Manuel Utilisateur AFS

Version AFS :	6.5
Version de documentation :	1.1
Date :	05/02/2008
Référence :	AFS/DOC/MANUTIL

Table des matières

© Antidot 2007

1. Interroger le moteur de recherche

Une fois installé et configuré, une fois les sources de données indexées, AFS se présente comme un Web Service appelable de façon très simple par des appels HTTP en Get ou en Post. Les paramètres de l'URL servent à indiquer au moteur quel service de recherche est appelé ainsi que tous les paramètres d'appel.

L'appel à AFS peut se faire de deux façons différentes :

- les requêtes vont directement de l'utilisateur au moteur qui a alors la charge de générer une réponse complète et exploitable par le programme client ;
- les requêtes transitent par une application intermédiaire (portail web, application métier) qui récupérera les réponses et pourra à son tour appliquer les traitements nécessaires.

Ces deux modes d'interrogation du moteur peuvent exister simultanément en mettant en place des feuilles de style XSL propres à chaque cas.

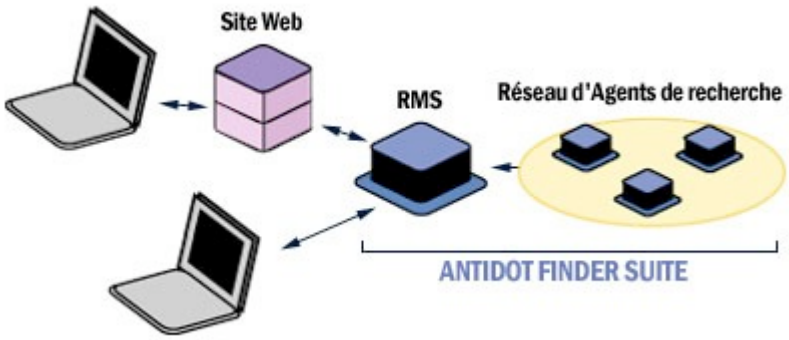


Illustration 1 : modes d'intégration.

Dans les deux cas, les requêtes d'appel sont prise en charge par le logiciel *findall*, un CGI qui aura été préalablement installé sur les frontaux de recherche. Celui-ci qui se charge de transmettre les requêtes au cœur d'AFS, de récupérer et de formater les réponses, et enfin de les renvoyer à l'appelant dans le corps de la réponse HTTP.

Les paramètres d'appel se classent en trois catégories distinctes :

- les paramètres propres à la requête de l'utilisateur ;
- les paramètres propres à AFS ;
- les paramètres spécifiques au service de recherche.

© Antidot 2007

1.1. Les paramètres de la requête utilisateur

Le paramètre KEYWORDS contient les mots que l'utilisateur recherche. Les mots de ce paramètre sont évalués en fonction du mode de recherche par défaut de l'utilisateur.

1.1.1. Principes de base de la recherche par mots-clé

- Pour rechercher les documents contenant le mot A et le mot B, il faudra utiliser la syntaxe suivante : A + B
- Pour rechercher les documents contenant le mot A et pas le mot B, il faut utiliser : A -B
- Pour rechercher les documents contenant le mot A ou le mot B : |A |B
- Pour effectuer une recherche à partir d'une expression, il est nécessaire de mettre celle-ci entre guillemets (ex : «A B»).
- Pour rechercher les mots qui commencent par..., il faut ajouter une astérisque à droite du mot.

  NB : Cette utilisation nécessite que l'option de recherche étendue soit déclarée dans la configuration de réponse.
- Les mots dont le sens n'a que peu d'importance (ex : un, une, le, la, de, des...) sont ignorés.
- La « casse » d'un mot (majuscule ou minuscule) n'est pas prise en compte.
- Les accents et caractères spéciaux (ex : cédilles) sont interprétés selon leur lettre simple équivalente.

1.1.2. Paramètres avancés de recherche par mots-clé

- Les quatre paramètres suivants sont optionnels, et permettent de remplacer `KEYWORDS` :
- Le paramètre `MKEYWORDS` contient des mots dits "obligatoires".
- Le paramètre `OKEYWORDS` contient des mots dits "optionnels".
- Le paramètre `EKEYWORDS` contient des mots dits "exclus".
- Enfin, le paramètre `XKEYWORDS` contient des mots pour recherche en chaîne exacte : il permet d'obtenir le même résultat qu'en entourant de guillemets l'argument `MKEYWORDS` (exemple : `MKEYWORDS=''moteur` donne le même résultat que `XKEYWORDS=moteur de+recherche+`).

L'utilisation de ces paramètres permet de créer plusieurs présentations du service de recherche et par exemple de mettre en place des pages de recherche avancée :

- Le paramètre KEYWORDS permet de créer une zone de saisie unique dans laquelle l'utilisateur précisera le caractère obligatoire ou optionnel des mots recherchés `à l'aide de préfixe.
- Les paramètres MKEYWORDS, OKEYWORDS, EKEYWORDS et XKEYWORDS permettent de créer plusieurs zones de saisie qui déterminent si les mots saisis doivent impérativement ou éventuellement apparaître dans la réponse, ou au contraire, ne doivent pas y figurer, ...

Le paramètre `DEFAULT_MODE` détermine le mode de prise en compte des mots par défaut. Les mots de la requête contenus dans le paramètre `KEYWORDS` qui n'ont pas de préfixe seront considérés comme optionnels ou obligatoires, selon que `DEFAULT_MODE` est égal à `optional` ou `mandatory`.

© Antidot 2007

Nom du paramètre	Description	Valeurs possibles	Valeur par défaut
KEYWORDS	Contenu de la requête	Chaîne de texte au format CGI	Aucune
MKEYWORDS	Mots obligatoires de la requête	Chaîne de texte au format CGI	Aucune
OKEYWORDS	Mots optionnels de la requête	Chaîne de texte au format CGI	Aucune
EKEYWORDS	Mots exclus de la requête	Chaîne de texte au format CGI	Aucune
XKEYWORDS	Recherche chaîne exacte (« MKEYWORDS »)	Chaîne de texte au format CGI	Aucune
DEFAULT_MODE	Caractère obligatoire ou optionnel des mots de la requête	mandatory / optional	

© Antidot 2007

1.2. Les paramètres AFS

Le service de recherche AFS utilise les paramètres d'appel présentés dans le tableau suivant pour répondre correctement à l'utilisateur.

