAP/1.1 à SOAP Version 1.2

Le problème d'interaction entre un nœud SOAP1.1 et un autre nœud SOAP1.2 est géré de la manière suivante [3]:

1. Un nœud SOAP/1.1 recevant un message SOAP Version 1.2 va générer selon SOAP/1.1 une faute SOAP de décalage de version basée sur la construction de message de SOAP/1.1. C'est-à-dire que l'enveloppe aura un nom local de Envelope et un nom d'espace de noms de "http://schemas.xmlsoap.org/soap/envelope/".
2. Un nœud SOAP Version 1.2 recevant un message SOAP/1.1 pourrait:
   - Soit traiter le message comme un message SOAP/1.1 (s'il le supporte),
   - Sinon générer une faute SOAP de décalage de version basée sur la construction de message de SOAP/1.1 selon la sémantique de SOAP/1.1 et utilisant une liaison de SOAP1.1 au protocole sous-jacent. La faute SOAP devrait inclure un bloc d'en-tête Upgrade tel que défini dans cette la version 12, indiquant le support de SOAP en version 1.2. Ceci permet à un nœud SOAP/1.1 d'interpréter correctement la faute SOAP générée par le nœud SOAP 1.2.

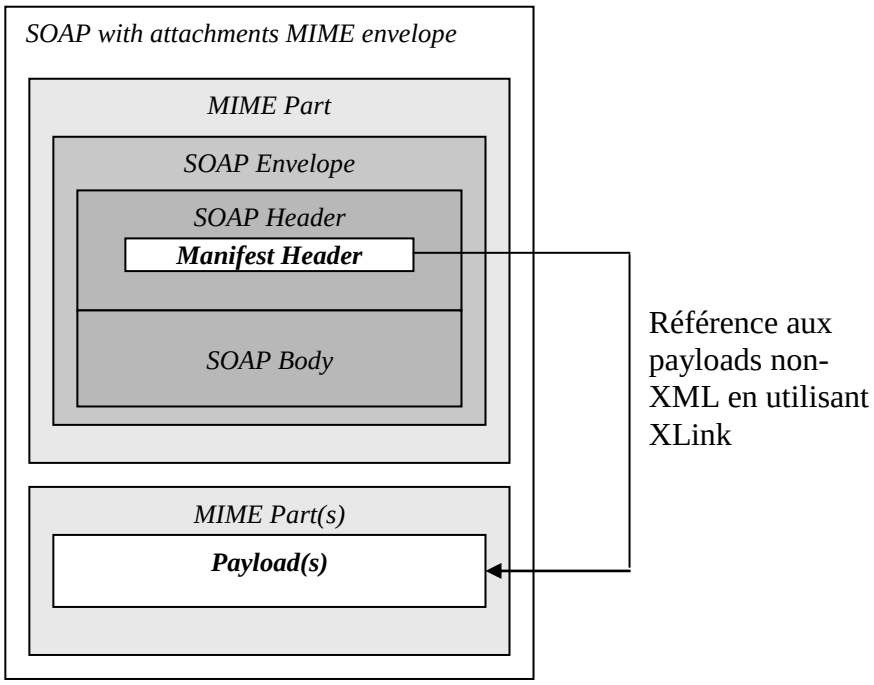
3. Message avec pièce jointe [1] :

L'utilisation de SOAP avec des données non XML est parfois très utile, par exemple si on désire transmettre des images à partir d'une caméra à travers une liaison sans fil en utilisant SOAP vers un serveur client.

Le support des données non XML requiert un packaging complémentaire du message qui peut être fourni par une structure ***MIME multipart*** et impacte la liaison d'un message à son protocole de transport sous-jacent.

L'enveloppe MIME contient un ensemble de parties MIME individuelles, la 1ère contient le message SOAP qui doit comporter un bloc ***Manifest Header*** crée par le ***Handler Message Manifest***. La 2ème partie MIME contient les ***payloads*** qui peuvent être des documents XML ou d'autres formats.

Un message avec attachement est structuré comme suit:



Structure d'un message avec attachements

Le ***Manifest Header*** peut contenir les éléments qui référencent les parties MIME séparées en utilisant leurs identificateurs de contenu. Ceci peut être réalisé en utilisant les références XLink.

Exemple 4:

```
<env:Envelope xmlns:env='http://www.w3.org/2001/12/soap-envelope'
    <env:Header>
      <n:Manifest xmlns:n="http://exampleorg/manifest">
           <n:Reference n:id="image01"
                   Xlink:href="cid:payload-1"
                   Xlink:role="http://example.org/image">
              <n.description> Photo numero 1 </n:description>
           </n:Reference>
           <n:Reference n:id="image02"
              Xlink:href="cid:payload-2"
              Xlink:role="http://example.org/image">
              <n.description> Photo numero 2 </n:description>
           </n:Reference>
      </n:Manifest>
    </env:Header>
    <env:Body>
    . . .
    </env:Body>
</env:Envelope>
```

Références :

[1] Jean-Marie Chauvet « Services Web avec SOAP, WSDL, UDDI, ebXML..» Eyrolles Librairie accessible par « https://www.eyrolles.com/Chapitres/9782212110470/chap02.pdf »

[2] SOAP - Quick Guide: « https://www.tutorialspoint.com/soap/index.htm »

[3] W3C SOAP 1.2 : « https://www.w3.org/TR/soap12-part1/ » et « https://www.w3.org/TR/soap12-part2/ »

[4] W3C Simple Object Access Protocol (SOAP) 1.1: « https://www.w3.org/TR/2000/NOTE-SOAP-20000508/ »

[5] Web Service(WS) Security Tutorial with SOAP Example : «https://www.guru99.com/security-web-services.html».