**Intégration**

**Utilisant**

**La technologie**

**Web Service**

Sommaire

[1- Introduction 3](#_Toc382808649)

[2- Pourquoi Web Service ? 4](#_Toc382808650)

[3- Les caractéristiques de Web Service: 5](#_Toc382808651)

[4- Web Service Architecture de base: 7](#_Toc382808652)

[5- Web Service Components 10](#_Toc382808653)

[- SOAP 11](#_Toc382808654)

[- UDDI 11](#_Toc382808655)

[- WSDL 12](#_Toc382808656)

[- REST: 14](#_Toc382808657)

[6- Web Service Security: 17](#_Toc382808658)

[7 - Integration and Interopérabilité: 19](#_Toc382808659)

[8- Interopérabilité avec Moodle: 20](#_Toc382808660)

[Annexe 1 – Exemple XML 26](#_Toc382808661)

[Annexe 2 – Exemple SOAP 27](#_Toc382808662)

[SOAP - Requête 27](#_Toc382808663)

[SOAP – Réponse 28](#_Toc382808664)

[Annexe 3 – Exemple WSDL 29](#_Toc382808665)

# Introduction

Les services Web sont standards et ouverts (XML, SOAP, HTTP, etc).

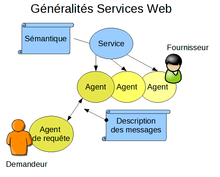
Les services Web sont basés sur les applications Web qui interagissent avec d'autres applications Web dans le but d'échanger des données

Les services Web peuvent convertir vos applications existantes en applications Web.

Dans ce tutoriel, vous apprendrez ce que les services Web sont exactement et pourquoi et comment les utiliser.

Un **service web** (ou **service de la toile**[1](http://fr.wikipedia.org/wiki/Service_web#cite_note-1)) est un [programme informatique](http://fr.wikipedia.org/wiki/Programme_informatique) de la famille des [technologies web](http://fr.wikipedia.org/wiki/Technologies_Web) permettant la communication et l'échange de données entre applications et systèmes hétérogènes dans des environnements distribués. Il s'agit donc d'un ensemble de fonctionnalités exposées sur [internet](http://fr.wikipedia.org/wiki/Internet) ou sur un [intranet](http://fr.wikipedia.org/wiki/Intranet), par et pour des applications ou machines, sans intervention humaine, de manière synchrone ou asynchrone. Le protocole de communication est défini dans le cadre de la norme [SOAP](http://fr.wikipedia.org/wiki/SOAP) dans la [signature](http://fr.wikipedia.org/wiki/Signature_num%C3%A9rique) du service exposé ([WSDL](http://fr.wikipedia.org/wiki/Web_Services_Description_Language)). Actuellement, le protocole de transport est essentiellement [HTTP](http://fr.wikipedia.org/wiki/Hypertext_Transfer_Protocol)(S).

Le concept a été précisé et mis en œuvre dans le cadre de *Web Services Activity*[2](http://fr.wikipedia.org/wiki/Service_web#cite_note-2), au [W3C](http://fr.wikipedia.org/wiki/W3C), particulièrement avec le protocole SOAP. Associé avec les [Échanges de Données Informatisés (EDI)](http://fr.wikipedia.org/wiki/%C3%89change_de_donn%C3%A9es_informatis%C3%A9), le consortium [ebXML](http://fr.wikipedia.org/wiki/Electronic_Business_using_XML) l'a utilisé pour automatiser des échanges entre entreprises. Cependant le concept s'enrichit avec l'approfondissement des notions de ressource et d'état, dans le cadre du modèle [REST](http://fr.wikipedia.org/wiki/Representational_State_Transfer), et l'approfondissement de la notion de service, avec le modèle [SOA](http://fr.wikipedia.org/wiki/Architecture_orient%C3%A9e_services).

[](http://commons.wikimedia.org/wiki/File:Whatisaserviceweb.png?uselang=fr)

Très grandes généralités sur un service web.

Dans sa présentation la plus générale, un service web se concrétise par un agent, réalisé selon une technologie informatique précise, par un fournisseur du service. Un demandeur, à l'aide d'un agent de requête, utilise ce service. Fournisseur et demandeur partagent une même sémantique du service web, tandis qu'agent et agent de requête partagent une même description du service pour coordonner les messages qu'ils échangent[3](http://fr.wikipedia.org/wiki/Service_web#cite_note-3).

Il existe plusieurs technologies derrière le terme services web :

* Les services web de type [*Representational state transfer*](http://fr.wikipedia.org/wiki/Representational_state_transfer) (REST) exposent entièrement ces fonctionnalités comme un ensemble de ressources ([URI](http://fr.wikipedia.org/wiki/URI)) identifiables et accessibles par la syntaxe et la sémantique du protocole [HTTP](http://fr.wikipedia.org/wiki/HTTP). Les Services Web de type [REST](http://fr.wikipedia.org/wiki/Representational_state_transfer) sont donc basés sur l'architecture du [web](http://fr.wikipedia.org/wiki/Web) et ses standards de base : [HTTP](http://fr.wikipedia.org/wiki/HTTP) et [URI](http://fr.wikipedia.org/wiki/URI) ;
* Les *Services Web* [*WS-\**](http://fr.wikipedia.org/wiki/Liste_des_sp%C3%A9cifications_des_Services_Web_WS-*) exposent ces mêmes fonctionnalités sous la forme de services exécutables à distance. Leurs spécifications reposent sur les standards SOAP et [WSDL](http://fr.wikipedia.org/wiki/Web_Services_Description_Language) pour transformer les problématiques d'intégration héritées du monde [Middleware](http://fr.wikipedia.org/wiki/Middleware) en objectif d'interopérabilité.

