



## ACube, Framework Ergonomique

### Spécification Générale des évolutions de la version 2.8.1



Version 1.0 du 25/01/2008

Etat : Validé



## SUIVI DES MODIFICATIONS

Version	Rédaction	Description	Vérification	Date
0.1	K. COIFFET	Initialisation ; Clavier Virtuel	CRO	17/03/2008
1.0	P. BUGAN	Tableau Croisé ; Agenda	CRO	17/03/2008
1.0	A. LESUFFLEUR	Gestion Cache ; Tableau pagination	CRO	17/03/2008
1.0	M. Fraudeau	Composant Aide ; Popup-Inline ; Tableau/Tableur Suivi de processus	CRO	17/03/2008



## SOMMAIRE

<b>SUIVI DES MODIFICATIONS .....</b>	<b>2</b>
<b>SOMMAIRE .....</b>	<b>3</b>
<b>1 OBJECTIFS DU DOCUMENT .....</b>	<b>4</b>
<b>2 SPECIFICATION FONCTIONNELLE GENERALE.....</b>	<b>5</b>
2.1 Clavier virtuel .....	5
2.1.1 Description fonctionnelle.....	5
2.1.2 Mise en œuvre.....	5
2.1.3 Exemple .....	5
2.2 Gestion du cache, versionnage des répertoires.....	5
2.2.1 Description fonctionnelle.....	5
2.2.2 Utilisation .....	6
2.2.3 Exemple .....	6
2.3 Tableaux, tableurs, pagination coté serveur .....	7
2.3.1 Description fonctionnelle.....	7
2.3.2 Utilisation .....	7
2.4 Tableaux, Tableurs edition .....	8
2.4.1 Description fonctionnelle.....	8
2.4.2 Limites .....	8
2.4.3 Utilisation .....	8
2.4.4 Exemple .....	11
2.5 Tableaux, Tableurs personnalisation de l'affichage.....	11
2.5.1 Description fonctionnelle.....	12
2.5.2 Utilisation .....	12
2.5.3 Exemple .....	12
2.6 Tableau Croisé .....	12
2.6.1 Description fonctionnelle.....	12
2.6.2 Utilisation .....	12
2.6.3 Exemple .....	13
2.7 Agenda.....	14
2.7.1 Description fonctionnelle.....	14
2.7.2 Exemple .....	15
2.8 Suivi de processus .....	17
2.8.1 Description fonctionnelle.....	17
2.8.2 Utilisation .....	17
2.8.3 Exemple .....	19
2.9 Popup inline .....	21
2.9.1 Description fonctionnelle.....	21
2.9.2 Utilisation .....	21
2.9.3 Exemple .....	22
2.10 Composant aide .....	22
2.10.1 Description fonctionnelle.....	22
2.10.2 Utilisation .....	22
2.10.3 Exemple .....	23



# 1 OBJECTIFS DU DOCUMENT

Il s'agit dans le cadre de la mise en place des évolutions au sein du Framework Ergonomique ACube de spécifier au travers de ce document le cadre fonctionnel des évolutions mises en place ainsi que leur mode de réalisation au sein de Framework.

Ce document sert donc à la fois de manuel utilisateur à destination des futurs utilisateurs du Framework, mais aussi de référence pour les futures maintenances sur le Framework.

## 2 SPECIFICATION FONCTIONNELLE GENERALE

Ce chapitre permet de définir dans les grandes lignes les nouvelles fonctionnalités mises en place dans le Framework Ergonomique. Il permet à un utilisateur de prendre rapidement connaissance de ces nouvelles fonctions et de leur contexte d'utilisation.

### 2.1 CLAVIER VIRTUEL

#### 2.1.1 DESCRIPTION FONCTIONNELLE

Le clavier virtuel est un composant permettant de protéger la saisie d'un mot de passe.

Ce clavier virtuel est constitué d'un nombre de touches configurable (maximum de 16) et d'un style paramétrable.

#### 2.1.2 MISE EN ŒUVRE

Ce composant nécessite une configuration particulière de la partie Serveur pour fonctionner. (Cf. Doc de fonctionnalités LISE J2EE v2.7.0).

