



---

# SI-ACVS WebServices

---

Intégration ACVS

V 0.3 / mardi 7 février 2006



LISTE DE DIFFUSION				
Organisme ou Entreprise	Noms des Destinataires	Nombre de copies	Pour	
			Action	Information
CDT83	B. BEZON	1	✓	
SYNEXIE	S. LARRASOAIN	1	✓	
SYNEXIE	D. TORCK	1	✓	

SUIVI DES VERSIONS			
Version	Date	Rédacteur	Commentaires
0.1	22/09/2004	Stephan LARRASOAIN	
0.2	17/11/2004	Calendau GUQUET	
0.3	07/02/2006	Stephan LARRASOAIN	

ETAT DES MISES A JOUR		
Chapitre	Motif et nature des mises à jour	Version



## SOMMAIRE

<b>1. INTRODUCTION .....</b>	<b>3</b>
<b>2. MODELE DE PROGRAMMATION DES SERVICES WEB XML .....</b>	<b>3</b>
2.1. VUE D'ENSEMBLE DES SERVICES WEB XML .....	3
<b>3. INFRASTRUCTURE DES SERVICES WEB XML .....</b>	<b>3</b>
<b>4. DESCRIPTION DE SERVICE WEB XML .....</b>	<b>3</b>
<b>5. FORMATS DE TRANSMISSION DE SERVICE WEB XML .....</b>	<b>3</b>
5.1. HTTP-GET ET HTTP-POST .....	3
5.2. SOAP .....	3
<b>6. UTILISATION DES SERVICES WEB DU SI-ACVS .....</b>	<b>3</b>
6.1. FORMAT DE TRANSMISSION DES SERVICES WEB DU SI-ACVS.....	3
6.2. SECURITE ET EN TETE SOAP OBLIGATOIRE .....	3
6.3. INFORMATIONS MISES A DISPOSITION PAR LES SERVICES WEB.....	3
<b>7. METHODES MISE A DISPOSITION PAR LES SERVICES WEB DU SI-ACVS.....</b>	<b>3</b>
7.1. GETCONFIGURATIONELEMENT .....	3
7.2. GETPRESTA.....	3
7.3. GETPRESTATIONS .....	3
7.4. GETFULLINFOSPRESTATION .....	3
7.5. GETCARACTERISTIQUE .....	3
7.6. GETÉTATCARACTERISTIQUE .....	3
7.7. GETCATEGORIE .....	3
7.8. GETSOUSCATEGORIE.....	3
7.9. GETCOMMUNE.....	3
7.10. REMARQUES : .....	3



## 1. INTRODUCTION

Ce document a pour ambition d'atteindre deux objectifs :

- Fournir des notions de bases sur ce que sont les Services Web XML.
- Décrire précisément ceux mis à disposition dans le SI-ACVS.

Ceci devra permettre à toute structure touristique référencée dans le système d'être en mesure de mettre en œuvre des procédures de dialogue entre une entité externe (site Web, système de réservation, ...) et le SI-ACVS.

Les services Web XML permettent d'échanger des données et d'appeler à distance la logique de l'application à l'aide de la messagerie XML.

Ce type d'architecture permet de transférer des données à travers entre systèmes hétérogènes, et de franchir des éléments actifs tels que les firewalls (pare-feux).

Même si l'accès distant à des données ou à la logique d'une l'application n'est pas un concept nouveau, le couplage faible dû au fonctionnement déconnecté de ce type d'architecture garantie un fonctionnement au sein d'un environnement hétérogène, atout inexistant jusqu'à lors.

Le seul préalable à mettre en place au niveau du client du service Web XML et du serveur lui-même, est de fixer un langage intelligible pour chacune des deux entités. Le XML répond parfaitement à cette demande, puisqu'il s'agit d'un mode de description simple à utiliser, dans n'importe quel environnement. En conséquence, les programmes peuvent accéder aux services Web XML, quel que soit le langage dans lequel ils sont écrits, le modèle de composant utilisé et le système d'exploitation sur lequel ils s'exécutent.



## 2. MODELE DE PROGRAMMATION DES SERVICES WEB XML

Le modèle de programmation des services Web XML comporte deux aspects fondamentaux :

- **Création d'un service Web XML** – La création d'un service Web XML, revient à créer une application qui expose des fonctionnalités aux clients de ce service Web XML. Ce service Web est accessible via une URL donnée, et renvoie, en fonction des paramètres passés dans la requête, un flux de réponse formaté en XML. De plus, cette application possède une adresse particulière à partir de laquelle elle va être capable de s'auto-décrire, de telle sorte qu'un client puisse connaître le mode de dialogue utilisé.

- **Accès à un service Web XML** – Lorsqu'une application cliente accède à un service Web XML, elle recherche les fonctionnalités contenues dans ce service Web XML, y fait référence et les utilise. Le client d'un service Web XML peut être une application basée sur un navigateur (comme un site Web), un composant ou même un autre service Web XML.

