

CH3.PROTOCOLE SOAP :

1- Introduction

2- Principes fondamentaux

1- Framework de messagerie

2- Invocation d'objets distants

3- Acheminement de messages

3- Structure de message SOAP

1- INTRODUCTION :

- SOAP (***Simple Object Access Protocol***) : est un protocole standardisé par W3C, qui assure le transfert des messages et les appels de procédures RPC.
- Le principal objectif du SOAP est de permettre la normalisation des échanges de données.
- Il gère cette normalisation en se basant sur deux standards :
 - ✓ Données encodées dans XML, et transportées sous forme de messages SOAP
 - ✓ Transport assuré souvent par le protocole http (https), ou bien les protocoles SMTP, POP, et FTP dans certains cas
- Il permet une interopérabilité avec divers environnements logiciels (.NET, Java/RMI, CORBA, ...) quel que soit leur plate-forme d'exécution.
- Il permet l'échange de données que ce soit en mode synchrone (requête/réponse) ou asynchrone (accompagnement des processus).

2- PRINCIPES FONDAMENTAUX :

Framework de messagerie :

Nœud SOAP (Nodes):

Un nœud SOAP est l'infrastructure technique qui alimente les différents scénarios de communication entre services.



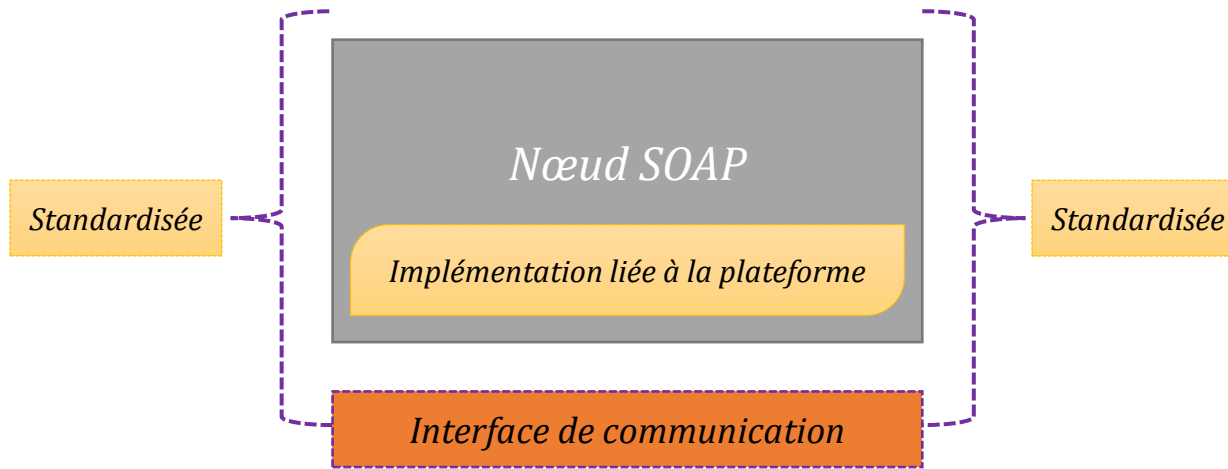
Nœud SOAP

- Un nœud SOAP peut être vu comme étant le processeur logique identifier par URI
- Il permet de recevoir, transmettre, et exécuter (manipuler) un ensemble d'opérations sur les messages

2- PRINCIPES FONDAMENTAUX :

Framework de messagerie :

Nœud SOAP (Interface de communication):



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

La standardisation de l'interface lui permet de communiquer dans différents environnements.

2- PRINCIPES FONDAMENTAUX :

Framework de messagerie :

Nœud SOAP (Classes):

- Selon le rôle joué par un nœud dans une communication, celui-ci peut être classé en :

➤ **Expéditeur initial** (Demandeur de service) (*Initial Sender*) ou (*Service Requestor*).

➤ **Récepteur Final** (Fournisseur de service) (*Ultimate Receiver*) ou (*Service Provider*).

➤ **Nœud intermédiaire** (*Intermediary*): il joue le rôle de récepteur/expéditeur en même temps.