#### 2.1.3 EXEMPLE

Exemple de fichier de configuration pour un clavier virtuel :

```
<PAGE>
  <PARAM_PASSWORD>j_password</PARAM_PASSWORD>
  <STYLE_CLAVIER>STYLE_CV_CIRCLE</STYLE_CLAVIER>
  <NB_ICONS>10</NB_ICONS>
  <PASSWORD_LENGTH>10</PASSWORD_LENGTH>
</PAGE>
```

Exemple d'instanciation du clavier virtuel (sans fichier de configuration XML) :

```
composantCV = new ComposantClavierVirtuel("composantCV", null, "j password",
STYLE_CV_CIRCLE, 13);
```

### 2.2 GESTION DU CACHE, VERSIONNAGE DES REPERTOIRES

#### 2.2.1 DESCRIPTION FONCTIONNELLE

Cette fonctionnalité permet de modifier les noms des répertoires (html | css | jsclient | xml | images | divers) en leur ajoutant un numéro de version.

La mise en cache navigateur des éléments statiques via le module « mod\_expires » Dans des formulaires incluant une quinzaine de JS/CSS du Framework (ex : formulaire avec onglet), cela permet donc de gagner en temps précieux.

Cependant, cela pose le problème des mises à jour applicatives : si l'application est mise à jour sur le serveur et que les fichiers sont déjà en cache sur un poste client, celui-ci ne le saura pas et continuera à travailler avec une interface obsolète.

Pour régler ce problème la solution consiste à faire apparaître le N° de version de l'application dans les répertoires de premier niveau. De cette manière à chaque mise à jour de l'application les chemins seront modifiés et les anciens fichiers mis en cache ne seront plus utilisés.

## 2.2.2 UTILISATION

Pour activer cette fonctionnalité, il faut régler à « true » le paramètre « build.tagDirVersion » présent dans le fichier « project.properties ».

Le paramètre « build.repertoireSpeAddOn » permet de choisir le pattern qui sera ajouté pour le nommage.

Pour configurer la gestion du cache au niveau d'apache, il faut ajouter les lignes suivantes dans le fichier de configuration (httpd.conf) :

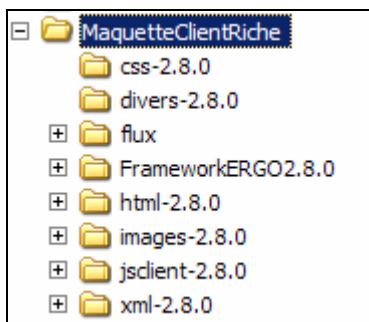
```
JkMount /arobas/flux/* tomcat1
# Depuis la racine de l'appli, le contenu de tous les repertoire du type
# xxxxx-xxxx (ex : "html-1.12" mais pas "html") doivent être mis en cache
<Location ~ "/NomAppli/[^\\/-]+\\-[^\\/]+">
ExpiresActive On
ExpiresDefault A15000000
</Location>
```

## 2.2.3 EXEMPLE

Avec les paramètres suivants :

```
# properties to activate a rename of base directory with tag version
build.tagDirVersion=true
build.repertoireSpeAddOn=-${build.versionLivvable}
```

Les dossiers sont renommés :



NB : Le FrameworkERGO est nommé selon le paramètre : « build.fwergow3cRepertoireInstall ».

## 2.3 TABLEAUX, TABLEURS, PAGINATION COTE SERVEUR

### 2.3.1 DESCRIPTION FONCTIONNELLE

Cette fonctionnalité permet de proposer une pagination dynamique qui est géré coté serveur et non plus coté client. Le flux XML envoyé correspond aux données présentes sur une seule page et non tout le tableau. Le flux XML est actualisé à chaque changement de page (et appelé une fois à la construction). Le flux comportera le nombre total d'item et la liste des items correspondant à la page en cours. Ce flux est appelé en passant en paramètre l'indice du premier item et le nombre d'items par page.