Les services Web XML sont pratiquement accessibles à partir de n'importe quel autre type d'application, y compris les autres services Web XML et les applications Web, Windows ou console. La seule condition est que le client doit pouvoir échanger des messages avec le service Web XML et traiter ce type de message.

### 2.1. Vue d'ensemble des services Web XML

Un service Web XML est une entité non visible, exposée sur un réseau local ou sur Internet (donc accessible par ce biais), capable d'exposer des méthodes donnant accès à la logique d'une application ou certains de ses aspects (export de données, traitements, etc...)

Un service Web repose uniquement sur des standards Internet, à savoir XML et le protocole HTTP. C'est grâce à l'utilisation de ces normes omniprésentes que l'interopérabilité entre systèmes disparates a été rendue possible. De plus, l'utilisation de tels standards permet de s'affranchir des problèmes de sécurité liés à l'utilisation de proxy ou de firewall, puisque les seuls protocoles utilisés pour la navigation sur Internet sont exploités.

Grâce aux Services Web, des systèmes hétérogènes vont être amenés à travailler de concert, certains déléguant d'autres tâches à d'autres pour en exploiter après le résultat.

Les architectures d'applications distribuées reposaient par le passé sur un assemblage de composants référencés, ce qui posait inévitablement le souci des langages ou systèmes incompatibles.

Les Service Web proposent un assemblage de sources variées, et dialoguant au moyen d'un système de messages unifiés, intelligibles et portables, des messages XML.

L'une des caractéristiques principales d'un service Web XML est le haut degré d'abstraction qu'il existe entre l'implémentation et la consommation d'un service. En utilisant la messagerie XML comme mécanisme de création du service et d'accès au service, les client et fournisseur du service Web XML n'ont plus besoin d'avoir une connaissance mutuelle qui aille au-delà des entrées, sorties et emplacements.

Les services Web XML ouvrent une nouvelle ère dans le développement d'applications distribuées. Il n'est plus question désormais de rivalités entre les modèles objet ou entre langages de programmation. Lorsque des systèmes sont étroitement couplés à l'aide d'infrastructures propriétaires, cela s'effectue aux dépens de l'interopérabilité des applications. Les services Web XML assurent l'interopérabilité à un niveau totalement nouveau, qui met un terme aux rivalités contre-productives décrites précédemment. Les services Web XML constituent la prochaine révolution d'Internet et, en tant que tels, ils deviendront la structure fondamentale qui relie tous les périphériques informatiques.



### 3. INFRASTRUCTURE DES SERVICES WEB XML

Les services Web XML doivent présenter les caractéristiques suivantes :

- **Être faiblement couplés** : deux systèmes sont considérés comme faiblement couplés si la seule mission qui leur est imposée consiste à reconnaître les messages texte autodésignés décrits précédemment. En revanche, les systèmes étroitement couplés nécessitent une charge mémoire personnalisée importante pour permettre la communication et exigent une plus grande connaissance réciproque des deux systèmes.

- **Assurer la communication omniprésente** : il est peu probable que quelqu'un crée aujourd'hui ou dans un avenir proche un système d'exploitation qui n'offre pas la possibilité de se connecter à Internet, via un canal de communication standard. Dès lors, la possibilité de connecter presque n'importe quel système ou périphérique à Internet garantit que ces systèmes ou périphériques sont disponibles universellement à tout autre système ou périphérique connecté à Internet.

- **Proposer un format de données universel** : l'adoption de normes ouvertes existantes sur des méthodes de communication propriétaire en boucle fermée permet à tout système prenant en charge les mêmes normes ouvertes de comprendre les services Web XML. L'utilisation de messages texte autodésignés que les services Web XML et leurs clients peuvent partager sans nécessairement connaître les fondements de chaque système sous-jacent permet une communication entre des systèmes autonomes et hétérogènes. Cette fonctionnalité est assurée par les services Web XML grâce au langage XML.

Les services Web XML utilisent une infrastructure qui assure les éléments suivants : un mécanisme de découverte pour la localisation des services Web XML, une description de service pour définir leur mode d'utilisation et des formats de transmission standard au moyen desquels communiquer. L'illustration ci-dessous présente un exemple de cette infrastructure.