2- PRINCIPES FONDAMENTAUX :

Framework de messagerie :

Invocation de service d'objets distants :

- 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.
- **Exemple:**
 - ✓ appel du service ***GetLastTradePrice(DIS)*** : un exemple d'un appel de procédure distante (RPC) pour obtenir le prix d'une action en bourse (marché financier).
 - ✓ "DIS" est l'argument envoyé au serveur pour lui dire : *"Donne-moi le dernier prix de l'action Disney."*
 - ✓ le suivant code représente une **requête HTTP POST** transportant un message **SOAP**.

2- PRINCIPES FONDAMENTAUX :

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/2003/11/soap-envelope">

<env:Body>

<m: GetLastTradePrice

Env:encodingStyle=="http://www.w3.org/2003/11/soap-encoding">

Xmlns:m="http://example.org/2001/06/quotes">

<symbol>DIS</symbol>

</m: GetLastTradePrice>

</env:Body>

</env:Envelope>

L'adresse du serveur qui

Indique que le contenu
est du XML

le nom de la **méthode
(fonction)** que l'on veut
exécuter sur le serveur

l'**argument** (le paramètre
DIS) passé à la fonction

2- PRINCIPES FONDAMENTAUX :

Framework de messagerie :

Invocation d'objets distants (étapes d'invocation)

1. 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 HTTP POST).
2. Le message est transmis au serveur via une connexion HTTP
3. Le serveur reçoit le message et fait appel à l'objet concerné
4. L'objet effectue le traitement demandé et renvoie les résultats au serveur SOAP
5. Le serveur SOAP encapsule les résultats et les envoie dans un document XML au client via le protocole HTTP (HTTP Response).
6. Le client SOAP décode le message et envoie les résultats au demandeur initial.

2- PRINCIPES FONDAMENTAUX :

Framework de messagerie :

Messages SOAP et acheminement :

A l'arrivée d'un message a un nœud:

- Vérifier la structure du message: enveloppe, entête, et corps du message.
- Traiter les entrées d'entêtes destinées à lui-même.
- S'il y a des entrées mandataires qui ne peuvent être traitées; il renvoi un message d'erreur à l'expéditeur.
- Sinon; toute entrée traitée (obligatoire ou non); ou bien non traitée (n'est pas obligatoire) sera supprimée.
- Si le message est destiné à autrui; il réachemine le message contenant le corps tel qu'il est, mais l'entête garde uniquement les entrées qui doivent rester.
- Si le message est destiné à lui-même; il traite ainsi le corps du message; si celui-ci nécessite une réponse il renvoi un message de réponse à l'expéditeur initial.

3- STRUCTURE DE MESSAGE SOAP :

Un message SOAP est un document XML; donc il ne transporte que du texte.

Cependant, le transport de données de différents formats (image, vidéo, son, pdf,...) est possible comme-même, en y rajoutant à ce fichier les données qu'on souhaite transporter sous forme de pièces-jointes.

➤ Le message qui résulte du 1^{er} cas est un
« *message SOAP simple* ».



Document xml

➤ Le message qui résulte du 2^{ème} cas est un
« *message avec pièces jointes* ».