Les standards [WS-\*](http://fr.wikipedia.org/wiki/Liste_des_sp%C3%A9cifications_des_Services_Web_WS-*) sont souvent décriés, comme risquant de générer une course à la performance technologique[4](http://fr.wikipedia.org/wiki/Service_web#cite_note-4). Toutefois leur robustesse dans le milieu des services entre professionnels est reconnue, et ils restent largement utilisés. Aussi l'on préfère les faire évoluer[5](http://fr.wikipedia.org/wiki/Service_web#cite_note-5).

**Cas d’utilisation:**

* Vous avez écrit un programme qui génère des devis. Vos citations sont en dollars américains. Votre client est en Allemagne. Vous pouvez appeler un programme qui est situé sur l'Internet quelque part pour obtenir le taux de change actuel de l'euro.
* Vous acceptez les cartes de crédit pour le paiement. Après que votre client tape son numéro de carte de crédit dans votre application, vous appelez un programme sur l'ordinateur de votre banque pour obtenir l'achat approuvée instantanément.
* Vous avez accepté une commande d'un client, et que vous voulez embarquer les marchandises via UPS. Vous pouvez appeler un programme en cours d'exécution sur le système informatique de UPS et qu'elle calcule le coût de l'expédition pendant que vous attendez.
* Plus tard, vous pouvez suivre la même expédition par appel d'un programme de suivi sur le système de UPS. Vous pouvez avoir des informations à la minute sur l'endroit où le paquet est.

# Pourquoi Web Service ?

Voici les avantages de l'utilisation de Web Service ?

• **Exposer la fonction existante sur le réseau :**

Un Web Service est une unité de code managé qui peut être invoqué à distance via le protocole HTTP, qui est activé à l'aide de requêtes HTTP. Ainsi, le Web Service vous permet d'exposer les fonctionnalités de votre code existant sur ​​le réseau. Une fois qu'il est exposé sur le réseau, une autre application peut utiliser la fonctionnalité de votre programme.

• **Connexion d'applications différentes - l'interopérabilité :**

Le Web Service permet aux différentes applications de communiquer entre eux et de partager des données et des services entre eux. D'autres applications peuvent également utiliser les services de Web Service. Par exemple VB ou. NET peut parler à Java Web Service et vice versa. Ainsi, Web Service est utilisé pour faire la plate-forme d'applications et la technologie indépendante.

• **Protocole standardisé:**

Web Service utilise le protocole normalisé standard de l'industrie pour la communication. Tous les quatre couches (Service de transport, de messagerie XML, Description des services et des couches de service de découverte ) utilisent le protocole bien défini dans la pile de protocoles de Web service. Cette normalisation de la pile de protocole donne de nombreux avantages commerciaux comme large éventail de choix, la réduction du coût en raison de la concurrence et l'augmentation de la qualité.

• **Faible coût de communication:**

Web Service utilise SOAP sur le protocole HTTP pour la communication, de sorte que vous pouvez utiliser votre Internet à faible coût existante pour mettre en œuvre le Web Service. Cette solution est beaucoup moins coûteuse par rapport aux solutions propriétaires. En plus, le Web Service peut également être mis en œuvre sur d'autres mécanismes de transport fiables tels que FTP , etc

* Les services Web fournissent l'[interopérabilité](http://fr.wikipedia.org/wiki/Interop%C3%A9rabilit%C3%A9_informatique) entre divers [logiciels](http://fr.wikipedia.org/wiki/Logiciel) fonctionnant sur diverses [plates-formes](http://fr.wikipedia.org/wiki/Plate-forme_%28informatique%29).
* Les services Web utilisent des [standards et protocoles ouverts](http://fr.wikipedia.org/wiki/Format_ouvert).
* Les protocoles et les [formats de données](http://fr.wikipedia.org/wiki/Formats_de_donn%C3%A9es) sont au format texte dans la mesure du possible, facilitant ainsi la compréhension du fonctionnement global des échanges.
* Les [outils de développement](http://fr.wikipedia.org/wiki/Environnement_de_d%C3%A9veloppement_int%C3%A9gr%C3%A9), s'appuyant sur ces standards, permettent la création automatique de programmes utilisant les services Web existants.
* Interopérabilité – Web Service permet l'échange de données entre les différentes applications et plates-formes.
* Web Service peut publier des fonctions et des messages pour le reste du monde.
* Web Service est un composant applicatif réutilisable.

