Services Web – SOAP

Fabrice Rossi

http://apiacoa.org/contact.html.

Université Paris-IX Dauphine

Plan du cours SOAP

1. les évolutions de SOAP
2. l'enveloppe SOAP
3. l'en-tête SOAP
4. messages d'erreurs
5. le modèle de données
6. la représentation XML du modèle de données
7. SOAP et HTTP
8. modèle RPC
9. discussion

SOAP

Plusieurs choses dans une même spécification (1.2) :

- l'**enveloppe** SOAP qui permet de représenter des messages en XML
- un cadre général pour implémenter l'échange de messages SOAP au dessus d'un protocole internet (*binding*)
- une implémentation du cadre général en HTTP
- un modèle de données SOAP et la traduction en XML des données qui obéissent à ce modèle (*encoding*)
- un modèle RPC (*Remote Procedure Call*) basé sur les messages et le modèle de données

Évolution de SOAP

L'ancêtre : XML-RPC

- 1998-1999
- `http://www.xmlrpc.com/spec`
- UserLand (avec Microsoft)
- propose :
  - un modèle de données basique avec une traduction très simple en XML :
    - type fondamentaux (entiers, réels, booléens, chaînes de caractères et dates)
    - tableaux
    - *structs*
  - un modèle purement RPC
  - un protocole basique : POST HTTP

Exemple de requête XML-RPC

request

```
1  POST /RPC2 HTTP/1.0
2  User-Agent: mozilla
3  Host: localhost.localdomain
4  Content-Type: text/xml
5  Content-length: 186
6
7  <?xml version="1.0"?>
8    <methodCall>
9      <methodName>examples.getStateName</methodName>
10     <params>
11       <param> <value><i4>41</i4></value> </param>
12     </params>
13   </methodCall>
```

Exemple de réponse XML-RPC

answer

```
1  HTTP/1.1 200 OK
2  Connection: close
3  Content-Length: 176
4  Content-Type: text/xml
5  Date: Fri, 17 Jul 1998 19:55:08 GMT
6  Server: apache
7
8  <?xml version="1.0"?>
9    <methodResponse>
10     <params>
11       <param>
12          <value><string>South Dakota</string></value>
13       </param>
14    </params>
15   </methodResponse>
```

Évolution de SOAP (2)

La spécification actuelle : SOAP 1.1

- mai 2000
- `http://www.w3.org/TR/SOAP/`
- DevelopMentor, IBM, Lotus, Microsoft et UserLand
- Simple Object Access Protocol
- propose :
  - l'enveloppe SOAP (format général des messages)
  - un modèle de données relativement évolué (avec une traduction vers XML) :
    - types fondamentaux des schémas XML du W3
    - *structs*
    - listes
    - notions de pointeurs
  - un modèle RPC basique
  - un protocole basique : POST HTTP

Exemple de requête SOAP 1.1

request

```
1  POST /StockQuote HTTP/1.1
2  Host: www.stockquoteserver.com
3  Content-Type: text/xml; charset="utf-8"
4  Content-Length: nnnn
5  SOAPAction: "Some-URI"
6
7  <SOAP-ENV:Envelope
8    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
9    SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
10    <SOAP-ENV:Body>
11        <m:GetLastTradePrice xmlns:m="Some-URI">
12            <symbol>DIS</symbol>
13        </m:GetLastTradePrice>
14    </SOAP-ENV:Body>
15 </SOAP-ENV:Envelope>
```

Exemple de réponse SOAP 1.1

answer

```
1  HTTP/1.1 200 OK
2  Content-Type: text/xml; charset="utf-8"
3  Content-Length: nnnn
4
5  <SOAP-ENV:Envelope
6    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
7    SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
8     <SOAP-ENV:Body>
9         <m:GetLastTradePriceResponse xmlns:m="Some-URI">
10            <Price>34.5</Price>
11        </m:GetLastTradePriceResponse>
12    </SOAP-ENV:Body>
13 </SOAP-ENV:Envelope>
```

Évolution de SOAP (3)

La spécification en cours de développement : SOAP 1.2

- CR décembre 2002 (*a priori*, c'est la version finale)
- `http://www.w3.org/TR/soap12-part0/`
- contrôlée par le W3C
- traduction de l'acronyme SOAP abandonnée
- essentiellement un travail de modularisation et d'abstraction :
  - le format de l'enveloppe reste presque le même
  - le modèle de données est séparé de sa représentation en XML
  - les messages peuvent être échangés avec d'autres protocoles que le simple POST HTTP (mécanisme de *binding*)

Maturité

- XML-RPC est une technologie parfaitement mure, avec de très nombreuses implémentations (en particulier en terme de langages cibles)
- SOAP 1.1 est relativement mur :
  - bon support dans les serveurs d'applications
  - implémentation open source complète : AXIS du projet apache (`http://ws.apache.org/axis/`)
- SOAP 1.2 :
  - le W3C considère que la spécification est terminée
  - les implémentations sont en cours de réalisation
  - certaines implémentations sont relativement complètes

Dans ce cours : SOAP 1.1.