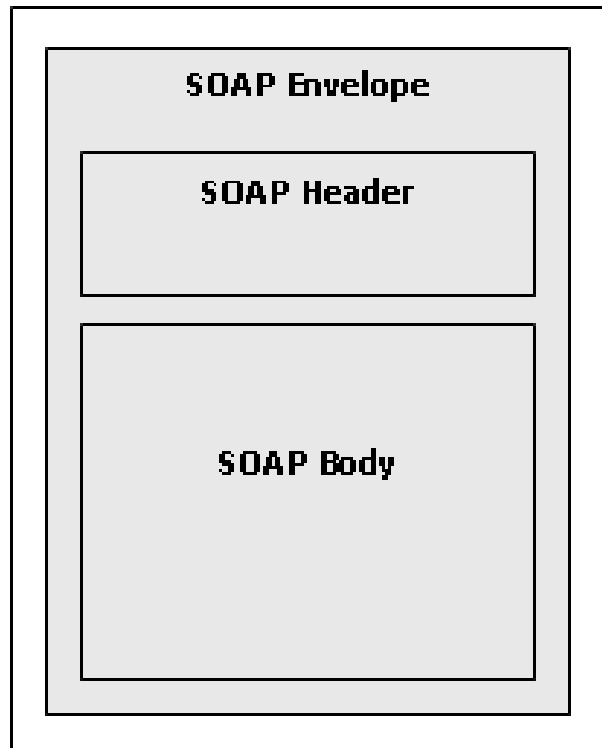


Document MIME

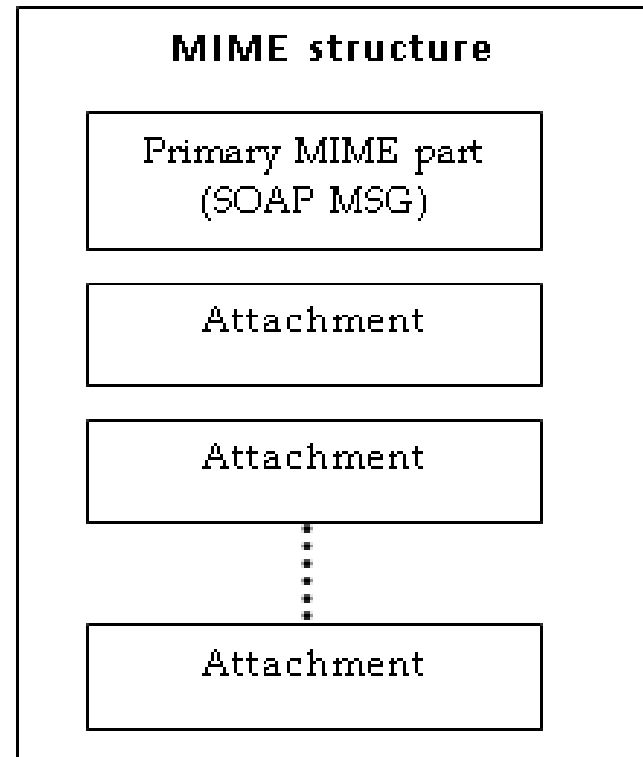
MIME: *Multipurpose Internet Mail Extensions*

3- STRUCTURE DE MESSAGE SOAP :

Un message SOAP est structuré comme suit :



Message SOAP simple



Message SOAP avec pièces jointes

3- STRUCTURE DE MESSAGE SOAP :

Structure de Message SOAP simple :

Un message SOAP simple est composé des éléments suivants :

- Une enveloppe (*envelope*): c'est le récipient qui sert à contenir les autres éléments (entête et corps).
- Un entête (*header*) : c'est un élément souvent facultatif, il contient des données (***entrées entête***) (block-header) à destination des différents nœuds tout au long du chemin de transport (***nœuds intermédiaires et récepteur final***).
- Le corps du message (*body*) : c'est un élément clé, il englobe toutes les informations à destination du ***récepteur final***.

3- STRUCTURE DE MESSAGE SOAP :

Squelette d'un message SOAP :