Nom du paramètre	Description
`C`	Identifiant du service de recherche (identifiant client). Obligatoire.
`X`	Numéro du site. Optionnel, vaut `0` par défaut.
`ARG`	Argument de l'agent unique.
`M`	Numéro délivré par l'attribut `Machine` du tag `QueryAgentInformation` du résultat XML.
`MIRROR`	*Paramètre réservé.*
`N`	Numéro de la page de réponse demandée par l'utilisateur.
`OFFSET`	Valeur contenue dans l'attribut `Offset` du tag `Pager` de la réponse XML.
`PAGE`	*Paramètre réservé.*
`SE`	*Paramètre réservé.*
`SERVER`	Numéro de serveur sur lequel la recherche est effectuée. Ce numéro est délivré dans l'attribut `ServerId` du tag `QueryAgentInformation`.
`SITE`	Adresse du site sur lequel la recherche est effectuée. L'adresse du site est fournie par l'attribut URL du tag `QueryAgentInformation`.
`UNIQUE`	Paramètre qui désigne le seul agent qui doit traiter la requête. Exemple: `UNIQUE=user1`
`SORT_ORDER`	Permet de trier les résultats selon un champ ou plusieurs champs et un ordre ascendant ou descendant. `SORT_ORDER=champ|DESC`
`GROUP_BY`	Permet de regrouper les réponses sur les valeurs d'un champ, en limitant le nombre de réponses à remonter par groupe. Exemple, pour regrouper les réponses sur les valeur du champ « `categorie` » en se limitant à 5 réponses par groupe : `GROUP_BY=categorie|5`
`GROUP_BY_NB_GROUPS`	En complément du paramètre GROUP_BY il permet de définir le nombre de groupes que l'on souhaite obtenir par page de résultats.
`GROUP_BY_N`	En mode GROUP_BY, c'est ce paramètre qui permet de passer d'une page (de groupes) à l'autre,
`FILTER_EXPR`	Permet d'appeler le moteur en fixant une contrainte sur les résultats. Ex : `FILTER_EXPR=GT([Prix]; 3000)` permet d'obtenir uniquement les resultats dont le champ « `Prix` » a une valeur supérieure à 3000.
`NB_REPLY`	Permet de spécifier le nombre de réponses renvoyées par page.
`afs:output`	Permet de préciser le format de sortie que AFS doit utiliser pour rendre sa réponse; seule valeur disponible, « `xml` ». Exemple: `afs:output=xml`

Nom du paramètre	Description
`afs:pages`	Permet de filtrer la liste de réponses pour ne ressortir que les réponses dont le numéro interne de page est fourni. Exemple: `afs:pages=888`
`SEED`	Pour certains services, ajoute une part d'aléatoire dans l'ordre de tri des réponses

Le paramètre « `ARG` » est utilisé conjointement avec le paramètre « `UNIQUE` ». Le paramètre « `UNIQUE` » sert à sélectionner un agent unique pour répondre à la requête. Quand il est présent, le paramètre « `ARG` » contient un paramètre utilisé par cet agent. Ces deux paramètres sont propagés au travers des attributs « `Unique` » et « `Arg` » de l'élément « `QueryAgentInformation` » des pages de réponses XML.

Les paramètres « `MIRROR` », « `PAGE` » et « `SE` » sont réservés par Antidot.

Les paramètres « `OFFSET` », « `SERVER` », « `SITE` » et « `M` » proviennent d'informations contenues dans une page de réponses XML. Ils ne doivent être renseignés que lorsqu'ils figurent dans la page de réponses, et que l'utilisateur souhaite poursuivre une recherche en cours (Exemple : l'utilisateur demande la page de réponses suivante).

Le paramètre « `C` » désigne l'identifiant de service unique attribué par Antidot.

Le paramètre « `X` » désigne le numéro du site du service à utiliser. Le numéro de site est attribué par Antidot. Si ce paramètre est absent, les services de recherche utilisent la configuration par défaut du service.

Les paramètres « `SORT_ORDER` » et « `GROUP_BY` » permettent respectivement le tri ou le regroupement des réponses selon un champ de l'index. Pour être placé dans l'index, ce champ doit absolument être déclaré d'une part dans la section `<Store_Items>` du fichier `aixml` et d'autre part dans le fichier de configuration `shmxml`.

Concernant le paramètre « `SORT_ORDER` », il est possible de :

- préciser le sens de tri :
  - `SORT_ORDER=date|ASC` : sens ascendant (cas par défaut si rien n'est précisé)
  - `SORT_ORDER=date|DESC` : sens descendant
- préciser des champs alternatifs (en cas d'égalité sur les champs précédants) :
  - `SORT_ORDER=date|ASC-prix|DESC` : on trie tout d'abord sur la date de façon ascendante, puis, sur égalité de date, on trie sur le prix dans le sens déscendant
- passer des critères de tri spécifiques à la recherche en cours :
  - `afs:relevance` : calcul global de pertinence d'un réponse
  - `afs:weight` : mesure du poids des mots-clé trouvés
  - `afs:pathlen` : mesure de distance entre les mots-clé
  - `afs:nb_words` : nombre de mots trouvés

Concernant le paramètre « `GROUP_BY` », il est possible de :

- limiter le nombre de réponses à remonter par groupe :
  - `GROUP_BY=ID|5` : 5 réponses par groupe

Le paramètre « `FILTER_EXPR` » permet d'appeler le moteur en contraignant les résultats à certaines valeurs.