Doit-on savoir écrire du SOAP ?

- NON :
  - les API le font pour nous
  - aspect bas-niveau : on peut envoyer des paquets TCP/IP sans savoir comment ils sont structurés
  - c'est le rôle des implémenteurs
- OUI :
  - on comprend mieux la sémantique grâce à la syntaxe
  - il faut savoir lire les messages pour débugger une application
  - on comprend souvent mieux un service web avec un exemple d'interaction (traduction : WSDL est assez complexe)
- pré-requis : XML et les schémas du W3

Modèle de messages

- message SOAP : transmission d'informations d'un émetteur vers un récepteur (unidirectionnel)
- symétrie totale : si on utilise SOAP pour implémenter un mécanisme question/réponse, les messages ont le même format global
- notion de routage : un message peut passer d'un émetteur à son récepteur final en passant par des récepteurs intermédiaires qui peuvent modifier le message (même idée que pour SMTP)
- notion d'erreur : le format prévoit des messages spéciaux correspondant à la description d'une erreur

L'enveloppe SOAP

L'enveloppe SOAP contient :

- un en-tête (optionnel) :
  - mécanisme générique d'extension de SOAP
  - assez différent des en-têtes du mail
  - **ne** correspond **pas** en général à l'adresse du destinataire
- un corps (le message proprement dit, obligatoire) :
  - contenu structuré mais relativement arbitraire :
    - traduction du modèle de données en XML
    - basé sur les schémas du W3C
  - sauf pour un message d'erreur

Représentation XML

- espace de noms XML : `http://schemas.xmlsoap.org/soap/envelope/` (préfixe classique `env`)
- élément racine : `env:Envelope`
- en-tête : `env:Header`
- corps : `env:Body`
- **Rappel** : XML est *case-sensitive* (`envelope`≠`Envelope`)
- forme générale :

template.xml

```
1  <?xml version="1.0"?>
2  <env:Envelope
3    xmlns:env=="http://schemas.xmlsoap.org/soap/envelope/">
4    <env:Header>
5      <!-- en-tête -->
6    </env:Header>
7    <env:Body>
8      <!-- corps -->
9    </env:Body>
10 </env:Envelope>
```

Les espaces de noms

SOAP repose entièrement sur les espaces de noms XML :

- les éléments spécifiques à SOAP sont dans des NS spécifiques :
  - `http://schemas.xmlsoap.org/soap/envelope/` : “commandes” SOAP
  - `http://schemas.xmlsoap.org/soap/encoding/` : éléments définis pour la représentation des données
- les fils directs de `env:Header` **doivent** appartenir à des NS précisés dans le message
- les attributs de `env:Envelope` **doivent** appartenir à des NS
- les fils directs de `env:Body` **peuvent** appartenir à des NS précisés
- la norme SOAP 1.2 utilise des NS différents, mais conserve le même principe

L'en-tête

L'en-tête est constitué de plusieurs entrées dont le format n'est pas spécifié par la norme :

- le but de l'en-tête est de permettre la mise en place d'extensions aux mécanismes de base définis par SOAP :
  - WS-Security ajoute des entrées de signature
  - WS-RP (Routing Protocol) ajout des entrées pour le routage des messages SOAP
  - etc.
- chaque entrée doit avoir un NS associé
- la norme SOAP se contente de définir des attributs qui peuvent préciser le traitement des entrées de l'en-tête (dans le NS `env`)