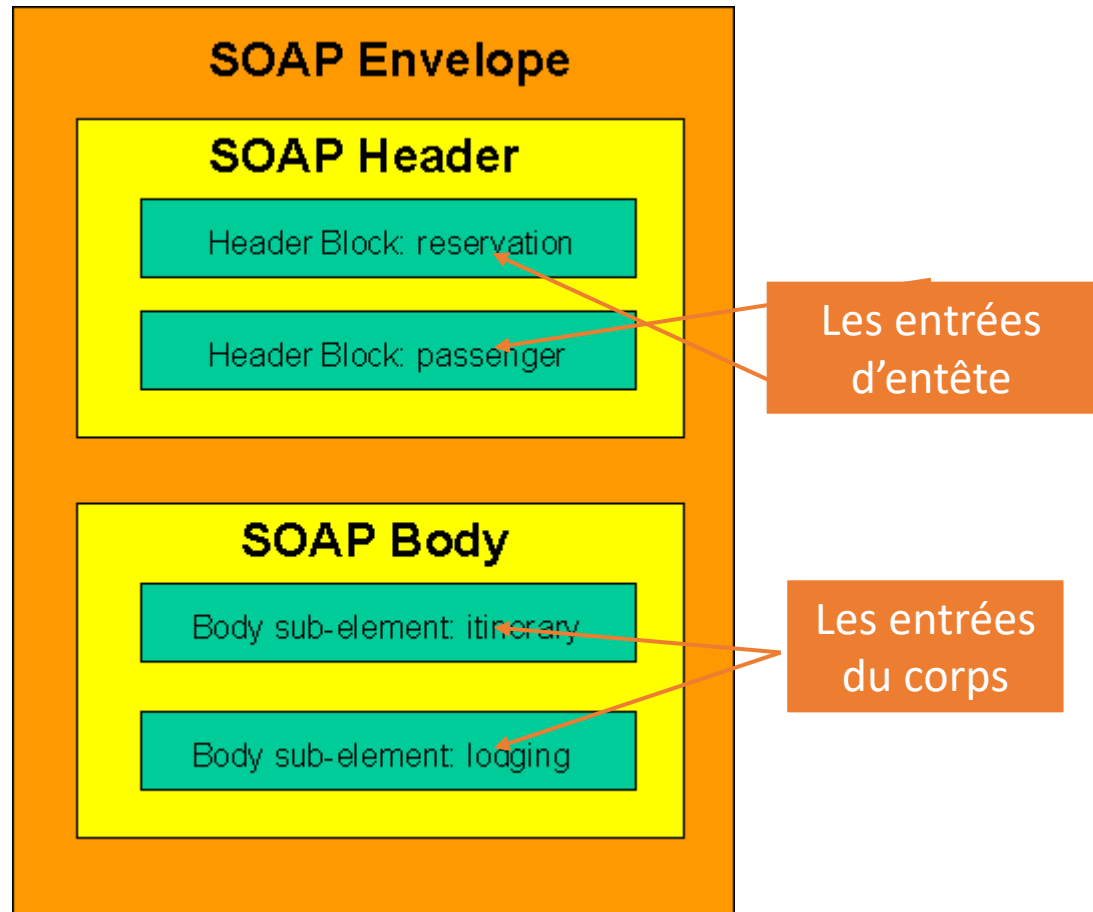
```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">  
  <env:Header>  
    <!-- Métadonnées du message --> (optionnel)  
  </env:Header>  
  
  <env:Body>  
    <!-- Données métier --> (obligatoire)  
  </env:Body>  
</env:Envelope>
```

3- STRUCTURE DE MESSAGE SOAP :

Message SOAP simple :

Exemple :

- un **message de requête SOAP** utilisé dans le contexte d'un **service web de réservation de voyage**
- Une **demande envoyée à un service web** pour réserver un trajet (aller + retour) pour un passager.



3- STRUCTURE DE MESSAGES SOAP :

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <m:reservation xmlns:m="http://travelcompany.example.org/reservation"
      env:role="http://www.w3.org/2003/05/soap-envelope/role/next">
      <m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d</m:reference>
      <m:dateAndTime>2007-11-29T13:20:00.000-05:00</m:dateAndTime>
    </m:reservation>
    <n:passenger xmlns:n="http://mycompany.example.com/employees"
      env:role="http://www.w3.org/2003/05/soap-envelope/role/next">
      <n:name>Fred Bloggs</n:name>
    </n:passenger>
  </env:Header>
  <env:Body>
    <p:itinerary xmlns:p="http://travelcompany.example.org/reservation/travel">
      <p:departure>
        <p:departing>New York</p:departing>
        <p:arriving>Los Angeles</p:arriving>
        <p:departureDate>2007-12-14</p:departureDate>
        <p:departureTime>late afternoon</p:departureTime>
        <p:seatPreference>aisle</p:seatPreference>
      </p:departure>
      <p:return>
        <p:departing>Los Angeles</p:departing>
        <p:arriving>New York</p:arriving>
        <p:departureDate>2007-12-20</p:departureDate>
        <p:departureTime>mid-morning</p:departureTime>
        <p:seatPreference></p:seatPreference>
      </p:return>
    </p:itinerary>
  </env:Body>
</env:Envelope>
```

L'enveloppe

3- STRUCTURE DE MESSAGE SOAP :

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

```
<env:Header>
```

```
<m:reservation xmlns:m="http://travelcompany.example.org/reservation"
  env:role="http://www.w3.org/2003/05/soap-envelope/role/next">
```

```
<m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d</m:reference>
```

```
<m:dateAndTime>2007-11-29T13:20:00.000-05:00</m:dateAndTime>
```

```
</m:reservation>
```

```
<n:passenger xmlns:n="http://mycompany.example.com/employees"
```

```
  env:role="http://www.w3.org/2003/05/soap-envelope/role/next">
```

```
<n:name>Fred Bloggs</n:name>
```

```
</n:passenger>
```

```
</env:Header>
```

```
<env:Body>
```

```
<p:itinerary xmlns:p="http://travelcompany.example.org/reservation/travel">
```