### 2.3.2 UTILISATION

Dans le flux de configuration du tableau, il faut ajouter la balise `<PAGER_SERVER>true</PAGER_SERVER>` avec la valeur à « true » :

```
<PAGER>
  <MAX_LIGNES>20</MAX_LIGNES>
  <SIDE>right</SIDE>
  <POST_NUMBER_REC>true</POST_NUMBER_REC>
  <PAGER_SERVER>true</PAGER_SERVER>
</PAGER>
```

Dans le flux de donnée du tableau, le serveur doit fournir les données pour la page seulement, et indique par la balise `<DATA_LENGTH>nombre_entier</DATA_LENGTH>`, le nombre d'éléments du tableau, la pagination est déduite du nombre d'éléments (`DATA_LENGTH`) et du nombre d'éléments par page (`<MAX_LIGNES>20</MAX_LIGNES>`) :

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<LISTE_ARTICLE>
  <DATA_LENGTH>26</DATA_LENGTH>
  <ARTICLE>
    <DESIGNATION>
      M A R C H E - TITRE DE PERCEPTION ET COPIES -
    </DESIGNATION>
    <UNITE>L006</UNITE>
    <REFERENCE>0325BPL</REFERENCE>
    <PRIX_UNITAIRE>0,20</PRIX_UNITAIRE>
    <TARIFDU>01062006</TARIFDU>
    <CASE>false</CASE>
  </ARTICLE>
  <ARTICLE>
    <DESIGNATION>
      M A R C H E - MANDAT BOITE DE 500 -
    </DESIGNATION>
    <UNITE>BTE</UNITE>
    <REFERENCE>03272PL</REFERENCE>
    <PRIX_UNITAIRE>13,09</PRIX_UNITAIRE>
    <TARIFDU>01062006</TARIFDU>
    <CASE>false</CASE>
```



```
</ARTICLE>  
<LISTE_ARTICLE>
```

## 2.4 TABLEAUX, TABLEURS EDITION

### 2.4.1 DESCRIPTION FONCTIONNELLE

Cette fonctionnalité permet à l'utilisateur d'éditer toutes les lignes d'un tableau simultanément. Via la configuration, chaque colonne peut être en édition ou non. Les cellules d'une même colonne possèdent un type de colonne. Elles possèdent désormais un sous-type [facultatif], et une configuration [facultative] de colonne. Ces deux notions, sous-type et configuration, permettent l'intégration des composants ElementForm (Text & Select) à l'intérieur d'un tableau d'Acube. Un nouveau tag a été rajouté afin de permettre le contrôle des données saisies par l'utilisateur. Enfin, le mode édition permet la sauvegarde en masse des éléments modifiés par celui-ci.

### 2.4.2 LIMITES

Le fonctionnement d'un tableau/tableau dans ce mode sera réduit :

- pas de pagination dans ce mode ;
- pas de tri par colonnes dans ce mode ;
- pas de changement d'ordre des colonnes dans ce mode ;
- pas d'édition des zones utilisées pour les regroupements tableurs ;

### 2.4.3 UTILISATION

#### 1. Colonne éditable

Pour activer l'édition sur une colonne :

```
<COLONNE>  
    <EDITABLE>true</EDITABLE>  
</COLONNE>
```

#### 2. Intégration d'un composant ElementFormSelect

Pour l'activer:

```
<COLONNE>  
    <EDITABLE>true</EDITABLE>  
    <LIBELLE>Unité</LIBELLE>  
    <TYPE>ELEMENTFORMSELECT</TYPE>  
    <TYPECONFIG>  
    <ELEMENT>
```



```

        <LIBELLE>statique:</LIBELLE>
        <OPTIONS>
            <OPTION>
                <LIBELLE>choix1</LIBELLE>
                <VALUE>valeur1</VALUE>
                <SELECTED>FALSE</SELECTED>
            </OPTION>
            <OPTION>
                <LIBELLE>choix2</LIBELLE>
                <VALUE>valeur2</VALUE>
                <SELECTED>FALSE</SELECTED>
            </OPTION>
        </OPTIONS>
    </ELEMENT>
</TYPECONFIG>
</COLONNE>

```

### 3. Intégration d'un composant ElementFormText