Exemple WS-RP

exemple-rp.xml

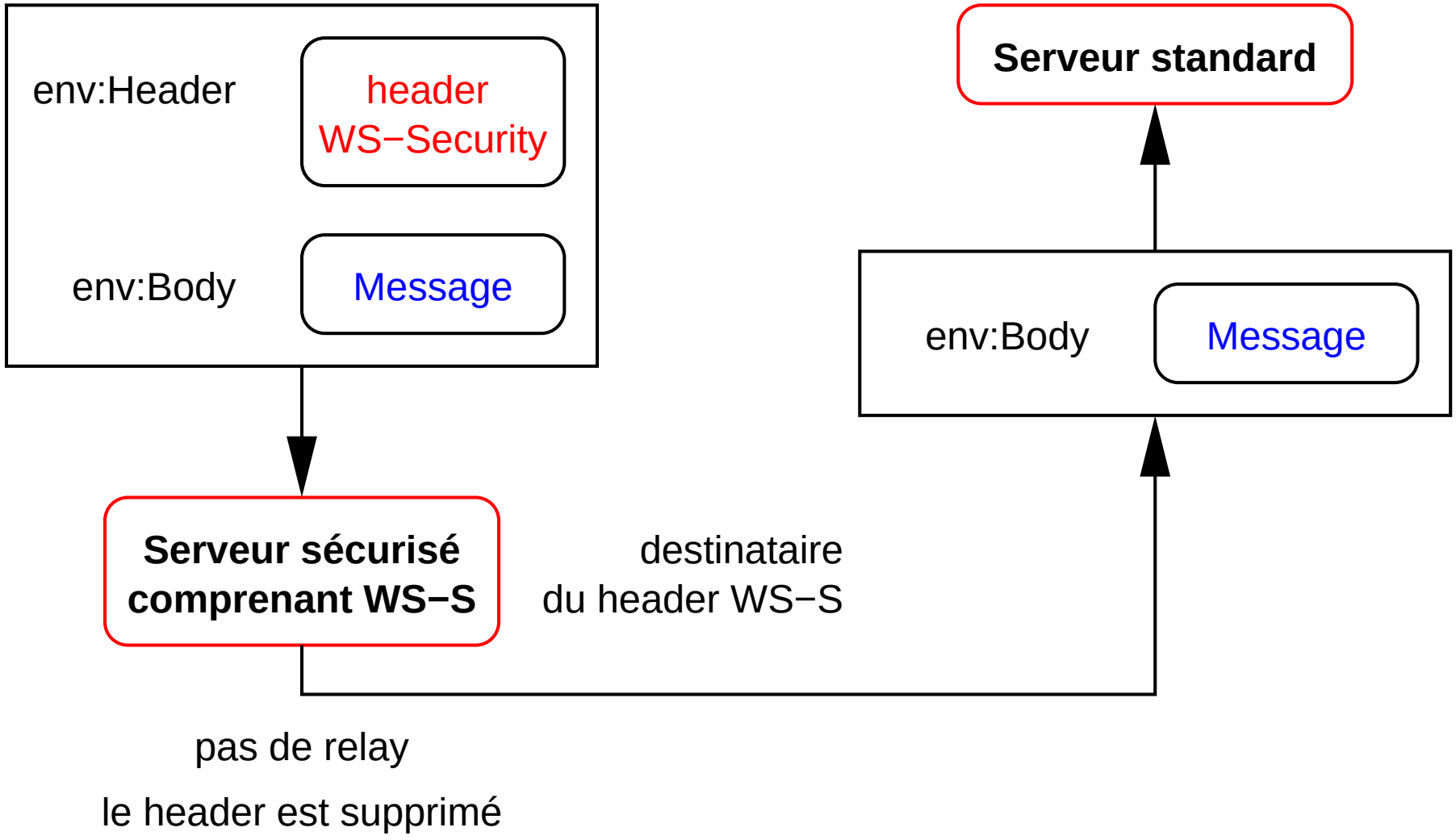
```
1  <env:Envelope
2     xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
3     <env:Header>
4        <m:path xmlns:m="http://schemas.xmlsoap.org/rp/">
5           <m:action>http://www.notification.org/update</m:action>
6           <m:to>soap://notification.com/some/endpoint</m:to>
7           <m:id>uuid:09233523-345b-4351-b623-5dsf35sgs5d6</m:id>
8        </m:path>
9     </env:Header>
10    <env:Body>
11       ...
12    </env:Body>
13 </env:Envelope>
```

Norme : http ://msdn.microsoft.com/library/default.asp ?url=/library/en-us/dnglobspec/html/ws-routing.asp

Attributs de l'en-tête

- éventuellement portés par les fils directs de `env:Header`
- `env:mustUnderstand` :
  - la valeur `1` indique que le récepteur du message doit comprendre l'entrée concernée ou terminer son traitement du message avec un statut d'erreur
  - les valeurs `1` et `0` sont remplacées respectivement par `true` et `false` dans SOAP 1.2
- `env:actor` :
  - permet de préciser la catégorie des récepteurs SOAP qui doivent étudier l'entrée portant l'attribut dans l'en-tête
  - si l'entrée est traitée, elle ne doit pas être propagée (en cas de routage)
  - mécanisme remplacé en 1.2 par deux attributs :
    - `env:role` pour les catégories
    - `env:relay` pour la propagation

Exemple


env:Header
header
WS−Security
env:Body
Message
Serveur standard
env:Body
Message
Serveur sécurisé
comprenant WS−S
destinataire
du header WS−S
pas de relay
le header est supprimé

Les messages d'erreur

Le corps d'un message d'erreur a un format particulier :

- la racine est `env:Fault`
- l'erreur est décrite grâce à quatre sous éléments (dans le NS `env`) :
  - `faultcode` : code d'erreur, présence obligatoire et contenu défini dans un NS
  - `faultstring` : traduction en langage naturel du code d'erreur (présence obligatoire)
  - `faultactor` : responsable de l'erreur (utilisé en cas de relais)
  - `detail` : pour donner des précisions, sous forme de sous-éléments
- le format a évolué de façon imporante en SOAP 1.2, mais les principes sont les mêmes