```
<p:departure>
```

```
<p:departing>New York</p:departing>
```

```
<p:arriving>Los Angeles</p:arriving>
```

```
<p:departureDate>2007-12-14</p:departureDate>
```

```
<p:departureTime>late afternoon</p:departureTime>
```

```
<p:seatPreference>aisle</p:seatPreference>
```

```
</p:departure>
```

```
<p:return>
```

```
<p:departing>Los Angeles</p:departing>
```

```
<p:arriving>New York</p:arriving>
```

```
<p:departureDate>2007-12-20</p:departureDate>
```

```
<p:departureTime>mid-morning</p:departureTime>
```

```
<p:seatPreference></p:seatPreference>
```

```
</p:return>
```

```
</p:itinerary>
```

```
</env:Body>
```

```
</env:Envelope>
```

L'entête

Les entrées
d'entête

3- STRUCTURE DE MESSAGE SOAP :

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <m:reservation xmlns:m="http://travelcompany.example.org/reservation"
      ...
    </m:reservation>
    <n:passenger xmlns:n="http://mycompany.example.com/employees"
      ...
    </n:passenger>
  </env:Header>
  <env:Body>
    <p:itinerary xmlns:p="http://travelcompany.example.org/reservation/travel">
      <p:departure>
        <p:departing>New York</p:departing>
        <p:arriving>Los Angeles</p:arriving>
        <p:departureDate>2007-12-14</p:departureDate>
        <p:departureTime>late afternoon</p:departureTime>
        <p:seatPreference>aisle</p:seatPreference>
      </p:departure>
      <p:return>
        <p:departing>Los Angeles</p:departing>
        <p:arriving>New York</p:arriving>
        <p:departureDate>2007-12-20</p:departureDate>
        <p:departureTime>mid-morning</p:departureTime>
        <p:seatPreference></p:seatPreference>
      </p:return>
    </p:itinerary>
    <q:lodging xmlns:q="http://travelcompany.example.org/reservation/hotels">
      <q:preference>none</q:preference>
    </q:lodging>
  </env:Body>
</env:Envelope>
```

L'entête

Le corps

Les entrées
du corps

3- STRUCTURE DE MESSAGE SOAP :

1) L'enveloppe du message :

- C'est l'élément récipient qui englobe tous les autres éléments du message.
- Elle sert à séparer la partie soap des différents entêtes ajoutés par les protocoles de transport (http par exemple).

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn

<?xml version='1.0' Encoding='UTF-8' ?>

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

    ...

</env:Envelope>
```

- Elle commence par un mot clé (préfixe) « **env** » suivi par nom local « *envelope* ».
- Ensuite un espace de nom précédé par le mot (attribut) « xmlns »
- Cet espace de nom indique la version dans laquelle est écrit ce message.

3- STRUCTURE DE MESSAGE SOAP :

1) L'enveloppe du message :

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

L'espace de nom "*http://www.w3.org/2003/05/soap-envelope*" indique la version 1.2 du SOAP, tandis que "*http://schemas.xmlsoap.org/soap/envelope*" indique la version 1.1.

D'autres espaces de nom peuvent apparaître aussi à la suite de cette déclaration (autres attributs):

```
encodingStyle= "http://www.w3.org/2003/05/soap-encoding"
```

```
xml="http://www.w3.org/XML/1998/namespace"
```

3- STRUCTURE DE MESSAGE SOAP :

1) L'enveloppe du message

L'espace de nom introduit par l'attribut « `encodingStyle` » indique les règles de sérialisation (le codage) utilisées dans le message SOAP.

```
encodingStyle= " http://www.w3.org/2003/05/soap-encoding"
```

Son effet porte sur l'enveloppe et sur tous ses éléments descendants (c'est-à-dire sur tout le message).

La ***valeur de l'attribut*** est une liste ordonnée d'un ou plusieurs URI identifiant la ou les règles de sérialisation de message SOAP :

```
encodingStyle= "http://schemas.xmlsoap.org/soap/encoding"  
"http://my.host/encoding/restricted http://my.host/encoding"
```

3- STRUCTURE DE MESSAGE SOAP :

1) nL'enveloppe du message :

- Le nom local utilisé dans la déclaration de l'enveloppe « <env:Envelope> » est le même pour chaque élément (message) défini par le protocole SOAP.
- Pour les éléments ajoutés par les applications le nom local diffère de celui de l'enveloppe « **m:reservation** ».

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <m:reservation xmlns:m="http://travelcompany.example.org/reservation"
                  env:role="http://www.w3.org/2003/05/soap-envelope/role/next">
      . . .
    </m:reservation>
    . . .
  </env:Header>
  <env:Body>
    . . .
  </env:Body>
</env:Envelope>
```

3- STRUCTURE DE MESSAGE SOAP :

2) L'entête du message :

C'est une partie souvent optionnelle. Néanmoins cette caractéristique ne diminue en aucun cas l'importance du rôle joué par cette partie.