```

<COLONNE>
    <EDITABLE>true</EDITABLE>
    <TYPE>ELEMENTFORMTEXT</TYPE>
    <SUBTYPE>NUMBER</SUBTYPE>
    <TYPECONFIG>
        <ELEMENT>
            <LIBELLE>libelle</LIBELLE>
            <VALUE>case à remplir:</VALUE>
        </ELEMENT>
    </TYPECONFIG>
</COLONNE>

```

### 4. Possibilité de surcharge la configuration coté données

Voici un exemple de configuration sans surcharge :

```

<DATAELEMENT>
    <ID>1</ID>
    <COLTEXT>DATA</COLTEXT>
    <COLSELECT>VALUE</COLSELECT>
</DATAELEMENT>

```

Voici un exemple avec surcharge de la configuration:

```

<DATAELEMENT>
    <ID>1</ID>
    <COLTEXT>
        <ELEMENT>

```

```

        <LIBELLE>data</LIBELLE>
        <VALUE>DATA
        </VALUE>
    </ELEMENT>
</COLTEXT>
<COLSELECT>
    <ELEMENT>
        <LIBELLE>data</LIBELLE>
        <OPTIONS>
            <OPTION>
                <LIBELLE>choix1</LIBELLE>
                <VALUE>valeur1</VALUE>
            </OPTION>
            <OPTION>
                <LIBELLE>choix3</LIBELLE>
                <VALUE>valeur3</VALUE>
            </OPTION>
        </OPTIONS>
    </ELEMENT>
</COLSELECT>
</DATAELEMENT>

```

#### 5. Contrôle des saisies.

Lorsqu'une colonne est de type « ELEMENTFORMTEXT » il est peut être judicieux de contrôler les valeurs saisies par l'utilisateur. La balise « CELLVALIDATOR » renseigne le nom d'une fonction implémentée coté projet afin de valider le contenu d'une cellule. Cette méthode prendra en entrée la valeur saisie par l'utilisateur et renverra *true* ou *false* si la donnée est validée, sinon l'ancienne valeur sera remise.

```

<COLONNE>
    <EDITABLE>true</EDITABLE>
    <CELLVALIDATOR>projetjs_validatorCell()</CELLVALIDATOR>
</COLONNE>

```

Voici, coté projet la fonction désignée ci-près :

```

function projetjs_validatorCell() (value) {
    var res = false;
    --
    -- Algo de validation
    --
    return res;
}

```



## 6. Sauvegarde en masse

Stockage des éléments dans un tableau :

```
var myArray = ComposantTableauTest.getArrayModifiedArray();
```

Exemple de parcourt de la table des modifications :

```
var myArray = ComposantTableauTest.getArrayModifiedArray();
var tabCol = ComposantTableauTest.tabColonnes;
for (var ligne=0;ligne<myArray.length;ligne++) {
    var str = "";
    for (var colonne=0;colonne<tabCol.length;colonne++) {
        colId = tabCol[colonne].id;
        if (ComposantTableauTest.getArrayModifiedValues(ligne,
                                                         colId)!='undefined') {
            str+=ComposantTableauTest.getArrayModifiedValues(ligne, colId);
            str+=";";
        }
    }
}
```

## 2.4.4 EXEMPLE

Catalogue des articles				
Désignation		Unité	Référence	PU
data	formatCell = 7- AF	data  --choix3	00051CH	0,05
data	formatCell = 10- A	data  --choix3	00051CH	0,05
data	formatCell = 17- A	statique:  --choix1	00051CH	0,05
data	formatCell = 11- A	statique:  --choix1	00051CH	0,05
data	formatCell = 4- AF	data  --choix3	00051CH	0,05
data	formatCell = 1- AF	data  --choix3	00051CH	0,05

## 2.5 TABLEUX, TABLEURS PERSONNALISATION DE L’AFFICHAGE

## 2.5.1 DESCRIPTION FONCTIONNELLE

Cette fonctionnalité permet de reformater les données affichées par le tableau. On peut ainsi remodeler les données, réduire la taille des données ou ajouter des caractères supplémentaires. Cette fonction, définie dans le projet client, prendra en entrée la donnée et renverra la modifier. Cette fonction sera appelée lors de l'affichage des données dans le tableau.

## 2.5.2 UTILISATION

Voici le paramétrage :