Ce paramètre attend une expression logique qui peut faire référence à :

- des champs de l'index. Ces champs doivent alors être déclarés dans le fichier de configuration `aixml` et dans le fichier `shmxml`. Les champs sont référencés par leur nom entre crochets. Les champs ne peuvent pas être de type collection, car ces derniers ne renvoient non pas une valeur mais une liste de valeurs.
- des valeurs constantes.
  - Les différent types de données supportés sont les différents types de données qu'il est possible de manipuler dans un index AFS
    - chaine de caractères
    - entier
    - flottant
    - booléen
    - date
  - Pour l'utilisation de chaînes de caractères, il convient de les entourer de simples guillemets (quote). Si la chaîne contient elle même des simples guillemets, il faut les échapper en utilisant le caractère backslash (`\`), qui lui même doit être échappé s'il est utilisé en tant que caractère. Exemple :
    - La chaîne de caractères « `abc` » sera passée au filtre comme : « `'abc'` »
    - La chaîne de caractères « `ab'c` » sera passée au filtre comme : « `'ab\'c'` »
    - La chaîne de caractères « ab\c » sera passée au filtre comme : « `'ab\\c'` »
  - Pour l'utilisation de nombre flottant, le séparateur à utiliser entre partie entière et partie décimale est le point (`.`). Exemple:
    - le nombre « `2,12` » sera écrit dans une expression « `2.12` »
    - le nombre « `100 000,01` » sera écrit dans une expression « `100000.01` »
  - Pour l'utilisation de date, il convient de passer par la fonction « `DATE` » qui construit une date à partir de 3 entiers: année, mois et jour. Exemple: `DATE(2007;03;01)`
- Des fonctions (`EQ`, `GT`, `GE`, `LT`, `LE`, `NOT`, `OR`, `AND`, `IF`, `DATE`, `NOW`, ...); les différents paramètres des focntions sont séparés par un point-virgule (`;`).

Voici la liste des fonctions disponibles :

- **Fonctions de comparaison :**

Attendent exactement deux paramètres qui peuvent être de n'importe quel type (chaîne de caractères, nombres (entier, flottant, booléen) ou des dates

- `EQ` : égal à ; ex : `EQ([Prix]; 1000)`
- `GT` : supérieur à ; ex : `GT([Prix]; 1000)`
- `GE` : supérieur ou égal à ; ex : `GT([Prix]; 1000)`
- `LT` : inférieur à ; ex : `LT([Prix]; 1000)`
- `LE` : inférieur ou égal à ; ex : `LE([Prix]; 1000)`

- **Fonction d'intervalle :**

- `IN` : Entre deux valeurs, bornes comprises. Cette fonction attend 3 paramètres:
  - la valeur à analyser
  - la borne inférieure

- la borne suppérieure

  ex : `IN([Prix]; 500; 1000)`

  ex : `IN([date]; DATE(2007; 01; 01); NOW())`

- **Fonctions logiques :**

- `NOT` : « non » logique ; attend un seul paramètre ;

  ex : `NOT(EQ([Prix]; 1000))`

- `OR` : « ou » logique ; attend au moins un paramètre ;

  ex : `OR(GT([Prix]; 1000)); EQ([Disponibilité]; 'O'))`

- `AND` : « et » logique ; attend au moins un paramètre ;

  ex : `AND(EQ([Disponibilité]; '200'); EQ([Disponibilité]; 'O'))`

- **Fonction de branchement :**

- `IF` : attend 3 paramètres :
  - une valeur booléenne, le test
  - la valeur à renvoyer dans le cas où le test est vrai,
  - puis la valeur à renvoyer si le test est faux.

  ex : `IF(GT(1; [Disponibilité]); 0; 2))`

- **Fonctions de date :**

- `NOW` : renvoie la date du jour (pas de paramètre); ex: `NOW()`
- `DATE` : attend 3 paramètres (3 entiers)
  - l'année
  - le mois
  - le jour du mois

    ex : `DATE(2007; 03; 01)`
- `IS_TODAY` : vérifie si la date fournie pointe la date du jour ; ex: `IS_TODAY([date])`
- `IS_YESTERDAY` : vérifie si la date fournie pointe la date de la veille ; ex: `IS_YESTERDAY([date])`
- `DAY_DIFF` : attend deux paramètres qui sont des dates, calcule le nombre de jours qui sépare ces dates ; ex: `DAY_DIFF([date]; NOW())`
- `WEEK_DIFF` : attend deux paramètres qui sont des dates, calcule le nombre de semaines qui sépare ces dates ; ex: `WEEK_DIFF([date]; NOW())`
- `MONTH_DIFF` : attend deux paramètres qui sont des dates, calcule le nombre de mois qui sépare ces dates ; ex: `MONTH_DIFF([date]; NOW())`
- `YEAR_DIFF` : attend deux paramètres qui sont des dates, calcule le nombre d'années qui sépare ces dates ; ex: `YEAR_DIFF([date]; NOW())`

© Antidot 2007

1.2.1. Possibilité de fixer des valeurs par défaut à des paramètres :

Dans le fichier de configuration de réponse, dans la section `Agents/CGI_Front_End/Service/Parameters` de chaque service, l'option suivante permet de fixer des valeurs par défaut à certains paramètres d'appel du moteur :

```
<Service name="Service">
  <Parameters>
    <Default_Parameter key="SORT_ORDER">prixFILTER|DESC</Default_Parameter>
    <Default_Parameter key="a">a1</Default_Parameter>
    <Default_Parameter key="a">a2</Default_Parameter>
    <Default_Parameter key="a">a3</Default_Parameter>
  </Parameters>
</Service>
```

la valeur par défaut n'est utilisée que si le paramètre n'est pas dans l'URL d'appel
Il est possible de fixer plusieurs valeurs pour un paramètre (dans notre exemple, a a pour valeurs a1, a2 et a3)

1.2.2. Contribution aux statistiques du back-office

Les liens des réponses de la page de réponses, pointent logiquement directement sur l'information désignée. Afin de contribuer à l'enrichissement des statistiques du Back-Office (Hit-Parade), ces liens peuvent désigner le programme CGI d'AFS *redirect*. Son rôle est simplement d'enregistrer le lien de la réponse cliqué par l'internaute et de le rediriger vers l'URL attendue.