L'entête SOAP fournit un mécanisme flexible pour étendre les fonctionnalités (à savoir *l'authentification, la gestion des transactions, le paiement, etc..*) d'un message de manière décentralisée et modulaire sans connaissance préalable entre les parties communicantes.

L'entête SOAP composé 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*).

3- STRUCTURE DE MESSAGE SOAP :

2) L'entête du message :

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <m:reservation xmlns:m="http://travelcompany.example.org/reservation"
      env:role="http://www.w3.org/2003/05/soap-envelope/role/next">
      <m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d</m:reference>
      <m:dateAndTime>2007-11-29T13:20:00.000-05:00</m:dateAndTime>
    </m:reservation>
    <n:passenger xmlns:n="http://mycompany.example.com/employees"
      env:role="http://www.w3.org/2003/05/soap-envelope/role/next">
      <n:name>Fred Bloggs</n:name>
    </n:passenger>
  </env:Header>
  ...
</env:Envelope>
```

Chaque *entrée* d'entête (reservation, passenger) est identifiée par son nom d'élément complet : **nom de l'entrée (préf + nom local) + l'URI**.

Tous les éléments enfants immédiats de cette entrée doivent avoir le même *nom local (préfixe)*, mais peuvent être qualifiés par un espace de nom et un style de codage différent.

3- STRUCTURE DE MESSAGE SOAP :

2) L'entête du message :

Chaque entrée peut contenir zéro ou plusieurs attributs permettant de préciser quelques informations sur cette entrée (nature de l'entrée, le nœud concerné par le traitement, la nature du traitement ...)

Ces attributs sont :

- L'attribut « *role* » (« *actor* » pour la version 1.1).
- L'attribut « *mustUnderstand* ».
- L'attribut « *encodingStyle* ».
- L'attribut « *relay* ».

3- STRUCTURE DE MESSAGE SOAP :

2) L'entête du message :

➤ L'attribut « *role* » (« *actor* » pour la version 1.1) :

Utilisé pour désigner le nœud concerné par le traitement de cette entrée. Un destinataire peut être : un ou plusieurs nœuds intermédiaires et/ou le récepteur final.

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <m:reservation xmlns:m="http://travelcompany.example.org/reservation"
      env:role="http://www.w3.org/2003/05/soap-envelope/role/next">
      <m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d</m:reference>
      <m:dateTime>2007-11-29T13:20:00.000-05:00</m:dateTime>
    </m:reservation>
    <n:passenger xmlns:n="http://mycompany.example.com/employees"
      env:role="http://www.w3.org/2003/05/soap-envelope/role/next">
      <n:name>Fred Bloggs</n:name>
    </n:passenger>
  </env:Header>
  ...
</env:Envelope>
```

3- STRUCTURE DE MESSAGE SOAP :

2) L'entête du message :

➤ L'attribut « *role* » (« *actor* » pour la version 1.1) :

Les **valeurs** possibles de l'**attribut** « *role* » sont :

1. « **next** » : Chaque nœud qui reçoit le message doit examiner ce bloc. S'il le traite, il peut le supprimer ou le transmettre au suivant.
2. « **UltimateReceiver** » : destinée au récepteur final uniquement.
3. « **none** » : aucun nœud n'est concerné.
4. Autre : uniquement le nœud qui porte ce nom.

En cas de l'absence de cet attribut, uniquement le récepteur final sera concerné.