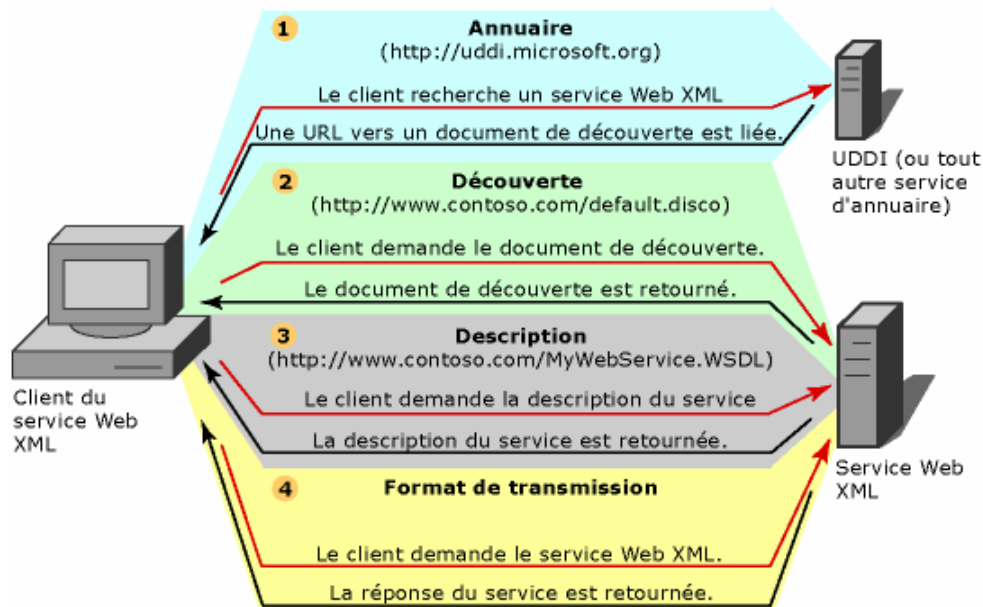


Figure 1: Infrastructure des services Web XML



## 4. DESCRIPTION DE SERVICE WEB XML

L'infrastructure des services Web XML est fondée sur la communication via des messages XML conformes à une description de service publié. Cette description de service est un document XML écrit à l'aide d'une grammaire XML appelée WSDL (Web Services Description Language), qui définit le format des messages reconnus par le service Web XML en question. Elle joue le rôle d'un accord définissant le comportement d'un service Web XML et indique aux clients potentiels comment interagir avec ce service. Le comportement d'un service Web XML est déterminé par les modèles de message définis et pris en charge par le service. Ces modèles stipulent de façon conceptuelle ce à quoi le consommateur du service doit s'attendre lorsqu'un message mis en forme correctement est envoyé au service Web XML.

Par exemple, le modèle demande/réponse associé à un service de type appel de procédure distante (RPC, *Remote Procedure Call*) pourrait définir le schéma de message SOAP à utiliser pour appeler une méthode particulière. Il indiquerait également le format qu'aurait le message SOAP de réponse consécutif.

Autre exemple de modèle de message : les interactions unidirectionnelles. Ce modèle est employé lorsque la communication unidirectionnelle doit avoir lieu. Dans cette situation, l'expéditeur ne recevra pas de messages du service Web XML, pas même des messages d'erreur. Il existe toutefois une exception : lorsque la communication unidirectionnelle est établie à l'aide d'un protocole traditionnellement de demande/réponse, un message d'erreur peut être retourné.

Outre les définitions de format de message et les modèles de messagerie, la description de service peut éventuellement contenir l'adresse associée à chaque point d'entrée du service Web XML. Le format de cette adresse est adapté au protocole utilisé pour accéder au service, tel qu'une adresse URL pour HTTP ou une adresse de messagerie pour SMTP.

Pour la spécification WSDL, consultez le site Web du W3C (<http://www.w3.org/TR/wsdl>).



## 5. FORMATS DE TRANSMISSION DE SERVICE WEB XML

Contrairement aux architectures distribuées basées sur DCOM, qui proposaient, au dessus de la logique métier d'une application, un protocole binaire, standard seulement entre serveurs Windows, les services web XML offrent un flux de données intelligible quel que soit le système client, puisqu'il s'agit d'interpréter une description formatée au moyen de balises et d'attributs.

On pourrait comparer ce type d'interopérabilité avec le fait que tous les serveurs Web sont compatibles avec tous les navigateurs Web, quels que soient les logiciels serveurs et client utilisés, grâce à l'utilisation du HTML, dont XML est d'ailleurs une extension.

Cette garantie d'interopérabilité est assurée par l'utilisation, pour véhiculer les messages, du protocole HTTP. Par ce protocole il est possible d'appeler des méthodes à distance, de leur passer des paramètres et de récupérer les réponses, en utilisant des types primitifs (entiers, réels, chaînes de caractères, ...) ou complexes, dans la limite où ces types complexes ou structurés peuvent être sérialisés pour l'envoi et désérialisés pour la lecture.

### 5.1. HTTP-GET et HTTP-POST

HTTP-GET et HTTP-POST sont des protocoles standard qui utilisent des verbes HTTP (Hypertext Transfer Protocol) pour le codage et le passage de paramètres sous la forme de paires nom/valeur, en même temps que la sémantique de demande associée. Chacun se compose d'une série d'en-têtes de demandes HTTP qui, entre autres, définissent l'objet de la demande émise par le client au serveur, lequel répond en envoyant une série d'en-têtes de réponse HTTP et les données requises si la demande aboutit.