Exemple : `cgi-bin/redirect?PARAMS=grippe&C=117&AGENT=user1&TARGET=http://xxxx`

Paramètre	Commentaire
`TARGET`	L'URL cible doit toujours être en dernière position dans la liste des paramètres
`PARAMS`	Les mots cherchés par l'internaute
`AGENT`	Le nom de l'agent qui a donné cette réponse
`C`	Identifiant client AFS

© Antidot 2007

1.3. Les paramètres spécifiques au service

En plus des paramètres standards présentés ci-avant, chaque service de recherche peut définir ses paramètres qui lui sont propres et qui serviront à passer des infos spécifiques lors des requêtes : filtres à appliquer, paramètres de tri, sélection des sources de données, identifiants pour application de règles de sécurité, ...

Le nom des paramètres est composé du nom du filtre (exemple : « `CATEGORY` ») suivi de « `_FILTER` » et il est possible de passer en argument une seule valeur ou une liste de valeurs séparées par des tirets. La valeur des paramètres passés aux filtres est la valeur de l'id des catégories dans le flux XML.

Exemple :

```
<ParameterList>
  <Parameter name="CAT"                value="WEB-PROD"/>
  <Parameter name="FILTER"             value="Editeurs"/>
  <Parameter name="KEYWORDS#PRETTY#"   value="mobile"/>
  <Parameter name="KEYWORDS"           value="mobile"/>
  <Parameter name="F_MANDATORY"        value="mobile"/>
  <Parameter name="F_OPTIONAL"         value=""/>
  <Parameter name="F_EXCLUDED"         value=""/>
  <Parameter name="F_PRETTY_KEYWORDS" value="%2Bmobile"/>
  <Parameter name="CATEGORIE1_FILTER" value="5"/>
</ParameterList>
```

```
<Parametric_Filters>
  <Filter_Param name="CATEGORIE" type="tree">
    <Category id="5" depth="1" nb="4"   label="categorie1"/>
    <Category id="3" depth="1" nb="180" label="categorie2"/>
    <Category id="6" depth="1" nb="18"  label="categorie3"/>
    <Category id="1" depth="1" nb="3"   label="categorie4"/>"
    <Category id="2" depth="1" nb="2"   label="categorie5"/>"
    <Category id="7" depth="1" nb="1"   label="categorie6"/>
  </Filter_Param>
</Parametric_Filters>
```

Chaque filtre peut être invoqué via une contrainte logique (combinaison de `OR`, `AND`, `NOT`) en utilisant le suffixe « `_EXPR` ». Au lieu de donner une unique valeur sur laquelle appliquer le filtre, vous avez la possibilité d'appliquer une combinaoson llogique de valeurs.

Ainsi, pour chaque filtre défini, il est possible de filtrer selon plusieurs valeurs, des valeurs alternatives ou encore de définir des valeurs interdites.

Les valeurs fournies au filtre dans ce cas ne sont pas échapées comme dans le cas de `FILTER_EXPR`; une valeur dans la combinaison de valeurs s'ecrit comme si vous l'aviez ecrite seule, en invoquant le filtre sans l'extension « `_EXPR` ».

Exemples d'appel du moteur :

Exemple	***Commentaire***
`CATEGORIE_FILTER_EXPR=OR(1; 2; 3)`	Ne remonte que les objets qui ont une catégorie égale à 1, ou bien à 2, ou bien à 3. présente différemment, la contrainte vaut: `CATEGORIE_FILTER=1` ou `CATEGORIE_FILTER=2` ou `CATEGORIE_FILTER=3`

Exemple	***Commentaire***
`CATEGORIE_FILTER_EXPR=OR(AND(1; 2); 3)`	Ne remonte que les objets qui ont soit la catégorie 3, soit simultanément la catégorie 1 et 2.
`CATEGORIE_FILTER_EXPR=NOT(OR(1; 2))`	Ne remonte que les objets qui n'ont ni la catégorie 1 ni la categorie 2 – d'un point de vu logique, cette expression est équivalente à `CATEGORIE_FILTER_EXPR=AND(NOT(1); NOT(2))`

Ce module est disponible pour tous les filtres cependant pour pouvoir l'utiliser il doit au préalable être déclaré dans le fichier de configuration de réponse.

NB : Pour des questions de performance, il est préférable de réaliser le maximum de traitement (filtrages, transformation de données, ...) au moment de l'indexation. Par exemple, si on veut uniquement des résultats dont le champ `x` a la valeur `y`, il vaut mieux filtrer les fiches au moment de l'indexation plutôt que filtrer à chaque appel du moteur.

© Antidot 2007

1.4. Exemple d'appels au moteur

```
http://www.monsite.fr/cgi-bin/?C=101&KEYWORDS=match+foot&CATEGORIE1_FILTER=5
```

© Antidot 2007

2. *Installer* une boîte de recherche

Pour qu'un utilisateur puisse interroger le moteur de recherche depuis une page Web, il faut installer sur celle-ci un formulaire HTML qui permette à l'utilisateur de saisir sa requête. Ce formulaire doit transmettre tous les paramètres liés à la requête de l'utilisateur.

2.1. Prérequis

Avant de procéder à l'installation de la boîte de recherche, vous devez connaître l'URL de votre service de recherche ainsi que le numéro du service. Dans le cas où vous avez choisi d'opérer AFS en mode hébergé, ces informations vous sont fournies par le support Antidot.