# Les caractéristiques de Web Service:

Web Service a des caractéristiques particulières :

**• XML**

Web Service utilise XML pour la représentation des données et des couches de transport de données. L’utilisation de XML élimine tout réseau, système d'exploitation, ou liaison plate-forme Donc, Web Services sont des applications hautement interopérables.

**• Système faiblement couplé**

Un consommateur de Web Service n'est pas lié directement au Web Service. L'interface de Web Service peut changer au fil du temps, sans compromettre la capacité du client d'interagir avec le service. Un système étroitement couplé implique que le client et la logique du serveur sont étroitement liés les uns aux autres, ce qui implique que si une interface change, l'autre doit également être mis à jour. L'adaptation d'une architecture faiblement couplée tend à rendre les systèmes logiciels plus maniables et permet une intégration simple entre les différents systèmes.

**• Capacité d’être synchrone ou asynchrone**

La synchronisation se réfère à la liaison du client pour exécuter le service. Dans les appels synchrones, les clients sont bloquée et en attente pour que le service complète son fonctionnement avant de poursuivre. Les opérations asynchrones permettent à un client d’appeler un service, puis d'exécuter d'autres fonctions. Les clients asynchrones récupèrent leur résultat à un moment plus tard, alors que les clients synchrones reçoivent leur résultat lorsque le service est terminé. La capacité asynchrone est un facteur clé dans les systèmes faiblement couplés.

**• Prise en charge des appels de procédure à distance (RPC)**

Les Web Service permettent aux clients d'invoquer des procédures des fonctions et des méthodes sur des objets distants en utilisant un protocole basé sur XML. Les procédures distants exposent les paramètres d'entrée et de sortie que le Web Service doit prendre en charge. Le développement de composants par Enterprise JavaBeans (EJB) et . Composants NET a de plus en plus une partie d'architectures et de déploiements d'entreprise au cours des deux dernières années. Ces deux technologies sont distribués et accessibles par le biais de divers mécanismes RPC. Le Web Service prend en charge RPC en fournissant des services équivalentes à celles d'un composant traditionnel, ou en traduisant les appels entrants dans une invocation d'un EJB ou d'un composant.NET.

**• L'échange de documents Supports**

Un des principaux avantages de XML est sa manière générique de représenter non seulement des données, mais aussi des documents complexes. Ces documents peuvent être simples, comme lors de la représentation d'une adresse actuelle, ou complexe, représentant un livre entier ou RFQ. Web Services prennent en charge l'échange transparent de documents pour faciliter l'intégration de l'entreprise.

# Web Service Architecture de base:

Il y a deux façons de voir l'architecture de Web service :

• La première consiste à examiner les rôles individuels de chaque acteur de Web service (illustré dans la figure 1).

• La deuxième est d'examiner la pile de protocoles de Web Services.

**Rôles de Web Service**

Il existe trois grands rôles au sein de l'architecture de Web Services:

* **Le fournisseur de services :**

C'est le fournisseur de Web service. Le fournisseur de service met en œuvre le service et le met à disposition sur Internet.

* **Le demandeur de services**

Il s'agit d’un consommateur de Web service. Le demandeur utilise un Web Service existant en ouvrant une connexion de réseau et en envoyant une requête XML.

* **Le registre de service**

Il s'agit d'un répertoire logique centralisé pour les services. Le registre fournit une place centrale où les développeurs peuvent publier de nouveaux services ou de trouver ceux qui existent déjà. Il sert donc comme une chambre de compensation centralisée pour les entreprises et leurs services.

**Web Service pile de protocole**

Une deuxième option pour visualiser l'architecture de services Web est d'examiner la pile de protocoles de services Web. La pile est encore en évolution, mais dispose actuellement de quatre couches principales :

* **Transport de services**

Cette couche est responsable du transport de messages entre les applications. Actuellement, cette couche comprend le protocole de transfert hypertexte (HTTP) , protocole de transfert de courrier (SMTP) , protocole de transfert de fichiers (FTP) , et des protocoles plus récents, tels que les blocs Extensible Exchange Protocol ( BIP ) .

* **Messagerie XML**

Cette couche est responsable du codage des messages en format XML commun de telle sorte que les messages peuvent être compris à chaque extrémité. Actuellement, cette couche comprend XML-RPC et SOAP.

* **Description du service**

Cette couche est responsable de décrire l'interface publique à un site web spécifique. Actuellement, la description de service est gérée par le « Web Service Description Langage » (WSDL).

* **Découverte de services**

Cette couche est chargée de centraliser les services dans un registre commun, et de fournir une fonctionnalité facile de publier et de trouver. Actuellement, la découverte de services est géré par « Universal Description, Discovery and Integration » (UDDI).

Comme les services du web évoluent, des couches supplémentaires peuvent être ajoutées, et des technologies supplémentaires peuvent être ajoutées à chaque couche.