HTTP-GET passe ses paramètres sous la forme de texte codé URL à l'aide du type MIME application/x-www-form-urlencoded, qui est ajouté à la fin de l'URL du serveur qui traite la demande. Le codage URL est une forme de codage de caractères garantissant que les paramètres passés sont constitués de texte conforme, comme lorsque l'espace est codé en %20. Les paramètres ajoutés sont également appelés chaîne de requête...

À l'instar des paramètres HTTP-GET, les paramètres HTTP-POST sont également codés URL. Cependant, au lieu d'être passés dans l'URL, les paires nom/valeur sont passées dans le message de demande HTTP proprement dit.

### 5.2. SOAP

SOAP est un protocole XML simple et léger pour l'échange d'informations de type et structurées sur le Web. Le principal objectif poursuivi lors du design de SOAP a été de privilégier la simplicité au maximum et d'assurer un minimum de fonctionnalité. Ce protocole définit une infrastructure de messagerie dépourvue de toute sémantique de transport ou d'application. Il s'agit donc d'un protocole modulaire et hautement extensible.

Parce qu'il circule au-dessus d'autres protocoles de transport standard, SOAP est en mesure de tirer parti de l'architecture ouverte existante d'Internet et d'être facilement accepté par n'importe quel système arbitraire capable de prendre en charge les normes Internet les plus élémentaires. Vous pouvez vous représenter l'infrastructure requise pour prendre en charge un service Web XML conforme à SOAP comme relativement simpliste bien que puissante, puisqu'elle n'ajoute pas énormément à l'infrastructure existante d'Internet tout en permettant un accès universel aux services créés à l'aide de SOAP.

La spécification du protocole SOAP se compose de quatre parties principales. La première définit une enveloppe extensible obligatoire pour l'encapsulation des données. Cette enveloppe SOAP définit un message SOAP et constitue l'unité de base de l'échange entre des processeurs de messages SOAP. Il s'agit de la seule partie obligatoire de la spécification.



La deuxième partie de la spécification du protocole SOAP définit des règles de codage des données facultatives pour représenter des graphiques dirigés et des types de données définis par l'application, ainsi qu'un modèle uniforme pour la sérialisation des modèles de données non syntaxiques.

La troisième partie décrit un modèle d'échange de messages de type RPC (demande/réponse). Chaque message SOAP est une transmission unidirectionnelle. Bien que SOAP prenne ses racines dans RPC, il ne s'agit pas uniquement d'un simple mécanisme de demande/réponse. Les services Web XML combinent souvent des messages SOAP pour implémenter de tels modèles, mais SOAP n'impose pas de modèle pour l'échange de messages et cette partie de la spécification est également facultative.

La quatrième partie, quant à elle, prescrit une liaison entre SOAP et HTTP. Elle est facultative également. Vous pouvez associer SOAP à n'importe quel mécanisme ou protocole de transport capable de transporter l'enveloppe SOAP, y compris SMTP, FTP ou même une disquette.

Pour la spécification SOAP, consultez le site Web W3C (<http://www.w3.org/TR/soap>).



## 6. UTILISATION DES SERVICES WEB DU SI-ACVS

### 6.1. Format de transmission des services web du SI-ACVS

Les requêtes de consommation des services Web du SI-ACVS doivent naturellement être conformes au protocole XML SOAP.

Cependant, elles doivent respecter un certain format que nous avons défini, afin de pouvoir incorporer le header spécifique à l'authentification.

Une description de la requête SOAP à émettre au serveur de Service Web pour chaque fonction disponible est réalisée dans la partie 7 de ce document.

### 6.2. Sécurité et En tête SOAP obligatoire

Les Web Services publiés au sein du SI-ACVS mettent en place une double authentification afin de sécuriser leur accès :

- Toutes les requêtes SOAP doivent contenir un « Header » particulier (cf. schéma ci-dessous) dans lequel sont transmis le login et le mot de passe du compte WebService correspondant à une structure (Chaque structure du SI-ACVS a la possibilité de créer des comptes WebServices ).

- Le serveur réalise ensuite une vérification de l'adresse IP source de la requête pour vérifier si cette IP fait bien partie de la liste des IP autorisées pour ce compte WebService. C'est la structure propriétaire du WebService qui paramètre la liste des IP autorisés.

```
<soap:Header>
  <AuthHeader xmlns="http://www.acvsnet.net/SI-ACVS/WebServices">
    <Login>string</Login>
    <Password>string</Password>
  </AuthHeader>
</soap:Header>
```

*Header obligatoire pour chaque requête sur les WebService.*

### 6.3. Informations mises à disposition par les services web

L'authentification réalisée pour chaque fonction permet au serveur de service Web de connaître la structure pour laquelle il doit retourner l'information. Ainsi les services web ne permettent de retourner que des informations dont la structure est « propriétaire ». Les informations retournées sont donc celles disponibles pour la zone de compétence définie dans le SI-ACVS.