Exemple :

- URL : `http://www.monsite.fr`
- Numéro AFS d'identification du service de recherche : `C=101`

Boîte de recherche simple

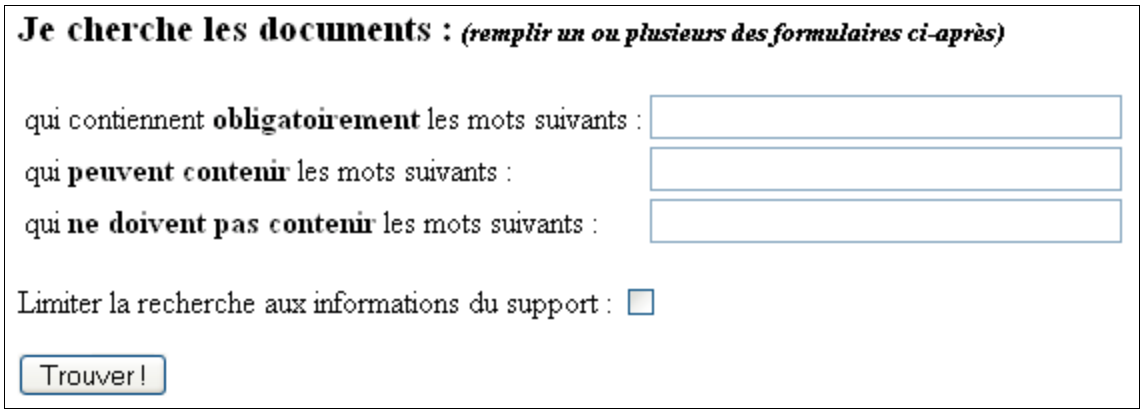
L'exemple ci-dessous montre comment intégrer une boîte de recherche simple sous la forme d'un formulaire HTML inclut dans une page Web. L'utilisateur dispose d'une zone de saisie et peut préciser si les mots doivent être obligatoires ou optionnels par défaut.

```
<FORM action="/cgi-bin/findall">
   <P>Je cherche les documents contenant :</P>
   <SELECT NAME="DEFAULT_MODE">
      <OPTION VALUE="mandatory" SELECTED="true">
      tous ces mots :
      </OPTION>
      <OPTION VALUE="optional">
      une partie de ces mots :
      </OPTION>
   </SELECT>
    
   <INPUT TYPE="text" NAME="KEYWORDS" SIZE="35" MAXLENGTH="150"></INPUT>
   <INPUT TYPE="HIDDEN" NAME="C" VALUE="101"></INPUT>
   <INPUT TYPE="submit" VALUE="Trouver !"></INPUT>
</FORM>
```

2.2. Boîte de recherche avancée

L'exemple ci-dessous montre comment intégrer une boîte de recherche avancée sous la forme d'un formulaire HTML inclut dans une page Web.

© Antidot 2007




```
<FORM NAME="preferences" ACTION="/cgi-bin/findall">
   <H3>Je cherche les documents : <EM><FONT SIZE="-1">(remplir un ou plusieurs
      des formulaires ci-après)</FONT></EM></H3>
   <TABLE BORDER="0">
      <TR>
         <TD>qui contiennent <B>obligatoirement</B> les mots suivants :</TD>
         <TD>
            <INPUT TYPE="text" NAME="MKEYWORDS" SIZE="35" MAXLENGTH="150"></INPUT>
         </TD>
      </TR>
      <TR>
         <TD>qui <B>peuvent contenir</B> les mots suivants :</TD>
         <TD>
            <INPUT TYPE="text" NAME="OKEYWORDS" SIZE="35" MAXLENGTH="150"></INPUT>
         </TD>
      </TR>
      <TR>
         <TD>qui <B>ne doivent pas contenir</B> les mots suivants :</TD>
         <TD>
            <INPUT TYPE="text" NAME="EKEYWORDS" SIZE="35" MAXLENGTH="150"></INPUT>
         </TD>
      </TR>
   </TABLE>
   <P>Limiter la recherche aux informations du support :
      <INPUT TYPE="checkbox" NAME="SUPPORT"></INPUT>
   </P>
   <INPUT TYPE="submit" VALUE="Trouver !"></INPUT>
   <INPUT TYPE="HIDDEN" NAME="C" VALUE="101"></INPUT>
</FORM>
```

2.3. Paramètres passés par Cookie

Les utilisateurs du service de recherche peuvent paramétrer leur préférences qui sont stockées dans des “cookies”.

Le tableau suivant présente la liste des cookies utilisés par le service de recherche AFS :

Nom du Cookie	Description	Valeurs possibles	Valeur par défaut
DEFAULT_MODE	Caractère obligatoire / optionnel des mots de la requête	Mandatory / Optional	Optional
NB_REPLY	Nombre de réponses par pages	10, 20, 30, 40	10

© Antidot 2007

Nom du Cookie	Description	Valeurs possibles	Valeur par défaut
F	Filtre sur les contenus à caractères juridiques (pornographiques, ...)	porno	Aucune
XSLT	Sélection du mode de test XSL	on	Aucune

© Antidot 2007

3. *Générer* des pages de réponses

A l'instar du format utilisé pour les fichiers de configuration, les réponses aux requêtes délivrées par AFS sont en XML. Ce format offre une grande souplesse d'intégration car il peut être manipulé et transformé en n'importe quel autre format : HTML, xHTML, WAP, ou même un autre format XML afin de nourrir une autre application.

3.1. Structure du flux de réponse AFS

Le flux de réponse AFS a la forme suivante :

- le nœud racine `SearchResult` qui possède un attribut `TransactionId` qui indique l'identifiant unique de la requête tel qu'il a été attribué par AFS.
- la balise `ClientInformation` qui récapitule les informations concernant le programme client qui a émis la requête. On retrouve y en autre les paramètres passés de façon transparente par cookie. Les attributs de cette balise sont :
  - `Agent` : le nom du programme client ;
  - `Language` : la langue par défaut telle que configurée par l'utilisateur pour le client ;
  - `NumberOfRepliesPerPage` : le nombre de réponse par page (provenant du cookie `NB_REPLY`) ;
  - `Format` : le format par défaut attendu par le client (ex: HTML) ;
  - `DefaultBehavior` : reprend la valeur du cookie `DEFAULT_MODE` (`Mandatory` ou `Optional`) et indique si l'ensemble des mots recherchés doivent être présents dans les réponses ou si certains sont optionnels.
- la section `QueryAgentInformation` qui reprend les différents paramètres de la recherche : mots clés recherchés, filtres, ... Ces informations sont importantes car elles devront pour la plupart être reprises et à nouveau envoyer au moteur pour avoir les réponses de la page suivante.
- la balise `ReplyDuration` qui indique le temps de réponse à la requête ;
- la section `SearchAgentsResults` qui contient la liste des réponses. Cette balise contient deux attributs qui sont `NumberOfReplies` (nombre de réponses) et `PrettyNumberofReplies` (nombre de réponses présentées une fois les règles d'embellissement réalisées : clusterisation, ...)