Exemple

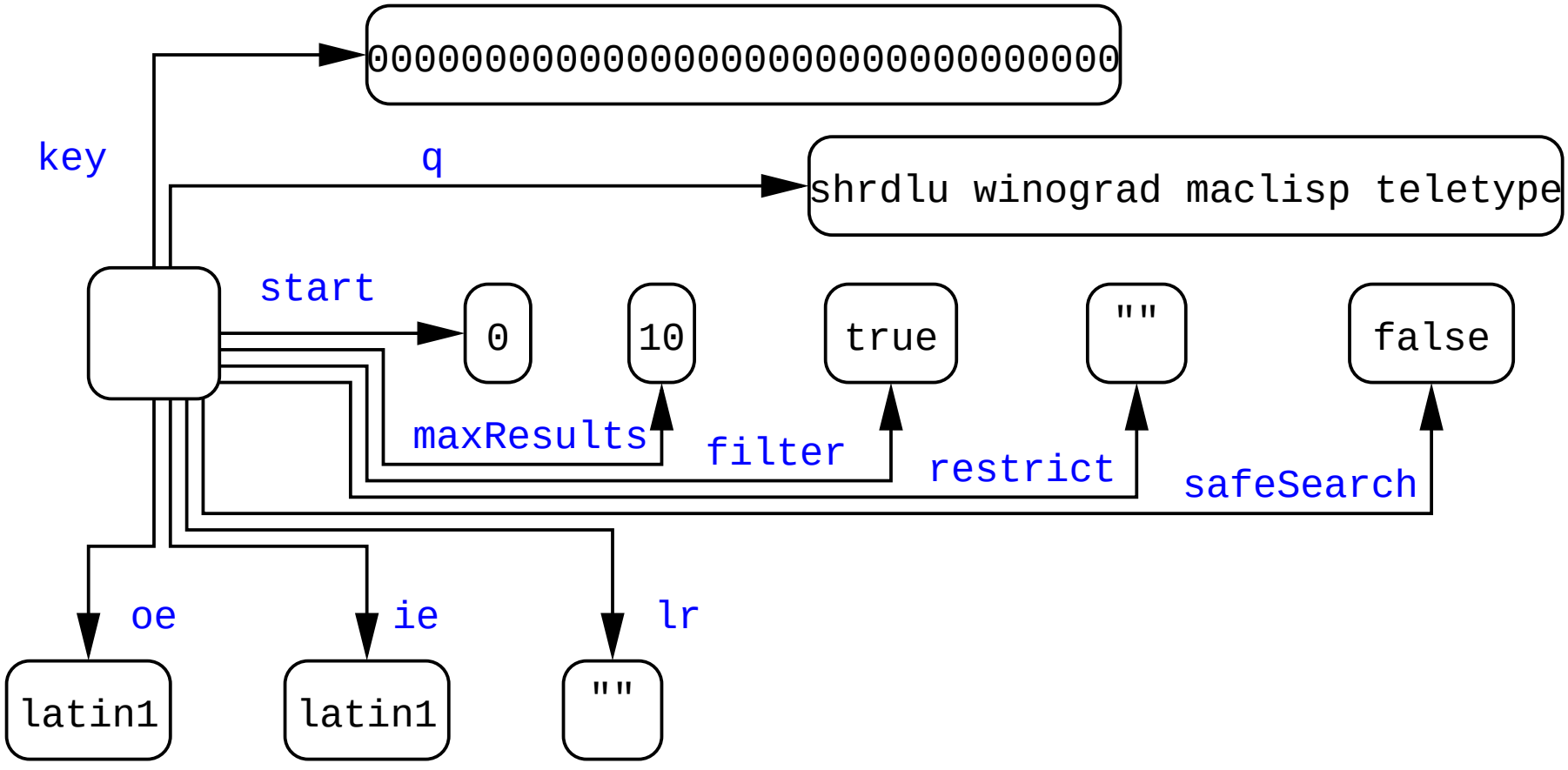
erreur.xml

```
1  <?xml version="1.0" encoding='ISO-8859-1'?>
2  <env:Envelope
3    xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
4    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
5    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
6    xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">
7    <env:Body>
8      <env:Fault>
9         <env:faultcode>env:Server</env:faultcode>
10        <env:faultstring>'xx' is not a valid integer value</env:faultstring>
11     </env:Fault>
12   </env:Body>
13 </env:Envelope>
```

Le modèle de données

- modèle abstrait : graphe orienté étiqueté (formalisation introduite dans 1.2)
- basé sur des types fondamentaux (les “feuilles” du graphe) et des constructeurs de type composé, *struct* et listes (les noeuds internes du graphe), associés à ces accesseurs (les arêtes du graphe)
- types fondamentaux :
  - ceux des schémas du W3C (`http://www.w3.org/TR/xmlschema-2/`)
  - entiers, réels, chaînes de caractères, binaire codé en base64, etc.
  - dates, durée, URI, etc.
- types composés :
  - énumération (une valeur parmi une liste)
  - *struct* (*compound*) : accès nommé
  - liste (*array*) : accès numéroté