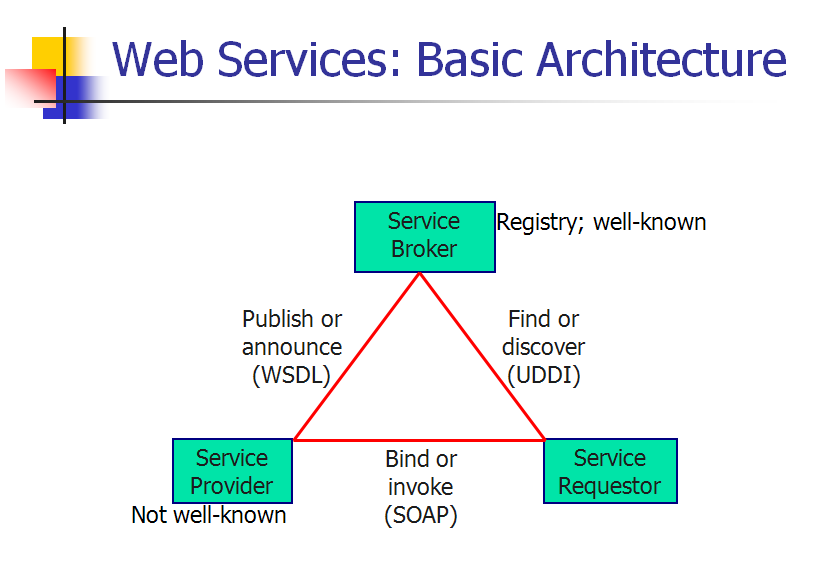


Figure 1

# Web Service Components

Au cours des dernières années, trois technologies primaires ont émergé comme des normes mondiales qui forment le noyau de la technologie des Web services d'aujourd'hui. Les standards du Web Service sont illustrés dans la figure 3.

Ces technologies sont:

**- XML:**

XML est un langage de balise extensible. (Référer à l'annexe 1 pour l’exemple de XML).

* L'objectif principal est d'aider les ordinateurs à partager les données. XML est un méta-langage utilisé pour créer des langages. XML est un concept vaste.
* Chaque document XML est à base de text:

- Le partage de données entre les différents ordinateurs.

- Le partage de données dans Internet.

- L'indépendance de plate-forme.

* Les avantages de XML sont :
* XML est un partage facile des données.
* Des documents de texte sont lisibles entre les périphériques.
* Les documents peuvent être modifiés avec un éditeur de texte.
* XML est facile à manipuler via des langages de programmation.

## - SOAP

SOAP est un protocole basé sur XML pour l'échange d'informations entre les ordinateurs. (Référer à l’Annexe 2 pour l’exemple de SOAP: Requête - Réponse)

* SOAP est un protocole de communication
* SOAP est un protocole simple d'accès aux objets
* SOAP permet la communication entre les applications
* SOAP est un format pour l'envoi de messages
* SOAP est conçu pour communiquer par l'intermédiaire d'Internet
* SOAP est indépendant de la plateforme
* SOAP est un langage indépendant
* SOAP est simple et extensible
* SOAP permet d’utiliser Firewall
* SOAP sera développé comme un standard du W3C
* SOAP est basé sur XML
* SOAP n'est pas lié à un protocole de transport spécifique. En fait, vous pouvez utiliser SOAP via HTTP, SMTP ou FTP. Une idée prometteuse est donc d'utiliser SOAP sur BEEP.

## - UDDI

UDDI est un standard basé sur XML pour décrire, éditer, et trouver les Web Services.

* UDDI signifie Universal Description, Discovery and Integration.
* UDDI est une spécification pour un registre distribué de Web Services.
* UDDI est une plateforme cadre indépendant, ouvert.
* UDDI peut communiquer via SOAP, CORBA, Java RMI Protocole.
* UDDI utilise WSDL pour décrire les interfaces de services Web.
* UDDI est vu avec SOAP et WSDL comme l'une des trois normes de base de services Web.
* UDDI est une initiative de l'industrie ouverte permettant aux entreprises de se découvrir et de définir comment ils interagissent sur ​​Internet.

## - WSDL

WSDL est un langage basé sur XML pour décrire les Web Services et comment y accéder. Le modèle de donnée est illustré dans la figure 2.

(Référer à l’Annexe 3 pour l’exemple de WSDL)

* WSDL signifie « Web Services Description Language».
* WSDL est un protocole basé sur XML pour l'échange d'informations dans des environnements décentralisés et distribués.
* WSDL est le format standard pour la description du Web Service.
* WSDL décrit comment accéder à un service Web ainsi que les opérations de son rendement.
* WSDL est un langage pour décrire comment faire l’interface avec les services basés sur XML.
* WSDL est une partie intégrante de UDDI, un registre des entreprises dans le monde entier basé sur XML.
* WSDL est le langage utilisé par UDDI.
* WSDL a été développé conjointement par Microsoft et IBM.