```
COLONNE>
    < FORMAT>projetjs_formatDataCell ()</FORMAT>
</COLONNE>
```

Voici la fonction définie coté projet :

```
function projetjs_formatDataCell ()(value){
    var res = value;
    --
    -- Algo de formattage
    --
    return res;
}
```

## 2.5.3 EXEMPLE



## 2.6 TABLEAU CROISE

### 2.6.1 DESCRIPTION FONCTIONNELLE

Cette fonctionnalité est une extension du composant tableau, elle permet d'afficher un tableau avec un total par colonne et par ligne.

Chaque colonne peut être paramétrée comme « sommable » ou non via le flux de configuration XML.

En mode pagination, la configuration permet d'afficher soit le total d'une colonne de la page courante, soit le sous-total colonne (le total de la colonne de la toute première ligne du tableau jusqu'à la page courante), soit les deux en même temps.

### 2.6.2 UTILISATION

Pour définir une colonne sommable :

```
<COLONNE>
    <SOMMABLE>OUI</SOMMABLE>
</COLONNE>
```

Configuration de la pagination :

```
<PAGE>
  <TABLEAU_CROISE>
    ...
    <PAGER>
      <MAX_LIGNES>15</MAX_LIGNES>
      <SIDE>right</SIDE>
      <POST_NUMBER_REC>true</POST_NUMBER_REC>
      <SHOW_TOTAL_PER_PAGE>true</SHOW_TOTAL_PER_PAGE>
      <SHOW_SUB_TOTAL>true</SHOW_SUB_TOTAL>
    </PAGER>
    ...
  </TABLEAU_CROISE>
</PAGE>
```

- SHOW\_TOTAL\_PER\_PAGE : permet d'afficher le total par colonne de la page courante
- SHOW\_SUB\_TOTAL : permet d'afficher le sous-total colonne

### 2.6.3 EXEMPLE

L'instanciation, hormis le nom de classe, est identique à celle du composant tableau :