Exemple d'une *struct*



Traduction en XML

SOAP définit une représentation en XML de son modèle de données (un *encoding*) :

- pour l'utiliser, il faut définir l'attribut `env:encodingStyle` et lui donner la valeur `http://schemas.xmlsoap.org/soap/encoding/` (noté `enc`)
- le modèle est facultatif : on peut utiliser une autre représentation, mais il faut que l'émetteur et le récepteur la comprennent
- la représentation est celle des schémas du W3C, avec des extensions pour la notion de type composé
- elle utilise les notions d'ID XML et de `uriReference` pour implémenter des références internes, ce qui permet de réduire les volumes de données

Exemple de traduction

googleSearch-part.xml

```
<ns1:doGoogleSearch
    xmlns:ns1="urn:GoogleSearch"
    xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/1999/XMLSchema"
    env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <key xsi:type="xsd:string">00000000000000000000000000000000</key>
  <q xsi:type="xsd:string">shrdlu winograd maclisp teletype</q>
  <start xsi:type="xsd:int">0</start>
  <maxResults xsi:type="xsd:int">10</maxResults>
  <filter xsi:type="xsd:boolean">true</filter>
  <restrict xsi:type="xsd:string"></restrict>
  <safeSearch xsi:type="xsd:boolean">false</safeSearch>
  <lr xsi:type="xsd:string"></lr>
  <ie xsi:type="xsd:string">latin1</ie>
  <oe xsi:type="xsd:string">latin1</oe>
</ns1:doGoogleSearch>
```

Traduction XML (2)

Principes de base :

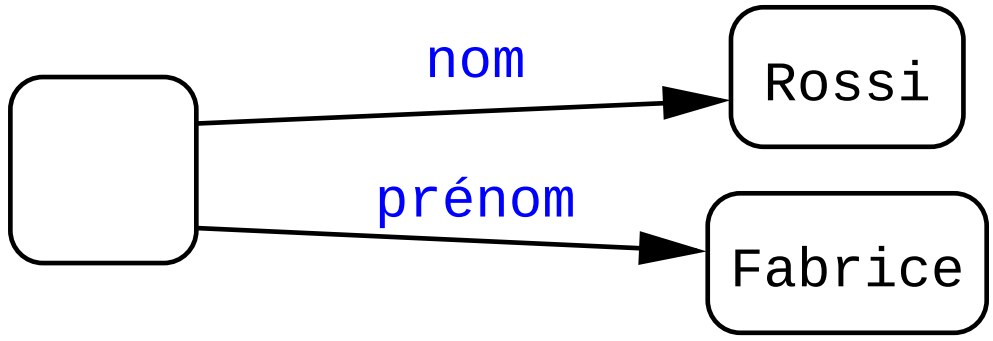
- toute valeur est représentée comme le contenu d'un élément
- quand la valeur correspond à un type de base, celui-ci est précisé directement ou indirectement par l'élément :
  - directement par un attribut `type` du NS des schémas (`http://www.w3.org/1999/XMLSchema-instance`, noté `xsi`) dont la valeur correspond à un type fondamental des schémas (dans le NS `http://www.w3.org/1999/XMLSchema`, noté `xsd`)
  - directement par le nom de l'élément (choisit dans le NS `enc`)
  - indirectement par le type du tableau (le cas échéant)
- exemple : `maxResults` de type `xsd:int` dans le transparent précédent

Traduction des *structs*

Une *struct* est une valeur composée :

- un nœud du graphe sans étiquette dont partent des arcs étiquetés vers d'autres valeurs
- les étiquettes des arcs correspondent aux champs de la structure
- traduction XML :
  - un élément (sans type associé)
  - dont le contenu est constitué d'éléments nommés selon les champs de la structure
- correspond à la représentation classique des données structurées en XML
- dans SOAP 1.1, on peut avoir un type composé plus général que les *structs* dans lequel plusieurs champs peuvent porter le même nom : cette possibilité a été supprimée dans SOAP 1.2

Exemple



personne-part.xml

```
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <pers:personne
3      xmlns:pers="http://apiacoa.org/teaching/xml/personnes"
4      xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
5      xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
6      xmlns:xsd="http://www.w3.org/1999/XMLSchema"
7      env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
8    <nom xsi:type="xsd:string">Rossi</nom>
9    <prénom xsi:type="xsd:string">Fabrice</prénom>
10 </pers:personne>
```

Traduction des listes

Les tableaux de SOAP (*array*) sont en fait des listes (le type n'a pas besoin d'être uniforme) :