**WSDL – Data Model**

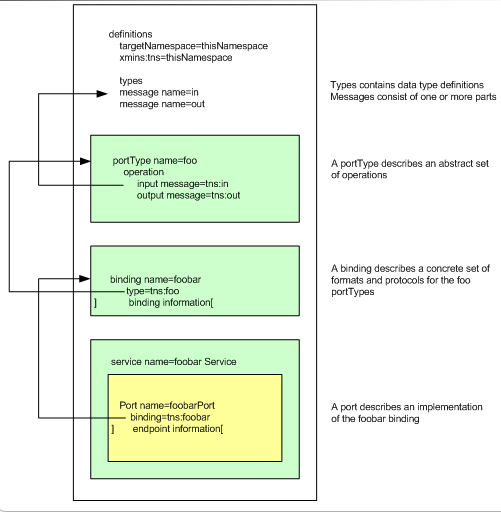


Figure 2

## - REST:

* REST signifie “Representational State Transfer “.
* REST est un style architectural pour les systèmes en réseau qui limite la sémantique des connecteurs (et non les composants sémantiques)
* Grand impact; SOAP presque déplacées et WDSL

**Les services Web RESTFUL :**

• URL unique est la représentation d'un objet

• Un client sélectionnant un lien est une transition d'état, dans la réception de la page suivante (état suivant) de l'application.

• RESTFUL utilise HTTP pour les opérations CRUD (Create / Read / Update / Delete).

**Les Caractéristiques du REST :**

* Le REST est un Client - Serveur.
* Le REST n’a pas d’identité - les demandes ne peuvent pas profiter des contextes stockées sur un serveur
* La mise en cache - réponses peuvent être étiquetés
* Interface uniforme - URI, hypermédia
* Composants stratifiés
* Les clients peuvent être étendus par le code-sur-demande

**REST contre SOAP**

Le REST est léger. Il est lisible par un humain, facile à construire.

Il permet un accès nommée ressources par le biais d'une interface unique et cohérente. Il permet un format de données différent. Exemple: getusr (ID)

Le SOAP est facile à consommer parfois, il est rigide à un contrat.

Le SOAP expose des opérations, il permet l’utilisation de XML. Exemple: effectuer des tâches (ID, tâche)

L'idée principale:

REST - Expose une ressource (objet) via GET, POST, etc ...

SOAP: expose des opérations nommées pour exécuter la logique des entreprises.

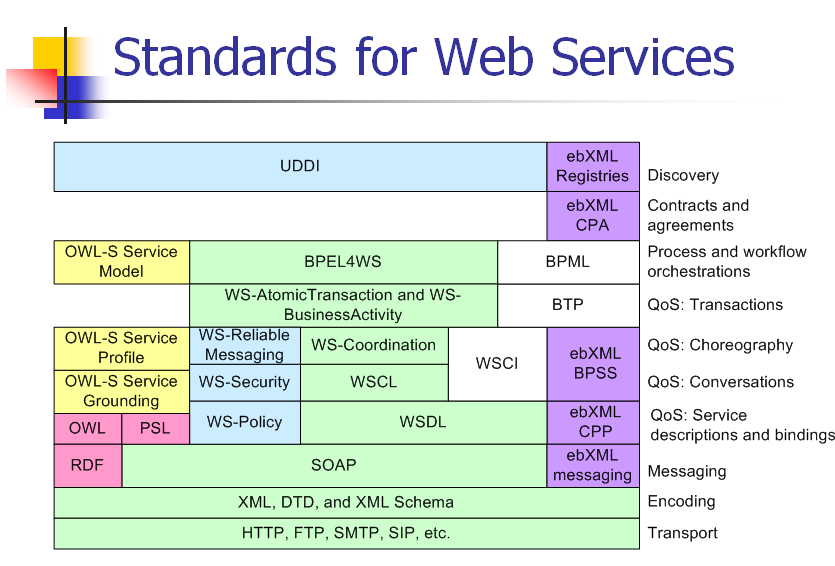


Figure 3

# Web Service Security:

La sécurité est essentielle pour les Web Services. Toutefois, ni le XML-RPC ni les spécifications SOAP ne font toutes les exigences de sécurité ou d'authentification explicites.

Il ya trois problèmes de sécurité spécifiques avec des services Web:

* Confidentialité
* Authentification
* Sécurité de réseau

**Confidentialité**

Si un client envoie une requête XML à un serveur, alors la question qui se pose est la suivante: Est-ce que la communication demeure confidentielle?

La Réponse se trouve ici :

* XML-RPC and SOAP run primary on top of HTTP.
* HTTP a le support de Secure Sockets Layer (SSL).
* Communication peut être cryptée par le Secure Sockets Layer SSL.
* SSL est une technologie éprouvée et largement déployé.

Mais le simple Web Service peut consister en une chaîne d'applications. Par exemple, un grand service pourrait lier les services de trois autres demandes. Dans ce cas, le protocole SSL n'est pas suffisant, les messages doivent être chiffrés à chaque nœud le long du trajet de service, et chaque nœud représente un maillon faible potentiel de la chaîne. Actuellement, il n'y a pas une solution convenable à ce problème, mais une solution prometteuse est le « W3C XML Encryption Standard ». Cette norme fournit un cadre pour chiffrer et déchiffrer des documents entiers XML ou seulement des parties d'un document XML.

**Authentification**

Si un client est connecté au Web Service, comment on peut identifier l’utilisateur? Et l'utilisateur est-il autorisé à utiliser le service?

Les options suivantes peuvent être considérées mais il n’y a pas une vision claire sur un système d’authentification forte.