Exemple :

```
<SearchResult TransactionId="1831308394">
  <!-- Informations sur le programme client -->
  <ClientInformation Agent="Firefox/1.5.0.3" Language="FRANCAIS"
            NumberOfRepliesPerPage="10" Format="HTML" DefaultBehavior="Mandatory"/>

  <!-- rappel de la requête et de ses paramètres -->
  <QueryAgentInformation QueryString="mobile" CgiQueryString="mobile"
          Machine="681300542"  ServiceId="101" X="" NbReply="10" CgiParams="">
     <ParameterList>
        <Parameter name="CAT" value="WEB-PROD"/>
        <Parameter name="FILTER" value="Editeurs"/>
        <Parameter name="KEYWORDS#PRETTY#" value="mobile"/>
        <Parameter name="KEYWORDS" value="mobile"/>
        <Parameter name="F_MANDATORY" value="mobile"/>
        <Parameter name="F_OPTIONAL" value=""/>
        <Parameter name="F_EXCLUDED" value=""/>
        <Parameter name="F_PRETTY_KEYWORDS" value="%2Bmobile"/>
     </ParameterList>
  </QueryAgentInformation>
```

© Antidot 2007

```
  <!-- le temps de réponse -->
  <ReplyDuration>0,353</ReplyDuration>

  <!-- la liste des réponses -->
  <SearchAgentsResults NumberOfReplies="106" PrettyNumberOfReplies="106">
      <!-- ICI la liste des réponses -->
  </SearchAgentsResults>
</SearchResult>
```

La section principale de ce flux est donc celle dépendant de la balise SearchAgentsResults puisqu'elle contient l'ensemble des réponses et des éléments annexes provenant d'autres modules fonctionnels constituant la solution : Kword, expressions connexes (RTE), filtres et taxonomies de navigation, ...

Les réponses appartenant à un groupe fonctionnel sont regroupées au sein d'une section introduite par la balise `SearchAgentReplies` (il n'y a plus de S à Agent). Cette balise admet deux attributs : `NumberOfReplies` et `NbClustered` (le nombre de réponses et le nombre d'entre elles qui sont clusterisées).

Pour chaque section `SearchAgentReplies`, une balise `ServiceType` indique le type de module fonctionnel dont il s'agit grâce aux attributs `ServiceId` et `ServiceName`. Puis une ou plusieurs entrées de type `SearchAgentReply` listent les éléments de réponse à afficher. Le contenu de ces éléments ainsi que des attributs de la balise de `ServiceType` dépend du module fonctionnel.

Exemple :

```
<!-- les réponses d'un premier module  -->
<SearchAgentsResults NumberOfReplies="106" PrettyNumberOfReplies="106">
  <SearchAgentReplies NumberOfReplies="1" NbClustered="1">
    <!-- il s'agit de données provenant du RTE (expressions connexes) -->
    <ServiceType ServiceId="relexpr" ServiceName="14"/>
    <SearchAgentReply ResultRating="1" Rank="1" ServiceId="relexpr">
      <!-- et ici les différentes expressions -->
  </ SearchAgentReply>
</SearchAgentReplies>

<!-- les réponses d'un deuxième module -->
<SearchAgentReplies NumberOfReplies="83" NbClustered="83">
  <!-- ce module calcule les catégories pour le paramétrique (PNS) -->
  <ServiceType ServiceId="user1" ServiceName="16"/>
  <SearchAgentReply ResultRating="0" Rank="4" ServiceId="user1">
      <!-- et là des données propres au paramétrique -->
  </SearchAgentReply>
</SearchAgentReplies>

<!-- les réponses d'un troisième module -->
<SearchAgentReplies NumberOfReplies="1246" NbClustered="1">
  <!-- il s'agit des réponses principales à afficher -->
  <ServiceType ServiceId="multiple" ServiceName="multiple"/>
  <SearchEngineReply ResultRating="0" NumberOfFoundWords="1" Rank="5"
                     Freshness="normal" Cached="2099" ServiceId="antibot">
      <!-- ici les données pour une réponse -->
  </SearchEngineReply>
  <SearchEngineReply ResultRating="0" NumberOfFoundWords="1" Rank="6"
                     Freshness="normal" Cached="1509" ServiceId="antibot">
      <!-- puis les données d'une autre réponse de la liste -->
   </SearchEngineReply>
   <!-- et ainsi de suite ... Il y a autant de sections SeachEngineReply
        que de réponses à afficher dans la page -->
   </SearchAgentReplies>
</SearchAgentsResults>
```

Les paragraphes suivants détaillent le format des réponses pour chaque type d'agent ou de

module fonctionnel.

3.1.1. Format du flux de réponses provenant de l'agent AntiBot

Les réponses apparaissent par ordre de pertinence avec pour chaque élément de réponse, des informations sur les données indexées (définies dans le fichier `aixml`).