- correspond à un nœud du graphe sans étiquette dont partent des arcs numérotés vers d'autres valeurs
- traduction XML :
  - un élément de type dérivé de `enc:Array` (ou directement l'élément `enc:Array`)
  - portant un attribut `enc:arrayType` qui précise le type des éléments et la taille du tableau (sous la forme `[n]`)
  - le type `xsd:ur-type` peut servir de joker pour avoir des types différents pour chaque élément
  - des sous-éléments dans l'ordre du tableau, avec si besoin le type pour chaque élément

Traduction des listes (2)

- on peut représenter des tableaux à plusieurs dimensions (modèle ligne par ligne)
- on peut représenter des tableaux creux ou partiellement transmis
- la traduction des tableaux a été modifiée dans SOAP 1.2 (surtout au niveau syntaxique) :
  - tableaux creux et partiels supprimés
  - les attributs `itemType` et `arraySize` remplacent l'attribut `arrayType`

Exemple

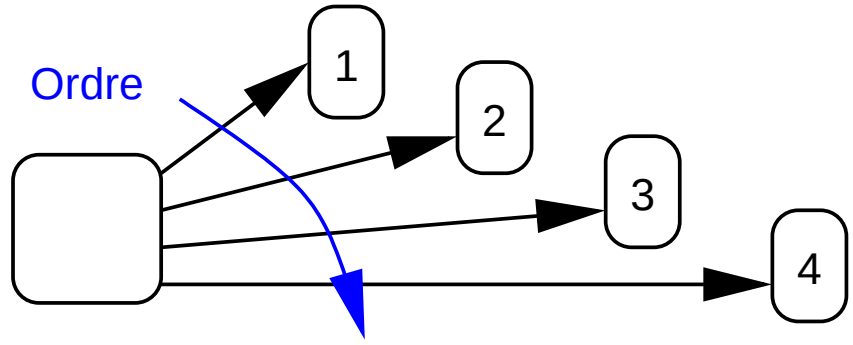


tableau-part1.xml

```
1  <enc:Array
2      xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
3      xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
4      xmlns:xsd="http://www.w3.org/1999/XMLSchema"
5      env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
6      xmlns:enc="http://schemas.xmlsoap.org/soap/encoding/"
7      enc:arrayType="xsd:int[4]">
8    <x>1</x>
9    <y>2</y> <!-- les noms n'importent pas -->
10   <x>3</x>
11   <x>4</x>
12 </enc:Array>
```

Exemple (2)

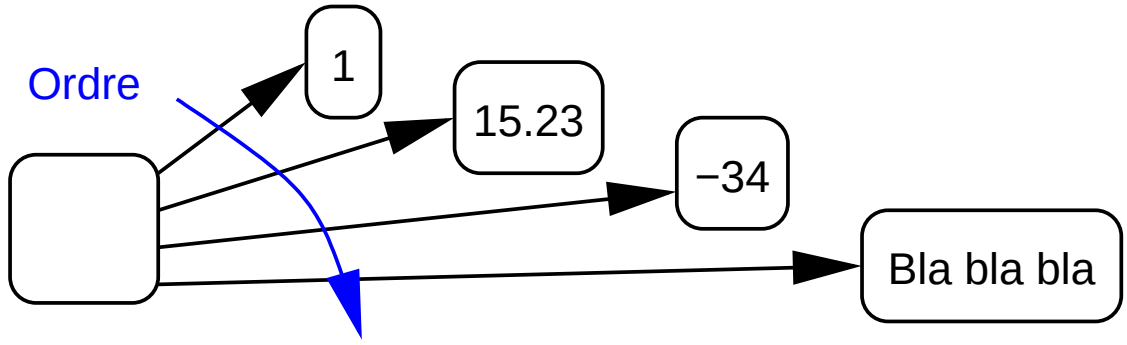


tableau-part2.xml

```
1  <enc:Array
2      xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
3      xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
4      xmlns:xsd="http://www.w3.org/1999/XMLSchema"
5      env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
6      xmlns:enc="http://schemas.xmlsoap.org/soap/encoding/"
7      enc:arrayType="xsd:ur-type[4]">
8    <x xsi:type="xsd:int">1</x>
9    <x xsi:type="xsd:decimal">15.23</x>
10   <x xsi:type="xsd:int">-34</x>
11   <x xsi:type="xsd:string">Bla bla bla</x>
12 </enc:Array>
```

Références croisées

Afin de limiter le volume des messages, SOAP utilise le mécanisme de références croisées d'XML :