## 7. METHODES MISE A DISPOSITION PAR LES SERVICES WEB DU SI-ACVS

### » Introduction :

Une description des services web du SI-ACVS est accessible en ligne à l'adresse suivante <http://www.acvsnet.net/SI-ACVS/WebServices/ACVSWebServices.asmx> . Les paragraphes suivants présentent plus en détail les différentes fonctions mises à disposition par les services web.

### 7.1. GetConfigurationElement

#### » Contenu :

Cette fonction permet de récupérer un groupe de donnée (DataSet) contenant trois groupes d'information (Table).

Les informations retournées sont les tables contenant les paramètres de base pour la structure. Ainsi on récupère :

- Dans une table Commune, la listes des communes de la zone de compétence de la structure.
- Dans une table GroupeCategorie, la liste des groupes de catégorie des prestations.
- Dans une table Categorie, la liste des catégorie disponible pour chaque groupe catégorie.

#### » Listes des paramètres :

- Cette méthode ne nécessite aucun paramètre.

Le texte suivant est un exemple de demande SOAP pour appeler cette fonction. Les **espaces réservés** affichés doivent être remplacés par des valeurs réelles.

```
POST /WebServices/ACVSWebServices.asmx HTTP/1.1
Host: www.acvsnet.net
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://www.acvsnet.net/SI-ACVS/WebServices/GetConfigurationElement"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <AuthHeader xmlns="http://www.acvsnet.net/SI-ACVS/WebServices">
      <Login>string</Login>
      <Password>string</Password>
    </AuthHeader>
  </soap:Header>
  <soap:Body>
    <GetConfigurationElement xmlns="http://www.acvsnet.net/SI-ACVS/WebServices"
/>
  </soap:Body>
</soap:Envelope>
```

#### » Réponse à la requête :

Le service web renvoie le format de réponse suivant à une requête conforme au format énoncé précédemment.



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

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <GetConfigurationElementResponse xmlns="http://www.acvsnet.net/SI-
ACVS/WebServices">
      <GetConfigurationElementResult>
        <xsd:schema>schema</xsd:schema>xml</GetConfigurationElementResult>
      </GetConfigurationElementResponse>
    </soap:Body>
  </soap:Envelope>
```

Ci-dessous un exemple du schéma de la réponse :

```
<xs:schema id="NewDataSet" xmlns="" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
  <xs:element name="NewDataSet" msdata:IsDataSet="true" msdata:Locale="fr-FR">
    <xs:complexType>
      <xs:choice maxOccurs="unbounded">
        <xs:element name="Commune">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="idCommune" type="xs:int" minOccurs="0" />
              <xs:element name="nomCommune" type="xs:string" minOccurs="0" />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="GroupeCategorie">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="idGroupeCategorie" type="xs:int" minOccurs="0" />
              <xs:element name="libGroupeCategorie" type="xs:string" minOccurs="0" />
              <xs:element name="fullLibGroupeCategorie" type="xs:string" minOccurs="0" />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="Categorie">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="idCategorie" type="xs:int" minOccurs="0" />
              <xs:element name="libCategorie" type="xs:string" minOccurs="0" />
              <xs:element name="fullLibCategorie" type="xs:string" minOccurs="0" />
              <xs:element name="idGroupeCategorie" type="xs:int" minOccurs="0" />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:choice>
    </xs:complexType>
  </xs:element>
</xs:schema>
```



Ci-dessous un exemple de DataSet retourné.