Le flux comprend dans la balise `<Page>` des informations sur l'adresse de la page avec son URL et son Id. Puis dans `<ResourceProperties>` le contenu de la description introduit avec la balise `<Content>`, et du titre avec la balise `<Title>`. Si le mot clé de la requête est compris dans le titre ou la description d'un élément de réponse, celle ci contient alors la balise `<Ew>` qui permet lors de l'affichage la mise en relief du mot clé avec un style de police gras par exemple.

```
<SearchEngineReply ResultRating="0"
                   NumberOfFoundWords="1"
                   Rank="2" Freshness="normal" Cached="138473" ServiceId="antibot">
  <CrawlProperties>
    <Page>
      <URI><!-- ici l'URL de la page --></URI>
      <PageId Id="138473"/> <!-- son id lors de l'indexation -->
    </Page>
    <CrawlSize SizeUnit="Ko">1</CrawlSize>
    <LastCrawlDate> <!-- Date du dernier crawl --> </LastCrawlDate>
  </CrawlProperties>
  <ResourceProperties>
    <Content>xxx xxx <Ew> xxx </Ew> xxx xxx<Etc/> <!-- La description de la reponse  -->
    </Content>
    <Title>xxx xxx <Ew> xxx </Ew>xxx xxx      <!-- Le titre  --></Title>
  </ResourceProperties>
  <Dm><!-- Données issues de l'indexation : numéro du document puis pertinence de la
reponse --></Dm>
</SearchEngineReply>
```

3.1.2. Format du flux de réponses provenant d'un agent XML

Le flux provenant d'un agent XML contient presque les mêmes informations que celui d'un agent antibot :

```
<SearchAgentReply ResultRating="1" Rank="6" ServiceId="user1">
  <XML_Agent Type="user1" Weight="1" Pathlen="1" Found_Words="1">
    <Title><!-- Titre --></Title>
    <URI><!-- URL --></URI>
    <Description><!-- Description --></Description>
  </XML_Agent>
  <Dm>page 7, weight 1, 1 words, pathlen 1</Dm>
</SearchAgentReply>
```

3.1.3. Format du flux de réponses provenant du RTE

Les RTE sont également affichées par ordre de pertinence. Chaque élément `<Related_Expression>` contient la valeur à passer au paramètre `KEYWORDS` dans l'attribut cgi, son score dans l'attribut score et a pour valeur le libellé à afficher.

```
<SearchAgentReply ResultRating="1" Rank="1" ServiceId="relexpr">
  <Related_Expressions nb="17" above="0" threshold="0">
    <Related_Expression score="2.28446">grippe aviaire</Related_Expression>
    <Related_Expression score="1.41421">virus de la grippe</Related_Expression>
    <Related_Expression score="1.41421">campagne de prévention</Related_Expression>
    <Related_Expression score="1.41421">prévention des maladies</Related_Expression>
    <Related_Expression score="1.41421">calendrier vaccinal</Related_Expression>
```

```
    <Related_Expression score="1"      >campagne d information</Related_Expression>
    <Related_Expression score="1"      >personnes âgées</Related_Expression>
    <Related_Expression score="1"      >personnes atteintes</Related_Expression>
  </Related_Expressions>
</SearchAgentReply>
```

3.1.4. Format du flux de réponse provenant du module de suggestion orthographique (hint)

Exemple, sur une recherche « barby », le moteur suggère « baby » - le bloc qui contient cette information est identifié par « `ServiceId="hint"` ».

```
    <SearchAgentReplies NumberOfReplies="1" NbClustered="1" >
      <ServiceType ServiceId="hint" ServiceName="26"/>
      <SearchAgentReply ResultRating="1" Rank="1" ServiceId="hint">
        <Hint>
          <Alt>baby</Alt>
        </Hint>
        <Cgi_Hint>lave+linge</Cgi_Hint>
      </SearchAgentReply>
    </SearchAgentReplies>
```

3.1.5. Format du flux de réponses incluant les filtres paramétriques

Les filtres paramétriques sont introduits par la balise `<Parametric_Filters>`, puis chaque filtre est contenu dans la balise `<Filter_Param>` avec son nom dans l'attribut `name` et son type dans l'attribut `type`. Les filtres peuvent être de type flat avec des valeurs à plat, de type tree, c'est à dire contenir une catégorie qui peut elle même contenir des catégories filles, ou de type interval avec des valeurs qui sont alors des intervalles.

Les filtres de type “`tree`” ou ”flat” sont transmis à l'URL par le paramètre `nomdufiltre_FILTER` qui prend la valeur de l'attribut id. L'attribut label donne l'intitulé de la catégorie pour le restituer lors de l'affichage et nb donne le nombre de réponses correspondant à cette catégorie.

Les filtres de type interval sont transmis à l'URL par les paramètres `MIN_nomdufiltre_FILTER` et `MAX_nomdufiltre_FILTER` qui prennent la valeur de value. Le nombre de réponses correspondant est obtenu par l'addition de `nb_exact` et nb du `Range` suivant moins `nb_exact` de ce même noeud `Range`.

Exemple : L'intervalle 0,75 – 210 contient ((1+3698)-0 =) 3699 réponses.

```
<Parametric_Filters>
  <Filter_Param name="CAT" type="tree">
    <Category id="5" depth="1" nb="4" label="categorie1"/>
    <Category id="3" depth="1" nb="180" label="categorie2"/>
    <Category id="6" depth="1" nb="18" label="categorie3"/>
    <Category id="1" depth="1" nb="3" label="categorie4”/>
    <Category id="2" depth="1" nb="2" label="categorie5”/>
    <Category id="7" depth="1" nb="1" label="categorie6"/>
  </Filter_Param>
  <Filter_Param id="PRIX" type="interval" min="0.75" max="599">
    <Range value="0.75" nb="1" nb_exact="1"/>
    <Range value="210" nb="3698" nb_exact="0"/>
    <Range value="400" nb="32" nb_exact="0"/>
    <Range value="599" nb="5" nb_exact="1"/>
  </Filter_Param>
</Parametric_Filters>
```

© Antidot 2007

3.1.6. Format du flux de réponses provenant de l'agent Kword

L'agent Kword qui permet de mettre en avant une réponse est disponible avec le `serviceId Promotion`. La balise `Title` donne le titre de la réponse . La section OptionReply dont l'id est égal à 1 donne le contenu de la réponse et son URL dans Page/URI et celui dont l'id vaut 2 donne l'URL de l'image de la réponse (également dans Page/URI), ce dernier est facultatif.

```
<SearchAgentReply ResultRating="0" Rank="1" ServiceId="promotion">
  <Title> <!--Titre --> </Title>
  <OptionReply ReplyId="1" ReplyLabel="xxxxx">
    <Page>
      <URI>xxx</URI>
      <PageId Id="0"/>
    </Page>
  </OptionReply>
  <OptionReply ReplyId="2" ReplyLabel="Picture">
      <Page>
        <URI>xxx</URI>
        <PageId Id="0"/>
      </Page>
  </OptionReply>
  <CampaignId>16239</CampaignId>
</SearchAgentReply>
```