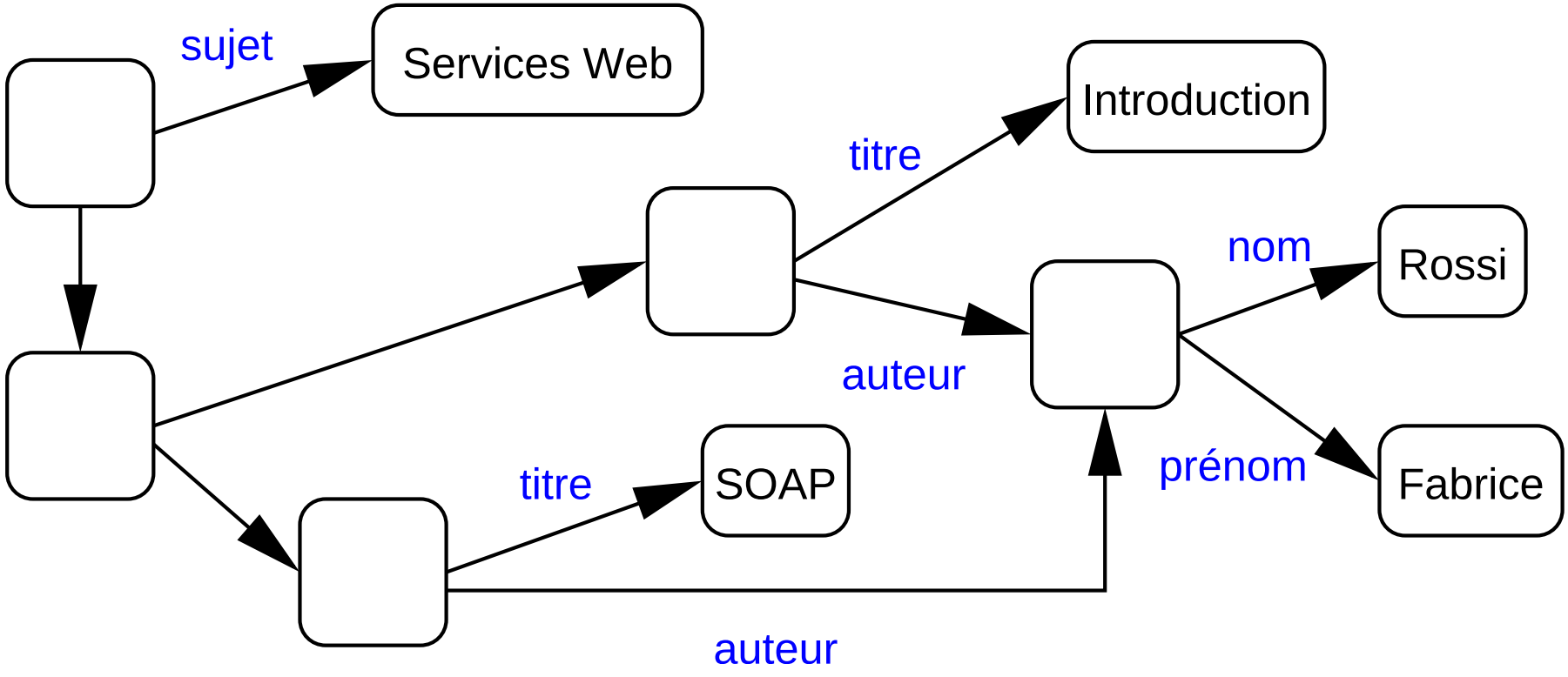
- un élément peut être identifié par un attribut de type ID et de nom `id`
- un élément peut être vide mais doit alors posséder un attribut `href` de type `xsi:uri-reference`
- l'interprétation d'un élément vide est son simple remplacement par l'élément référencé par son `href`
- en SOAP 1.2, `href` est remplacé par `enc:ref` de type IDREF

Exemple

reference-part.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<c:cours xmlns:c="http://apiacoa.org/teaching/xml/cours"
    xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/1999/XMLSchema"
    env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    xmlns:enc="http://schemas.xmlsoap.org/soap/encoding/"
  <c:sujet xsi:type="xsd:string">Services Web</c:sujet>
  <enc:Array enc:arrayType="c:slides[2]">
    <c:slides><c:titre xsi:type="xsd:string">Introduction</c:titre>
      <c:auteur id="Rossi">
        <nom xsi:type="xsd:string">Rossi</nom>
        <prénom xsi:type="xsd:string">Fabrice</prénom>
      </c:auteur>
    </c:slides>
    <c:slides><c:titre xsi:type="xsd:string">SOAP</c:titre>
      <c:auteur href="#Rossi"/>
    </c:slides>
  </enc:Array>
</c:cours>
```

Exemple


sujet
Services Web
titre
Introduction
nom
Rossi
auteur
titre
SOAP
prénom
Fabrice
auteur

SOAP et HTTP

- l'envoi d'un message SOAP correspond à une requête HTTP POST
- en cas d'utilisation requête/réponse, le message SOAP de réponse est dans la réponse HTTP
- en cas d'erreur, la réponse doit être un code 500 (erreur interne) accompagné d'un message SOAP d'erreur
- en cas de succès, la réponse utilise un code 2xx (en général 200 pour indiquer un succès total)
- SOAP utilise un en-tête HTTP spécifique (`SOAPAction`) constitué d'un URI qui précise le but du message (par exemple l'opération appelée dans le cas d'une utilisation RPC)
- HTTP 1.0 ou 1.1
- type du contenu `text/xml`
- quelques évolutions dans SOAP 1.2 (`SOAPAction` supprimé, etc.)

Exemple : POST

request-1.1-2