* HTTP comprend un support intégré pour l'authentification Basic et Digest, et des services peuvent donc être protégés de la même manière que les documents HTML qui sont actuellement protégés.
* Extensions de sécurité SOAP: Signature numérique (SOAP DSIG) qui s'appuie sur la cryptographie à clé publique pour signer numériquement des messages SOAP. Cela permet au client ou au serveur de valider l'identité de l'autre partie.

**Sécurité de réseau**

Il n'existe actuellement aucune réponse facile à ce problème, et il a fait l'objet de beaucoup de débats. Pour l'instant, une possibilité est de filtrer toutes les requêtes HTTP POST qui fixent leur type de contenu Text/XML.

Une autre solution est de filtrer l'en-tête d'action SOAP attribut HTTP. Fournisseurs de Firewall sont également en train de développer des outils visant explicitement à filtrer le trafic du Web Service.

# 7 - Intégration et Interopérabilité:

L'intégration est le processus d'adhésion des systèmes distincts de manière qu'ils apparaissent comme un ensemble en un point de vue particulier.

L'interopérabilité est la capacité d'échanger des données entre deux ou plusieurs systèmes, c'est la capacité de faire travailler des systèmes et des organisations ensemble.

Si deux ou plusieurs systèmes sont capables de communiquer et d'échanger des données, ils font preuve d’interopérabilité syntaxique. Formats de données spécifiés, protocoles de communication et autres sont fondamentales. Normes XML ou SQL sont parmi les outils d'interopérabilité syntaxique. Cela est également vrai pour les formats de données de niveau inférieur, comme le fait d’assurer des caractères alphabétiques qui sont stockés dans une même variation de l'ASCII ou un format Unicode dans tous les systèmes communicants.

Au-delà de la capacité de deux ou plusieurs systèmes informatiques pour échanger des informations, l'interopérabilité sémantique est la capacité à interpréter automatiquement les informations échangées de manière significative et avec précision, afin de produire des résultats utiles, tels que définis par les utilisateurs finaux des deux systèmes. Pour réaliser l'interopérabilité sémantique, les deux parties doivent se référer à un modèle de référence de l'échange d'informations commun. Le contenu des demandes de renseignements de change sont définies sans ambiguïté : l’information envoyée est la même que l’information comprise. 8- Interopérabilité avec Moodle:

Moodle est une plate-forme d'apprentissage conçue pour fournir les éducateurs, les administrateurs et les apprenants d'un système robuste, sécurisé, intégré et unique pour créer des environnements d'apprentissage personnalisés.

Moodle fournit le l’outil le plus flexible pour soutenir à la fois l'ap prentissage et 100 % des cours en ligne. Moodle est configuré en activant ou en désactivant des fonctionnalités de base, et en intégrant facilement tout le nécessaire pour un cours en utilisant sa gamme complète de fonctionnalités intégrés, y compris des outils de collaboration externes tels que les forums , les wikis et chats.

Parce qu'il est open-source, Moodle peut être personnalisé en plusieurs manières et adapté aux besoins individuels. Son modulaire mis en place et la conception interopérable permet aux développeurs de créer des plugins et intégrer des applications externes pour réaliser des fonctionnalités spécifiques.

Le support pour les standards de Web Services dans l'ensemble de base de code de Moodle, permet à l’administrateur d'exposer des fonctions particulières de Moodle pour être utilisé par:

• Les systèmes administratifs tels que les applications de Ressources Humaines ou SIS «Student Information System ».

• Les téléphones mobiles des clients.

Cadre contenant un très haut niveau de sécurité avec un système de jetons détaillé et un contrôle complet sur l'ensemble des fonctions exposées

Toutes les fonctions définies sont automatiquement disponibles via:

• XML-RPC

• AMF (Flash)

• REST

• SOAP (PHP)

Moodle Web Service – Comment ça marche?

Cet exemple simple vous donnera une idée de la façon dont notre infrastructure de Moodle Web Service fonctionne.

Le client envoie un nom d'utilisateur et un mot de passe au script de connexion de Web Service. Ce script retourne un jeton pour ce compte d'utilisateur.

Le client appelle une fonction particulière de Web Service par un serveur de protocole, y compris le jeton.

Le serveur de protocole utilise le jeton pour vérifier que l'utilisateur peut appeler la fonction. Le serveur de protocole appelle la fonction externe correspondante, située dans un fichier de « externallib.php» à l'intérieur du module correspondant.

La fonction externe vérifie que l’utilisateur actuel « has\_capability»  fait cette opération .La fonction externe appelle la fonction correspondante de base de Moodle (habituellement lib.php) .

La fonction de base peut retourner un résultat à la fonction externe. La fonction externe renvoie un résultat au serveur de protocole. Le serveur de protocole renvoie le résultat au client.