3.2. Affichage d'une page de réponse :Utilisation des fichiers Header et Footer et transformation XSLT

5.2.1 Principes de fonctionnement

Le CGI `FINDALL` est chargé de l'affichage des pages de réponses. La page affichée est le résultat d'une transformation XSL des réponses XML d'AFS. Le CGI `FINDALL` est donc obligé d'attendre les réponses du service de recherche avant de pouvoir commencer à afficher la page de réponses. Cela provoque un délai d'attente lors de l'affichage de la réponse pour l'utilisateur. Le principe de l'utilisation d'un Header HTML consiste à afficher l'en tête de la page de réponses avant d'avoir reçu les réponses du service de recherche.

Exemple de Header :

```
<HTML>
  <HEAD>
  <TITLE>Les r´esultats</TITLE>
  <LINK rel="stylesheet" type="text/css"
href="http://www.antisearch.net/css/antisearch.css"/>
<META
HTTP-EQUIV="Content-Type"
CONTENT="text/html; charset=iso-8859-1"/>
<SCRIPT language="JavaScript"
SRC="http://www.antisearch.net/scripts.js">
</SCRIPT>
</HEAD>
<BODY
bgcolor="#ffffff" leftmargin="0"
topmargin="0" marginheight="0" marginwidth="0">
<TABLE
width="770" border="0" bgcolor="#000099"
bordercolor="0" cellpadding="3" cellspacing="0">
<TR>
Manuel Administrateur - Page 4/5
<TD align="left" bgcolor="#071c9d" valign="top" width="203">
```

```
<A href="http://www.antisearch.net">
<IMG src="http://www.antisearch.net/images/antisearch.png"
width="164" height="69" border="0">
</A>
</TD>
</TR>
```

Puis, la transformation XSL va produire le corps de la page de réponses, en se basant sur les informations renvoyées par le service de recherche AFS. Le résultat produit doit être compatible avec les informations HTML déjà publiées dans l'entête de page.

L'utilisation du Header est complétée par celle d'un Footer qui permet de compléter correctement le document HTML. Les balises ouvertes de l'entête HTML doivent être fermées ici.

Exemple d'un Footer associé au fichier Header présenté ci dessus :

```
</TABLE>
</BODY>
</HTML>
```

5.2.2 Utilisation

Le fichier Header est publié dès le lancement du CGI FINDALL. Le fichier Footer est publié après la publication des résultats du service de recherche.
Le fichier Header doit donc contenir toutes les informations nécessaires à l'affichage de l'entête de page HTML. Cependant, l'affichage de cet entête est limité aux informations "statiques" qui ne dépendent pas de la requête. Le fichier Footer doit contenir les informations permettant d'afficher la fin du document HTML. De même, il ne connaît aucune information sur la requête. Les fichiers Header et Footer doivent compléter le résultat publié par les fichiers XSL chargés de la transformation des pages de réponses.

5.2.3. Localisation des fichiers

Les fichiers Header et Footer sont utilisés par les CGI FINDALL. Les CGI FINDALL vont lire ces fichiers par l'intermédiaire du système de fichiers de la machine (et non pas par un accès HTTP). Ils doivent donc être placés dans des répertoires accessibles par les CGI FINDALL sur toutes machines supportant des CGI FINDALL dans le même répertoire. Ils doivent aussi avoir le même nom sur chaque machine (voir le Manuel Administrateur Paragraphe 1.2.3 pour des détails sur la localisation de ces fichiers)

© Antidot 2007

4. *Évaluation* de la pertinence d'un service de recherche

L'ensemble des paramètres présentés dans ce manuel ainsi que ceux décrits dans le manuel d'administration peuvent influer notablement sur votre service de recherche. Compte tenu de l'aspect subjectif de la pertinence, Antidot a mis au point une méthodologie permettant d'évaluer et de comparer différents réglages afin de choisir celui qui convient le mieux.

Tout d'abord, il faut définir une liste de requêtes type :

- entre 20 et 40 requêtes ;
- mixer les requêtes génériques (mots très vagues représentant des concepts) et les requêtes très précises, les requêtes avec un seul mot et plusieurs mots.

Puis, pour chaque requête de la liste, il faut examiner les 10 premières réponses proposées par le moteur et pour chacune, décider de quel type elle est :

- DP : document pertinent par rapport à la question posée ;
- DL : document contenant au moins un lien vers un document pertinent ;
- DB : document banal traitant du sujet demandé, mais ne contenant pas suffisamment d'informations pour répondre à la question posée ;
- DD : document doublon d'une autre page proposée en meilleure position parmi les 10 premiers résultats ;
- DN : document nul correspondant au cas suivants :
  - hors sujet par rapport à la question posée
  - inexistant (Erreur 404) ;
  - site fermé,
  - nouvelle adresse : "cette page a déménagée, cliquez ici...",

Le barème suivant donne les différents scores appliqués aux documents réponses selon leur position parmi les 10 premiers résultats :

Position	***DP***	***DL***	***DB***	***DD***	***DN***
1	+50	-10	-25		-50
2	+25	0	-5	-20	-25
3 ou 4	+12	+2	0	-2	-4
5 à 10	+4	+2	0	-1	-2

On somme alors les scores générés par les dix premières réponses et on obtient une note pour cette requête.

On renouvelle pour chacune des requêtes de la liste et on additionne les notes obtenue, ce qui donne le score final pour un réglage donné.

A chaque modification de réglage, il est possible de renouveler cette procédure d'évaluation et le score obtenu permettra d'évaluer l'amélioration ou la détérioration de la pertinence de façon objective.

© Antidot 2007