```
<diffgr:diffgram xmlns:msdata="urn:schemas-microsoft-com:xml-msdata"
xmlns:diffgr="urn:schemas-microsoft-com:xml-diffgram-v1">
  <NewDataSet xmlns="">
    <Commune diffgr:id="Commune1" msdata:rowOrder="0">
      <idCommune>343</idCommune>
      <nomCommune>LA VALETTE-DU-VAR</nomCommune>
    </Commune>
    <GroupeCategorie diffgr:id="GroupeCategorie1" msdata:rowOrder="0">
      <idGroupeCategorie>1</idGroupeCategorie>
      <libGroupeCategorie>HEBE</libGroupeCategorie>
      <fullLibGroupeCategorie>Hébergement</fullLibGroupeCategorie>
    </GroupeCategorie>
    <GroupeCategorie diffgr:id="GroupeCategorie2" msdata:rowOrder="1">
      <idGroupeCategorie>2</idGroupeCategorie>
      <libGroupeCategorie>ANIM</libGroupeCategorie>
      <fullLibGroupeCategorie>Animation</fullLibGroupeCategorie>
    </GroupeCategorie>...
    <Categorie diffgr:id="Categorie1" msdata:rowOrder="0">
      <idCategorie>27</idCategorie>
      <libCategorie>ADMI</libCategorie>
      <fullLibCategorie>ADMINISTRATION </fullLibCategorie>
      <idGroupeCategorie>4</idGroupeCategorie>
    </Categorie>
    <Categorie diffgr:id="Categorie2" msdata:rowOrder="1">
      <idCategorie>8</idCategorie>
      <libCategorie>AGI </libCategorie>
      <fullLibCategorie>AGENCE IMMOBILIERE</fullLibCategorie>
      <idGroupeCategorie>1</idGroupeCategorie>
    </Categorie>...
  </NewDataSet>
</diffgr:diffgram>
```



## 7.2. GetPresta

### » Remarque :

Cette methode n'est plus à utiliser elle est gardée pour compatibilité avec les développements déjà réalisés. Utiliser GetPrestation ou GetFullInfosPrestation à la place

### » Contenu :

Cette fonction permet de récupérer un groupe de données (DataSet) contenant un groupe d'informations (Table).

Les informations retournées sont des prestations de différents types selon les paramètres renseignés :

### » Listes des paramètres :

- idPrestation : identifiant de la prestation demandé, -1 pour ne pas filtrer par l'idPrestation.
- idCategorie : identifiant de la catégorie de prestation, -1 pour ne pas filtrer par l'idCategorie.
- idCommune : identifiant d'une commune appartenant à la zone de compétence de la structure, -1 pour ne pas filtrer par commune.
- idGroupeCategorie : identifiant du groupe de catégorie, -1 pour ne pas filtrer par l'idGroupeCategorie.
- dayBorneInfOuverture : numéro du jour dans l'année courante à partir duquel on veut récupérer les prestations ouvertes, -1 pour ne pas filtrer par dayBorneInfOuverture.
- dayBorneSupOuverture : numéro du jour dans l'année courante jusqu'auquel on veut récupérer les prestations ouvertes, -1 pour ne pas filtrer par dayBorneSupOuverture.

Le texte suivant est un exemple de demande SOAP pour appeler cette fonction. Les **espaces réservés** affichés doivent être remplacés par des valeurs réelles.

```
POST /WebServices/ACVSWebServices.asmx HTTP/1.1
Host: www.acvsnet.net
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://www.acvsnet.net/SI-ACVS/WebServices/GetPresta"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <AuthHeader xmlns="http://www.acvsnet.net/SI-ACVS/WebServices">
      <Login>string</Login>
      <Password>string</Password>
    </AuthHeader>
  </soap:Header>
  <soap:Body>
    <GetPresta xmlns="http://www.acvsnet.net/SI-ACVS/WebServices">
      <_idPrestation>int</_idPrestation>
      <_idCategorie>int</_idCategorie>
      <_idCommune>int</_idCommune>
      <_idGroupeCategorie>int</_idGroupeCategorie>
      <dayBorneInfOuverture>int</dayBorneInfOuverture>
      <dayBorneSupOuverture>int</dayBorneSupOuverture>
    </GetPresta>
  </soap:Body>
</soap:Envelope>
```



» Réponse à la requête :

Le service web renvoie le format de réponse suivant à une requête conforme au format énoncé précédemment.

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

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <GetPrestaResponse xmlns="http://www.acvsnet.net/SI-ACVS/WebServices">
      <GetPrestaResult>
        <xsd:schema>schema</xsd:schema>xml</GetPrestaResult>
      </GetPrestaResponse>
    </soap:Body>
  </soap:Envelope>
```

Ci-dessous un exemple du schéma de la réponse :

```
<xs:schema id="NewDataSet" xmlns="" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
  <xs:element name="NewDataSet" msdata:IsDataSet="true" msdata:Locale="fr-FR">
    <xs:complexType>
      <xs:choice maxOccurs="unbounded">
        <xs:element name="Prestation">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="idPrestation" type="xs:int" minOccurs="0" />
              <xs:element name="libPrestation" type="xs:string" minOccurs="0" />
              <xs:element name="infoXML" type="xs:string" minOccurs="0" />
              <xs:element name="idCategorie" type="xs:int" minOccurs="0" />
              <xs:element name="idEtatDisponibilite" type="xs:int" minOccurs="0" />
              <xs:element name="idGroupeCategorie" type="xs:int" minOccurs="0" />
              <xs:element name="bGestionJourJ" type="xs:boolean" minOccurs="0" />
              <xs:element name="repere" type="xs:string" minOccurs="0" />
              <xs:element name="stringOuverture" type="xs:string" minOccurs="0" />
              <xs:element name="stringDispo" type="xs:string" minOccurs="0" />
              <xs:element name="information" type="xs:string" minOccurs="0" />
              <xs:element name="l2_CompRemise" type="xs:string" minOccurs="0" />
              <xs:element name="l3_CompDistrib" type="xs:string" minOccurs="0" />
              <xs:element name="l4_NumVoie" type="xs:string" minOccurs="0" />
              <xs:element name="l4_NomVoie" type="xs:string" minOccurs="0" />
              <xs:element name="l5_LieuDit_ServDistrib" type="xs:string" minOccurs="0" />
              <xs:element name="l6_CP" type="xs:string" minOccurs="0" />
              <xs:element name="l6_Ville" type="xs:string" minOccurs="0" />
              <xs:element name="idDepartement" type="xs:int" minOccurs="0" />
              <xs:element name="idPays" type="xs:int" minOccurs="0" />
              <xs:element name="telephone_fixe" type="xs:string" minOccurs="0" />
              <xs:element name="telephone_cellulaire" type="xs:string" minOccurs="0" />
              <xs:element name="fax" type="xs:string" minOccurs="0" />
              <xs:element name="mail" type="xs:string" minOccurs="0" />
              <xs:element name="site_web" type="xs:string" minOccurs="0" />
              <xs:element name="nomCommune" type="xs:string" minOccurs="0" />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:choice>
    </xs:complexType>
  </xs:element>
</xs:schema>
```