```
composantTableauCroise = new
CrossTabComponent("ComposantTableauCroiseTest", XMLInfosTableauCroise, URL XMLDynamiqueCrossTab, true, false);
```

**NB :** Un exemple complet et détaillé est disponible dans la JSDoc (classe CrossTabComponent)

Rendu d'un tableau croisé sans pagination :

Tableau croisé des articles						
Désignation ▲	Unité ◆	Prix 1 ◆	Prix 2 ◆	Prix 3 ◆	Prix 4 ◆	TOTAL
Article 1	F	0,02	10,00	5,89	0,00	15,91
Article 10	F	0,02	0,02	0,02	0,02	0,08
Article 2	F	0,02	17,41	2,11	0,99	20,53
Article 3	F	0,02	110,00	47,08	0,02	157,12
Article 4	F	189,10	18,14	99,00	0,02	306,26
Article 5	F	123,45	543,21	100,00	0,02	766,68
Article 6	F	54,00	0,00	0,00	0,09	54,09
Article 7	F	0,02	0,02	0,02	0,02	0,08
Article 8	F	0,02	0,02	0,02	0,02	0,08
Article 9	F	0,02	0,02	0,02	0,02	0,08
TOTAL		366,69	698,84	254,16	1,22	1 320,91
Pied du tableau croisé						

Rendu d'un tableau croisé avec pagination :

**Tableau croisé des articles**

Désignation	Unité	Prix 1	Prix 2	Prix 3	Prix 4	TOTAL
Article 6	F	54,00	0,00	0,00	0,09	54,09
Article 10	F	0,02	0,02	0,02	0,02	0,08
Article 6	F	54,00	0,00	0,00	0,09	54,09
Article 5	F	123,45	543,21	100,00	0,02	766,68
Article 5	F	123,45	543,21	100,00	0,02	766,68
Article 1	F	0,02	10,00	5,89	0,00	15,91
Article 2	F	0,02	17,41	2,11	0,99	20,53
Article 7	F	0,02	0,02	0,02	0,02	0,08
Article 7	F	0,02	0,02	0,02	0,02	0,08
Article 5	F	123,45	543,21	100,00	0,02	766,68
Article 3	F	0,02	110,00	47,08	0,02	157,12
Article 9	F	0,02	0,02	0,02	0,02	0,08
Article 7	F	0,02	0,02	0,02	0,02	0,08
Article 7	F	0,02	0,02	0,02	0,02	0,08
Article 6	F	54,00	0,00	0,00	0,09	54,09
<b>TOTAL PAGE</b>		<b>532,53</b>	<b>1 767,16</b>	<b>355,20</b>	<b>1,46</b>	<b>2 656,35</b>
<b>SOUS-TOTAL</b>		<b>4 022,12</b>	<b>8 421,97</b>	<b>2 869,97</b>	<b>17,51</b>	<b>15 331,57</b>

Pied du tableau croisé

106 - 120 / 238 8 / 16

## 2.7 AGENDA

### 2.7.1 DESCRIPTION FONCTIONNELLE

Cette fonctionnalité permet de mettre en place un calendrier personnalisé, dans lequel des jours spéciaux peuvent être définis.

Le flux de configuration XML définit les types de jours spéciaux (états) pouvant être associés à un jour précis.

Un état est défini par :

- un id (nombre entier) : permet de relier un jour spécial à un état
- un nom : permet d'afficher un texte alternatif au passage de la souris
- une classe de style CSS : permet un affichage différent des jours spéciaux

Le flux dynamique de données XML définit chaque jour spécial avec l'id de l'état qui lui est associé, mais aussi la fenêtre de définition (en mois) des jours spéciaux définis, c'est cette fenêtre qui permettra de lancer automatiquement la fonction d'approvisionnement des données si jamais l'agenda sort de cette fenêtre.

Une fonction paramétrable est exécutée à chaque clic sur un jour, elle a comme paramètre :

- l'objet Date représentant le jour cliqué
- l'objet AgendaState représentant l'état du jour (null si pas d'état)

Une fonction paramétrable est exécutée lorsque l'agenda sort de sa fenêtre de définition, c'est à cette fonction que la récupération des données dynamiques est confiée. Elle a comme paramètre l'objet Date représentant le mois courant de l'agenda.

Trois types d'approvisionnement sont envisageables :

- pas d'approvisionnement (calendrier classique) : la fonction doit retourner le booléen false
- approvisionnement synchrone : la fonction doit retourner le document XML contenant les données
- approvisionnement asynchrone :
  - la fonction doit retourner le booléen « true »
  - lors du retour de la requête, la fonction « feed » du composant Agenda doit être exécutée avec le document XML contenant les données en paramètres

## 2.7.2 EXEMPLE

Exemple de configuration :

```
<PAGE>
  <AGENDA>
    <STATES>
      <STATE>
        <ID>1</ID>
        <NAME>Jour perso 1</NAME>
        <CSS_CLASS>jour_perso1_css</CSS_CLASS>
      </STATE>
      <STATE>
        <ID>2</ID>
        <NAME>Jour perso 2</NAME>
        <CSS_CLASS>jour_perso2_css</CSS_CLASS>
      </STATE>
    </STATES>
  </AGENDA>
</PAGE>
```

Exemple de données dynamiques :

```
<AGENDA_DATA>
  <PERIOD>
    <!-- du 01/03/2008 au 31/03/2008 -->
    <FROM>
      <YEAR>2008</YEAR>
      <MONTH>03</MONTH>
    </FROM>
    <TO>
      <YEAR>2008</YEAR>
      <MONTH>03</MONTH>
    </TO>
  </PERIOD>
  <DAYS>
    ...
    <!-- le 05/03/2008 sera un jour "Jour perso 1" -->
    <DAY>
      <DATE>05032008</DATE>
```

```

<STATE>1</STATE>

</DAY>
...
</DAYS>
</AGENDA_DATA>

```

Exemple d'instanciation :

```

// Création de l'agenda
var myAgenda = new AgendaComponent("agenda_div_container_id");

// où XMLAgendaConfig est un XMLObject contenant le flux de configuration
myAgenda.configure(XMLAgendaConfig.xmlDoc);