--Type definition for Course

TYPE Rec\_Course IS RECORD

(fullname VARCHAR2(1024),

shortname VARCHAR2(1024),

categoryid VARCHAR2(255),

idnumber VARCHAR2(255),

startdate VARCHAR2(255),

status VARCHAR2(50),

id VARCHAR2(100));

--Define table of courses

TYPE Tab\_Type\_Crs IS TABLE OF Rec\_Course INDEX BY BINARY\_INTEGER;

--Get course information to PL/SQL stored procedure utilizing Moodle web service

FUNCTION Get\_Course\_Information(p\_crs\_id VARCHAR2) RETURN Tab\_Type\_Crs

IS

v\_debug\_mode BOOLEAN := FALSE;

v\_req utl\_http.req;

v\_resp utl\_http.resp;

v\_msg VARCHAR2(80);

v\_entire\_msg VARCHAR2(32767) := NULL;

v\_conversion\_factor NUMBER;

v\_url VARCHAR2(4000) := '';

V\_Tab\_Crs Tab\_Type\_Crs;

value VARCHAR2 (1024);

value1 VARCHAR2 (4000);

Cursor Sel\_Xml IS

Select

EXTRACT(XMLTYPE(value1),'/RESPONSE/MULTIPLE/SINGLE/KEY[1]/VALUE/text()').GetStringVal() id,

EXTRACT(XMLTYPE(value1),'/RESPONSE/MULTIPLE/SINGLE/KEY[5]/VALUE/text()').GetStringVal() fullname,

EXTRACT(XMLTYPE(value1),'/RESPONSE/MULTIPLE/SINGLE/KEY[2]/VALUE/text()').GetStringVal() shortname,

EXTRACT(XMLTYPE(value1),'/RESPONSE/MULTIPLE/SINGLE/KEY[3]/VALUE/text()').GetStringVal() categoryid

from dual;

R\_Xml Sel\_Xml%ROWTYPE;

i number;

BEGIN

V\_Tab\_Crs(1).idnumber := p\_crs\_id;

--Calling the moodle web method called core\_course\_get\_course\_by\_field utilizing REST

V\_url := 'http://my-moodle.com:8081/'|| 'webservice/rest/server.php?'||

'wstoken=98bf3542f43b469cea348cfdf7392336'||'&wsfunction=core\_course\_get\_course\_by\_field&criteria[0][key]=idnumber&criteria[0][value]='||V\_Tab\_Crs(1).idnumber;

--Request

v\_req := utl\_http.begin\_request(url => v\_url, method => 'GET');

--Get Response

v\_resp := utl\_http.get\_response(v\_req);

i := 0;

BEGIN

LOOP

utl\_http.read\_line(v\_resp, value,false);

value1 := value1||value;

i := i+1;

END LOOP;

UTL\_HTTP.END\_RESPONSE(v\_resp);

EXCEPTION

WHEN utl\_http.end\_of\_body THEN

UTL\_HTTP.END\_RESPONSE(v\_resp);

WHEN OTHERS THEN

begin

utl\_http.end\_response(v\_resp);

EXCEPTION WHEN OTHERS THEN NULL;

END;

END;

OPEN Sel\_Xml;

LOOP

FETCH Sel\_Xml INTO R\_Xml;

EXIT WHEN Sel\_Xml%NOTFOUND;

V\_Tab\_Crs(1).id := R\_Xml.id;

V\_Tab\_Crs(1).fullname := R\_Xml.fullname;

V\_Tab\_Crs(1).shortname := R\_Xml.shortname;

V\_Tab\_Crs(1).categoryid := R\_Xml.categoryid;

END LOOP;

CLOSE Sel\_Xml;

RETURN V\_Tab\_Crs;

END Get\_Crs\_info;

Annexe 1 – Exemple XML

<?xml version="1.0"?> <!– not part of the document per se 🡪

<arbitrary:toptag xmlns=“http://one.default.namespace/if-needed”

xmlns:arbitrary=“http://wherever.it.might.be/arbit-ns”

      xmlns:random=“http://another.one/random-ns”>

     <arbitrary:atag attr1=“v1” attr2=“v2”>

Optional text also known as PCDATA

<arbitrary:btag attr1=“v1” attr2=“v2” />

</arbitrary:atag>

<random:simple\_tag/>

<random:atag attr3=“v3”/> <!– compare with arbitrary:atag above 🡪

</arbitrary:toptag>

Annexe 2 – Exemple SOAP

SOAP - Requête

POST /temp HTTP/1.1

Host: www.socweather.com

Content-Type: text/xml; charset="utf-8"

Content-Length: xxx

SOAPAction: "http://www.socweather.com/temp"

<!-- The above are HTTP headers -->

<?xml version=“1.0”?>

<env:Envelope

xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"

env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>

<env:Body>

<m:GetTemp xmlns:m="http://www.socweather.com/temp.xsd">

<m:City>Honolulu</m:City>

<m:When>now</m:When>

</m:GetTemp>

</env:Body>

</env:Envelope>

SOAP – Réponse

HTTP/1.1 200 OK

Content-Type: text/xml; charset="utf-8"

Content-Length: xxx

SOAPAction: "http://www.socweather.com/temp"