3- STRUCTURE DE MESSAGE SOAP :

2) L'entête du message :

➤ L'attribut « *role* » (« *actor* » pour la version 1.1) :

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

3- STRUCTURE DE MESSAGE SOAP :

2) L'entête du message :

➤ L'attribut « *mustUnderstand* » :

- Utilisé pour désigner si le traitement (la compréhension) de cette entrée est obligatoire ou non.
- Si *mustUnderstand*= " *true* " : le nœud concerné doit :
 - Traiter puis supprimer cette entrée, si le traitement est possible.
 - Générer un message d'erreur de type « *mustUnderstand* » et le renvoyer à l'expéditeur initial du message original.
- Si *mustUnderstand*=" *false* " : le nœud concerné doit traiter puis supprimer l'entrée si le traitement est possible ; ou bien uniquement la supprimer dans le cas contraire.
- L'absence de cet attribut est équivalent à la valeur *mustUnderstand*= "false".

3- STRUCTURE DE MESSAGE SOAP :

2) L'entête du message :

➤ L'attribut « *mustUnderstand* » :

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

3- STRUCTURE DE MESSAGE SOAP :

2) L'entête du message :

➤ L'attribut «*encodingStyle*» :

- Cet attribut est similaire a celui dans l'enveloppe, qui indique les règles de sérialisation (le codage)
- Son effet est valable sur l'entrée indiqué uniquement et ses sous-entrées à moins qu'une sous entrée redéfinit son propre codage.

L'absence de cet attribut dans une entrée signifie le suivi de celui de son élément ascendant.

3- STRUCTURE DE MESSAGE SOAP :

2) L'entête du message :

➤ L'attribut «*relay*» :

Indique 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 n'est pas possible par le nœud actuel.

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

Son absence est équivalente à « relay = false ».

3- STRUCTURE DE MESSAGE SOAP :

2) L'entête du message :

➤ L'attribut «*relay*» :

```
<env:Header>
  <m:TraceInfo xmlns:m="http://example.org/trace"
    env:role="http://example.com/Log"
    env:relay="true"
    env:mustUnderstand="false">
    <m:id>12345</m:id>
  </m:TraceInfo>
</env:Header>
```

- Comme `relay="true"`, le proxy transmet le message au nœud suivant **en gardant l'en-tête `<m:TraceInfo>` intact**.
- Si `relay` avait été à « `false` », le proxy aurait supprimé cet en-tête avant de faire suivre le message au récepteur final.

3- STRUCTURE DE MESSAGE SOAP :

Résumé du comportement du réacheminement :

Attribut <i>role</i>		Attribut <i>mustUnderstand</i>		Comportement du nœud	
role =	Nœud = concerné	mustUnd erstand =	Traitement = possible	Comportement sur l'entrée	Message résultat
<i>next</i>	Oui	<i>true</i>	Oui	TS + pt réinsérée si NIA	Msg réacheminé si NI
			Non	/	Msg d'erreur
		<i>false</i>	Oui	TS + pt réinsérée si NIA	Msg réacheminé si NI
			Non	Sup + réinsérée si relay et NI; pt NIA	
<i>Défini par user</i>	Oui	Les mêmes cas que pour la valeur next			
	Non	true/false	/	Réacheminée si NI	Msg réacheminé si NI
<i>utilimateReceiver</i>	Oui	true	Oui	Traitée	Msg de rép / rien
			Non	/	Msg d'erreur
		false	Oui/Non	Traitée si possible	Msg de rép / rien
	Non	true/false	/	Réacheminée	Msg réacheminé
<i>none</i>	Non	Oui/Non	/	Réacheminée si NI	Msg réacheminé si NI

3- STRUCTURE DE MESSAGE SOAP :

3) Le corps du message :

- C'est une partie obligatoire, elle contient des entrées destinées tous au récepteur final.
- Ces entrées peuvent être:
 1. des données **vrac** brutes (sous forme d'entrée),
 2. un ou des appels RPC,
 3. le(s) résultat(s) de(s) appel RPC (s'il s'agit d'un message de réponse),
 4. ou bien l'entrée « *fault* » s'il s'agit d'un message d'erreur.
- Le protocole SOAP n'exige l'utilisation d'aucune entrée prédéfinie dans le corps du message, sauf lorsqu'il s'agit d'un message d'erreur .
- Lorsqu'il s'agit d'un message d'erreur, le corps du message ne doit contenir que l'entrée « *fault* ».
- Chaque entrée peut contenir l'attribut « *encodingStyle* » défini précédemment.