### 7.3. GetPrestations

#### » Contenu :

Cette fonction permet de récupérer un groupe de données (DataSet) contenant un groupe d'informations (Table).

Les informations retournées sont des prestations de différents types selon les paramètres renseignés :

#### » Listes des paramètres :

- 
- idCategorie : identifiant de la catégorie de prestation, -1 pour ne pas filtrer par l'idCategorie.
- bSupprime : etat de suppression de la prestation, -1 pas de filtre, 0 non supprimées, 1 les supprimées
- idCommune : identifiant de la commune
- idEtatDisponibilite : etat de disponibilité du jour
- idGroupeCategorie : -1 pour ne pas filtrer par l'idGroupeCategorie
- triEvenement, nb de jour de recherche pour une occurrence ouverte de la prestation
- bMonth1, si une occurrence d'ouverture dans le mois en cours
- bMonth2 à bMonth12 : si une occurrence d'ouverture dans les autres mois
- bWeekEnd : si une occurrence d'ouverture le week end prochain
- idDepartement : identifiant du departement
- idTerritoire : identifiant du territoire
- idAnneeExercice : identifiant de l'année de recherche
- dayBorneInfOuverture : num du jour min de recherche
- dayBorneSupOuverture : num du jour max de recherche
- idSousCategorie : identifiant de sous categorie
- idEtatCaracteristique1 : identifiant d'un etat de caracteristique
- idEtatCaracteristique2 : identifiant d'un etat de caracteristique
- idEtatCaracteristique3 : identifiant d'un etat de caracteristique

#### » Listes des champs retournés :

- idPrestation
- libPrestation
- nomCommune
- fullLibCategorie : nom de la categorie
- etatOuverture : etatDispoFutur (cf front office)
- information : informatino d'ouverture
- idEtatDisponibilite : etat disponibilite du jour
- I2\_CompRemise
- I3\_CompDistrib
- I4\_NumVoie
- I4\_NomVoie
- I5\_LieuDit\_ServDistrib
- I6\_CP
- I6\_Ville
- idDepartement
- idPays
- telephone\_fixe
- telephone\_cellulaire
- fax
- mail
- site\_web
- repere : repere plan
- listPictos : liste des etats des pictos (etatcaracteristiques)
- listSousCategorie : list des sous categories
- triEvenement : nombre de jour avant prochaine ouverture



- detailPrestation : infos XML correspondant aux formulaire dynamique ACVS (dépend de la categorie)

## 7.4. GetFullInfosPrestation

### » Contenu :

Cette fonction permet de récupérer un groupe de données (DataSet) contenant un groupe d'informations (Table).

Les informations retournées sont des prestations de différents types selon les paramètres renseignés :

### » Listes des paramètres :

- idPrestation : identifiant de la prestation demandé, -1 pour ne pas filtrer par l'idPrestation
- idDepartement
- idGroupeCategorie
- idCategorie
- idSousCategorie
- idCommune
- bSupprime
- idTerritoire

### » Listes des champs retournés :

- idStructure :
- codeDepartement :
- nomDepartement :
- idCategorie :
- nomCategorie :
- listSousCategorie :
- nomCommune :
- codeINSEE :
- bSupprime :
- dateMAJ :
- idPrestation :
- nomPrestation :
- apn\_i2\_CompRemise : Elements de l'adresse de la prestation
- apn\_i3\_CompDistrib :
- apn\_i4\_NumVoie :
- apn\_i4\_NomVoie :
- apn\_i5\_LieuDit\_ServDistrib :
- apn\_i6\_CP :
- apn\_i6\_Ville :
- apn\_nomPays :
- apn\_tel\_fixe :
- apn\_tel\_cellulaire :
- apn\_fax :
- apn\_mail :
- apn\_site\_web :
- detailPrestation :
- bInvalidDate :
- ouverture1 :
- ouverture2 :
- dispo1 :
- dispo2 :
- information1 :
- information2 :