// où customValidator est la fonction exécutée lors d'un clic sur un jour
myAgenda.setDayClickCallBack(customValidator);

// où customProvider est la fonction exécutée lorsque l'agenda sort de sa fenêtre
// de définition
myAgenda.setProvideCallBack(customProvider);

// Affichage de l'agenda
myAgenda.render();

```

Rem : Un exemple complet et détaillé est disponible dans la JSDoc (classe AgendaComponent)

Rendu d'un agenda avec quelques jours spéciaux et plusieurs états personnalisés :





## 2.8 SUIVI DE PROCESSUS

### 2.8.1 DESCRIPTION FONCTIONNELLE

Ce nouveau composant permet de visualiser rapidement les étapes successives (items ou cellules) d'un processus métier et de mettre en valeur leur état d'avancement. Comme les autres composants d'Acube, il possède un fichier XML de configuration et un fichier XML de données.

### 2.8.2 UTILISATION

Configuration du composant processus :

```
<PAGE>
  <PROCESS>
    <STYLE>styleContainer</STYLE>
    <WIDTH>width</WIDTH>
    <DATA_LIST>nom du tag root de la liste</DATA_LIST>
    <DATA_LINE>nom du tag des items</DATA_LINE>
    <MESSAGE_EMPTY>message à afficher quand pas d'items</MESSAGE_EMPTY>
    <CELLS>
      <STYLE>style par défaut des items</STYLE>
      <WIDTH>taille par default des items</WIDTH>
    </CELLS>
  </PROCESS>
</PAGE>
```

<STYLE> : correspond au nom de la classe CSS qu'on applique à la table contenant les cellules étapes.

<WIDTH> : exprimé en pixel, elle correspond à la somme des largeurs des cellules contenues dans la tables.

<DATA\_LIST> : correspond au nœud root des données.

< DATA\_LINE > : correspond au nœud des items.

< MESSAGE\_EMPTY > : le message à afficher lorsqu'il n'y a pas de données dans le suivi de processus.

< CELLS > : correspond à la configuration par défaut des cellules (ou items) du processus métier.

< CELLS : STYLE > : nom de la classe CSS à appliquer lorsqu'elle n'est pas précisée dans les données.

< CELLS : WIDTH > : la valeur de la taille en pixel à appliquer lorsqu'elle n'est pas précisée dans les données.

Fichier des données :

```
<LIST_NODE>
  <ITEM>
    <LABEL>Nom de l'étape dans le processus</LABEL>
    <STATE>état du processus</STATE>
    <ORDER>numéro dans le processus</ORDER>
    <STYLE>style à utiliser pour l'affichage</STYLE>
    <WIDTH>taille de la cellule</WIDTH>
  </ITEM>
</LIST_NODE>
```



< LABEL > : Nom de l'étape dans le processus qu'on affiche dans la deuxième partie de la cellule.

< STYLE > : Le nom de la classe à utiliser pour cette cellule.

< STATE > : Permet la sélection du thème à appliquer à l'entête de la cellule.

< ORDER > : C'est un numéro unique à chaque item qui correspond à l'ordre d'affichage dans la table principale.

< WIDTH > : La valeur de la taille en pixel à appliquer lorsqu'elle n'est pas précisée dans les données.

#### Exemple d'instanciation:

```
//creation d'un object suivi de processus
    var myProcessFollowUp = new ComponentProcessFollowUp("divID");
//Configuration du composant
    myProcessFollowUp.setConfig(XMLStatiquePage);
//Import du fichier des données
    myProcessFollowUp.importData(XMLDataPage);
//Affichage du composant
    myProcessFollowUp.display();
```

#### Création d'un fichier CSS avec les styles définis plus haut:

Le fichier CSS se décompose en deux parties :

- les styles concernant la table principale.
- les styles concernant la table des cellules.