3- STRUCTURE DE MESSAGE SOAP :

3) Le corps du message :

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    ...
  </env:Header>
  <env:Body>
    <p:itinerary xmlns:p="http://travelcompany.example.org/reservation/travel">
      <p:departure>
        <p:departing>New York</p:departing>
        <p:arriving>Los Angeles</p:arriving>
        <p:departureDate>2007-12-14</p:departureDate>
        <p:departureTime>late afternoon</p:departureTime>
        <p:seatPreference>aisle</p:seatPreference>
      </p:departure>
      <p:return>
        <p:departing>Los Angeles</p:departing>
        <p:arriving>New York</p:arriving>
        <p:departureDate>2007-12-20</p:departureDate>
        <p:departureTime>mid-morning</p:departureTime>
        <p:seatPreference></p:seatPreference>
      </p:return>
    </p:itinerary>
    <q:lodging xmlns:q="http://travelcompany.example.org/reservation/hotels">
      <q:preference>none</q:preference>
    </q:lodging>
  </env:Body>
</env:Envelope>
```

3- STRUCTURE DE MESSAGE SOAP :

3) Le corps du message :

1. Message contenant des données **vrac** :

```
<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>
```

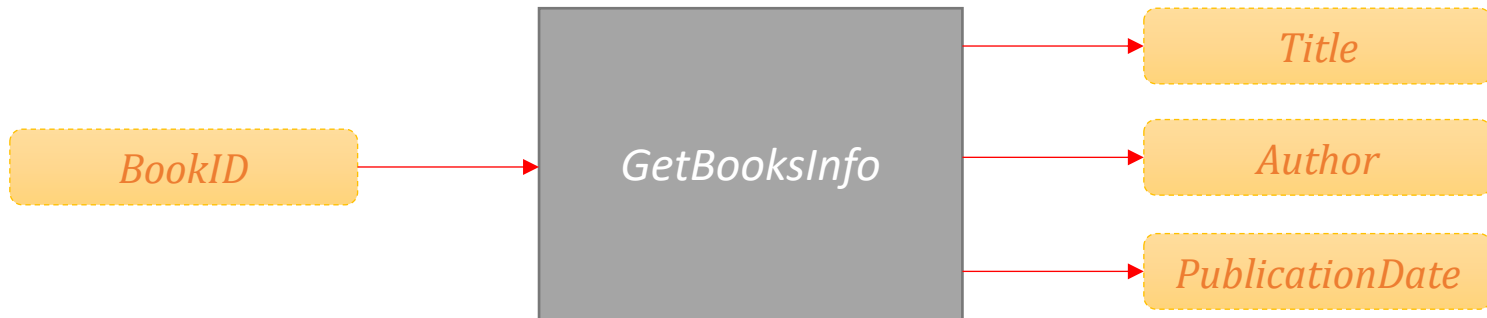
Dans cet exemple, ces informations sont considérées comme des **données en vrac** (sous forme d'entrée), car elles transmettent du contenu métier directement au récepteur.

3- STRUCTURE DE MESSAGE SOAP :

3) Le corps du message :

2. Message contenant un appel RPC :

Comme exemple, on considère la méthode « GetBooksInfo » définie par :



3- STRUCTURE DE MESSAGE SOAP :

3) Le corps du message :

2. Message contenant un appel RPC :

le message est une **requête** où un client demande au service "someorders" de lui fournir les détails d'un livre spécifique identifié par son ID.

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    ...
  </env:Header>
  <env:Body>
    <orders:GetBooksInfo xmlns:orders="http://www.someorders.com/orders">
      <BooksID> 978-0133858587 </BooksID>
    </orders:GetBooksInfo>
  </env:Body>
</env:Envelope>
```

Ici l'Objet / Service : orders et le Serveur (hôte probable) :
www.someorders.com

3- STRUCTURE DE MESSAGE SOAP :

3) Le corps du message :

3. Message de réponse :

- Dans une réponse RPC, La balise porte le nom de la méthode suivi du suffixe **"Response"**

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    ...
  </env:Header>
  <env:Body>
    <orders:GetBooksInfoResponse xmlns:orders="http://www.someorders.com/orders">
      <Title> Service-Oriented Architecture:
        Analysis and Design for Services and Microservices </Title>
      <Author> Thomas Erl </Author>
      <PublicationDate> December 12, 2016 </PublicationDate>
    </orders:GetBooksInfo>
  </env:Body>
</env:Envelope>
```

FIN