- listPictos :
- idPrestataire : identifiant du prestataire
- enseigne : enseigne du prestataire
- raisonSociale : raison sociale du prestataire
- numero\_SIRET : numéro de sret du prestataire
- ape\_I2\_CompRemise : Elements de l'adresse du prestataire
- ape\_I3\_CompDistrib
- ape\_I4\_NumVoie
- ape\_I4\_NomVoie
- ape\_I5\_LieuDit\_ServDistrib
- ape\_I6\_CP
- ape\_I6\_Ville
- ape\_nomPays
- ape\_tel\_fixe
- ape\_tel\_cellulaire
- ape\_fax
- ape\_mail
- ape\_site\_web
- apg\_I2\_CompRemise : Elements de l'adresse du gérant
- apg\_I3\_CompDistrib
- apg\_I4\_NumVoie
- apg\_I4\_NomVoie
- apg\_I5\_LieuDit\_ServDistrib
- apg\_I6\_CP
- apg\_I6\_Ville
- apg\_nomPays
- apg\_tel\_fixe
- apg\_tel\_cellulaire
- apg\_fax
- apg\_mail
- apg\_site\_web

## 7.5. GetCaracteristique

### » Contenu :

Cette fonction permet de récupérer un groupe de données (DataSet) contenant un groupe d'informations (Table).

Les informations retournées sont les caracteristiques du système ACVS

### » Listes des paramètres :

- idCategorie: identifiant de la categorie. -1 pas de filtre

### » Listes des champs retournés :

- idCaracteristiques : identifiant de la caracteristique
- libCaracteristique : libelle de la caracteristique
- idTypeCaracteristique : type de la caracteristique (1 :Label Commun,2 :Label Local,3 : Description Commune,4 :Description Locale)
- usageCaracteristique : usage de la caracteristique

## 7.6. GetEtatCaracteristique

### » Contenu :

Cette fonction permet de récupérer un groupe de données (DataSet) contenant un groupe d'informations (Table).

Les informations retournées sont les etats caracteristiques possible dans le système ACVS



» Listes des paramètres :

- idCaracteristique : identifiant de la caracteristique, -1 pas de filtre

» Listes des champs retournés :

- idEtatCaracteristiques : identifiant de l'état caracteristique
- libEtat : nom de l'état
- idCaracteristiques : identifiant de la carateristique d'origine
- bPubliable : etat de publication
- tri : valeur de tri de l'état dans la caracteristique.

## 7.7. GetCategorie

» Contenu :

Cette fonction permet de récupérer un groupe de données (DataSet) contenant un groupe d'informations (Table).

Les informations retournées sont les categories possédant au moins une prestation en fonctions des paramètres

» Listes des paramètres :

- idCommune : identifiant de la commune, -1 pas de filtre

» Listes des champs retournés :

- idCategorie : identifiant de la categorie
- idGroupeCategorie : identifiant du groupe categorie
- libCategorie : nom court de la categorie
- fullLibCategorie : nom complet de la categorie
- nbPrestation : nombre de prestation dans la categorie en fonction des filtres de la requete

## 7.8. GetSousCategorie

» Contenu :

Cette fonction permet de récupérer un groupe de données (DataSet) contenant un groupe d'informations (Table).

Les informations retournées sont les sous catégories possédant au moins une prestation en fonctions des paramètres

» Listes des paramètres :

- idCommune : identifiant de la commune, -1 pas de filtre

» Listes des champs retournés :

- idSousCategorie: identifiant de la sous categorie
- idCategorie: identifiant de la categorie mère
- libSousCategorie: nom court de la sous categorie
- fullLibSousCategorie: nom complet de la sous categorie
- nbPrestation : nombre de prestation dans la sous categorie en fonction des filtres de la requete

## 7.9. GetCommune

» Contenu :

Cette fonction permet de récupérer un groupe de données (DataSet) contenant un groupe d'informations (Table).

Les informations retournées sont les communes accessibles en fonctions du login utilisé possédant au moins une prestation en fonctions des paramètres



» Listes des paramètres :

- idGroupeCategorie
- idCategorie
- idSousCategorie

» Listes des champs retournés :

- idCommune : identifiant de la commune
- nomCommune : nom de la commune
- nbPrestation dans la commune en fonction des filtres de la requete

## 7.10. Remarques :

▪ La chaîne infoXML ou detailPrestation contient les informations au format XML des formulaires dynamique du SI-ACVS. Sa structure dépend donc de la catégorie de la prestation. Cependant, pour éviter la confusion entre le XML des formulaires dynamiques et celui inhérent à SOAP, le XML des formulaires a utilisé un encodage HTML, à savoir les signes < et > sont remplacés par &lt; et &gt;.

▪ StringOuverture et StringDispo sont des chaînes représentant les informations d'ouvertures et de disponibilité de la prestation sur un an (chaîne de 366 chiffres, le dernier donnant l'information dans le cas d'une année est bissextile). La correspondance des chiffres avec un état est donnée comme suit

» Etat ouverture :

- 1 Indéfini
- 2 Ouvert
- 3 Fermé

» Etat disponibilité :

- 1 Ouvert-Indéfini
- 2 Complet
- 4 Disponible
- 5 Annulé
- 6 Reporté
- 8 Fermé
- 9 Indéfini