```
1  POST /cgi-bin/tempconverter.exe/soap/ITempConverter HTTP/1.0
2  Content-Type: text/xml; charset=utf-8
3  Accept: application/soap+xml, application/dime, multipart/related, text/*
4  User-Agent: Axis/1.1RC2
5  Host: developerdays.com
6  Cache-Control: no-cache
7  Pragma: no-cache
8  SOAPAction: "urn:TempConverterIntf-ITempConverter#CtoF"
9  Content-Length: 447
10
11 <?xml version="1.0" encoding="UTF-8"?>
12 <soapenv:Envelope ...>
13 ...
14 </soapenv:Envelope>
```

Exemple : réponse (200)

answer-1.1-2

```
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Date: Mon, 31 Mar 2003 21:05:48 GMT
Content-Type: text/xml
Content-Length: 621
Content:

<?xml version="1.0" encoding='UTF-8'?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
                   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                   xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">
   <SOAP-ENV:Body>
      <NS1:CtoFResponse xmlns:NS1="urn:TempConverterIntf-ITempConverter"
          SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <return xsi:type="xsd:int">86</return>
      </NS1:CtoFResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Le modèle RPC

Un appel de fonction (une opération dans la terminologie services web) est décrit par un message SOAP particulier :

- l'appel est représenté par une *struct* :
  - le nom de la structure est celui de la fonction appelée
  - chaque paramètre de l'appel est considéré comme un champs de la structure
- le résultat est représenté par une *struct*
- le modèle de fonction autorise 3 modes de passage pour les paramètres :
  - mode `in` : paramètre utilisé mais non modifié, apparaît seulement dans l'appel
  - mode `in/out` : paramètre utilisé et modifié, apparaît dans l'appel et la réponse
  - mode `out` : paramètre de retour uniquement, appraît dans la réponse seulement

Exemple

Appel de la fonction `CtoF` (Celsius vers Fahrenheit) :

call.xml

```
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
3                xmlns:xsd="http://www.w3.org/2001/XMLSchema"
4                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
5    <env:Body>
6      <ns1:CtoF env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
7                xmlns:ns1="urn:TempConverterIntf-ITempConverter">
8        <temp xsi:type="xsd:int">30</temp>
9      </ns1:CtoF>
10   </env:Body>
11 </env:Envelope>
```

Un seul paramètre, `temp`, de type `xsd:int`

Résultat de l'appel

return.xml

```
1  <?xml version="1.0" encoding='UTF-8'?>
2  <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
3                     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
4                     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5                     xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">
6     <SOAP-ENV:Body>
7        <NS1:CtoFResponse xmlns:NS1="urn:TempConverterIntf-ITempConverter"
8            SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
9           <return xsi:type="xsd:int">86</return>
10       </NS1:CtoFResponse>
11    </SOAP-ENV:Body>
12 </SOAP-ENV:Envelope>
```

Remarques :

- convention classique pour le nom de la *struct* : nom de la fonction suivi de `Response`
- convention classique pour le nom de valeur renvoyée par la méthode : `return` (obligatoire en SOAP 1.2)

Solutions concurrentes

SOAP n'est pas le seul moyen de faire des RPC (!) :

- RPC classiques (ONC et DCE) :
  - protocole spécifique
  - traduction en binaire (XDR)
- CORBA :
  - protocole spécifique (IIOP)
  - traduction en binaire
- critique de SOAP :
  - problèmes :
    - volume énorme (10 fois plus gros et donc plus lent que du binaire)
    - protocole HTTP pas vraiment adapté (déconnecté)
  - avantages :
    - inter-opérabilité UNIX/Windows (en théorie !)
    - souple (*binding* SMTP, extensibilité, etc.)

Avenir

- Extensions directes de SOAP :
  - *binding* email : défini par une note W3
    (`http://www.w3.org/TR/2002/NOTE-soap12-email-2002062`
  - pièces jointes : SOAP *attachments* :
    - travail en cours
      `http://www.w3.org/TR/2002/WD-soap12-af-20020814/`
    - évolution des pièces jointes pour SOAP 1.1
      (`http://www.w3.org/TR/2000/NOTE-SOAP-attachments-2`
      elles mêmes basées sur MIME
- Régler les problèmes d'inter-opérabilité :
  - précision des réels
  - précision des dates et durées
  - entiers positifs
  - etc.