<?xml version=“1.0”?>

<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"

env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>

<env:Body>

<m:GetTempResponse xmlns:m="http://www.socweather.com/temp.xsd">

<DegreesCelsius>30</DegreesCelsius>

</m:GetTempResponse>

</env:Body>

</env:Envelope>

Annexe 3 – Exemple WSDL

<?xml version="1.0"?>

<!-- the root element, wsdl:definitions, defines a set of -->

<!-- related services -->

<wsdl:definitions

name="Temperature"

targetNamespace="http://www.socweather.com/schema"

xmlns:ts="http://www.socweather.com/TempSvc.wsdl"

xmlns:tsxsd="http://schemas.socweather.com/TempSvc.xsd"

xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"

xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">

<!-- wsdl:types encapsulates schema definitions of -->

<!-- communication types; here using xsd -->

<wsdl:types>

<!-- all type declarations are expressed in xsd -->

<xsd:schema targetNamespace="http://namespaces.socweather.com"

xmlns:xsd="http://www.w3.org/1999/XMLSchema">

<!-- xsd def: GetTemp [City string, When string] -->

<xsd:element name="GetTemp">

<xsd:complexType>

<xsd:sequence>

<xsd:element name="City" type="string"/>

<xsd:element name="When" type="string"/>

</xsd:sequence>

</xsd:complexType>

</xsd:element>

<!-- xsd def: GetTempResponse [DegreesCelsius integer] -->

<xsd:element name="GetTempResponse">

<xsd:complexType>

<xsd:all>

<xsd:element name="DegreesCelsius" type="integer"/>

</xsd:all>

</xsd:complexType>

</xsd:element>

<!-- xsd def: GetTempFault [errorMessage string] -->

<xsd:element name="GetTempFault">

<xsd:complexType>

<xsd:all>

<xsd:element name="errorMessage" type="string"/>

</xsd:all>

</xsd:complexType>

</xsd:element>

</xsd:schema>

</wsdl:types>

<!-- wsdl:message elements describe potential transactions -->

<!-- Most messages, as here, have only one part. Multiple -->

<!-- parts provide a way to aggregate complex messages -->

<!-- request GetTempRequest is of type GetTemp -->

<wsdl:message name="GetTempRequest">

<wsdl:part name="body" element="tsxsd:GetTemp"/>

</wsdl:message>

<!-- response GetTempResponse is of type GetTempResponse -->

<wsdl:message name="GetTempResponse">

<wsdl:part name="body" element="tsxsd:GetTempResponse"/>

</wsdl:message>

<!-- wsdl:portType describes messages in an operation -->

<wsdl:portType name="GetTempPortType">

<!-- wsdl:operation describes the entire protocol from -->

<!-- input to output or fault -->

<wsdl:operation name="GetTemp">

<!-- The order of input and output is significant; input -->

<!-- preceding output indicates the request-response -->

<!-- operation type -->

<wsdl:input message="ts:GetTempRequest"/>

<wsdl:output message="ts:GetTempResponse"/>

<wsdl:fault message="ts:GetTempFault"/>

</wsdl:operation>

</wsdl:portType>

<!-- wsdl:binding states a serialization protocol for -->

<!-- this service -->

<wsdl:binding name="TempSvcSoapBinding" type="ts:GetTempPortType">

<!-- leverage off soap:binding document style -->

<soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>

<!-- semi-opaque container of network transport details -->

<!-- classed by soap:binding above @@@ -->

<wsdl:operation name="GetTemp">

<soap:operation soapAction="http://www.socweather.com/TempSvc"/>

<!-- further specify that the messages in the -->

<!-- wsdl:operation "GetTemp" use SOAP? @@@ -->

<wsdl:input>

<soap:body use="literal" namespace="http://schemas.socweather.com/TempSvc.xsd"/>

</wsdl:input>

<wsdl:output>

<soap:body use="literal" namespace="http://schemas.socweather.com/TempSvc.xsd"/>

</wsdl:output>

<wsdl:fault>

<soap:body use="literal" namespace="http://schemas.socweather.com/TempSvc.xsd"/>

</wsdl:fault>

</wsdl:operation>

</wsdl:binding>

<!-- wsdl:service names a new service "TemperatureService" -->

<wsdl:service name="TemperatureService">

<wsdl:documentation>socweather.com temperature service

</wsdl:documentation>

<!-- connect it to the binding "TempSvcSoapBinding" above -->

<wsdl:port name="GetTempPort" binding="ts:TempSvcSoapBinding">

<!-- give the binding a network address -->

<soap:address location="http://www.socweather.com/TempSvc"/>

</wsdl:port>

</wsdl:service>

</wsdl:definitions>

**Bibliographie**

<http://en.wikipedia.org/wiki/Interoperability><http://www.slideshare.net/bsadd/rest-web-services 21524315?v=qf1&b=&from_search=4>  
<http://www.slideshare.net/pohjus/introduction-to-xml-presentation?v=default&b=&from_search=1>  
<http://www.slideshare.net/pohjus/xml-schema-w3c-presentation>  
<http://www.slideshare.net/AnnieComp/presentation-24088326?v=qf1&b=&from_search=12>  
<http://www.talend.com/solutions/service-oriented-architecture>  
<http://www.mn.uio.no/ifi/english/research/networks/hisp/integrated-health-information-architecture/ch-03.pdf><https://moodle.org/>  
<http://docs.moodle.org/dev/Web_services>  
<http://docs.moodle.org/dev/Creating_a_web_service_client>