```
//configuration du style css de la table principale
.styleContainer{
    width:900px;
    overflow:auto;
    overflow-x: scroll;
    overflow-y: hidden;
}

//style css du message pas de données
.styleContainer_nodata{
    background-color:    #F5F5F5;
    color: red;
}
```

```
//configuration d'un style de cellule

//configuration de la table de cellule
.style1_master{
```

```
border:2px solid #000000;
border-collapse:collapse;
}
.style1_master_tr{
}
.style1_master_td{
padding:1px;
}

//configuration des thèmes à utilisé pour les cellules.
.style1_table_status1{
background-color: #F5F5F5;
width: 100%;
height: 100%;
}

//css de l'entête des cellules
.style1_head_status1 {
background-color: #FF0000;
color: 000000;
height : 5px;
}

//css du body
.style1_body_status1{
background-color: #F5F5F5;
border-style: none;
color: 000000;
height : 50px;
width: 110px;
text-align : center;
}
```

## 2.8.3 EXEMPLE

### Exemple d'instanciation:

```
//creation d'un objet suivi de processus
var myProcessFollowUp = new ComponentProcessFollowUp("divID");
//Configuration du composant
myProcessFollowUp.setConfig(XMLStatiquePage);
//Import du fichier des données
myProcessFollowUp.importData(XMLDataPage);
//Affichage du composant
myProcessFollowUp.display();
```

### Fichier Css:



```
.styleContainer{
width:900px;
overflow:auto;
overflow-x: scroll;
overflow-y: hidden;
}

.styleContainer_nodata{
background-color: #F5F5F5;
color: red;
}

.style1_master{
border:2px solid #000000;
border-collapse:collapse;
}

.style1_master_tr{
}

.style1_master_td{
padding:1px;
}
```

```
.style1_table_status1{
background-color: #F5F5F5;
width: 100%;
height: 100%;
}

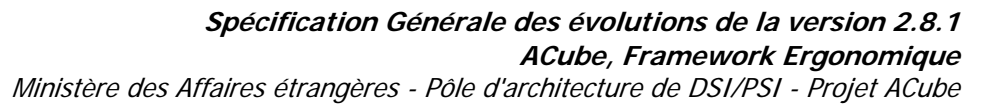
.style1_head_status1 {
background-color: #FF0000;
color: 000000;
height : 5px;
}

.style1_body_status1{
background-color: #F5F5F5;
border-style: none;
color: 000000;
height : 50px;
width: 110px;
text-align : center;
}

.style1_table_status2{
background-color: #F5F5F5;
width: 100%;
height: 100%;
}

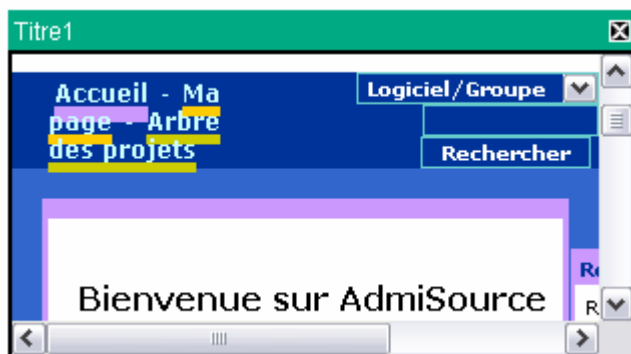
.style1_head_status2 {
background-color: #66CDAA;
height : 5px;
}

.style1_body_status2 {
background-color: #F5F5F5;
```



```
var composant_id= "divid";
var title= "Titre";
var url = "http://admisource.gouv.fr/";
//position
var position_top = 30;
var position_left = 50;
//dimension
var height = size*90;
var width = size*160;
//instance
ComposantPopUpInLine = new ComponentPopupInline (false,composant_id,title,
                                                    url, position_top, position_left, height, width);
//generation du code html
ComposantPopUpInLine.ecrireBind();
//affichage de la popup
ComposantPopUpInLine.display();
```

### 2.9.3 EXEMPLE



## 2.10 COMPOSANT AIDE

### 2.10.1 DESCRIPTION FONCTIONNELLE

Cette évolution permet de disposer de l'aide fonctionnelle d'Acube en plusieurs langues. C'est le système des onglets qui a été retenu. Cette évolution a été l'occasion d'une migration de ce composant vers popup inline pour l'ouverture de l'aide.

### 2.10.2 UTILISATION

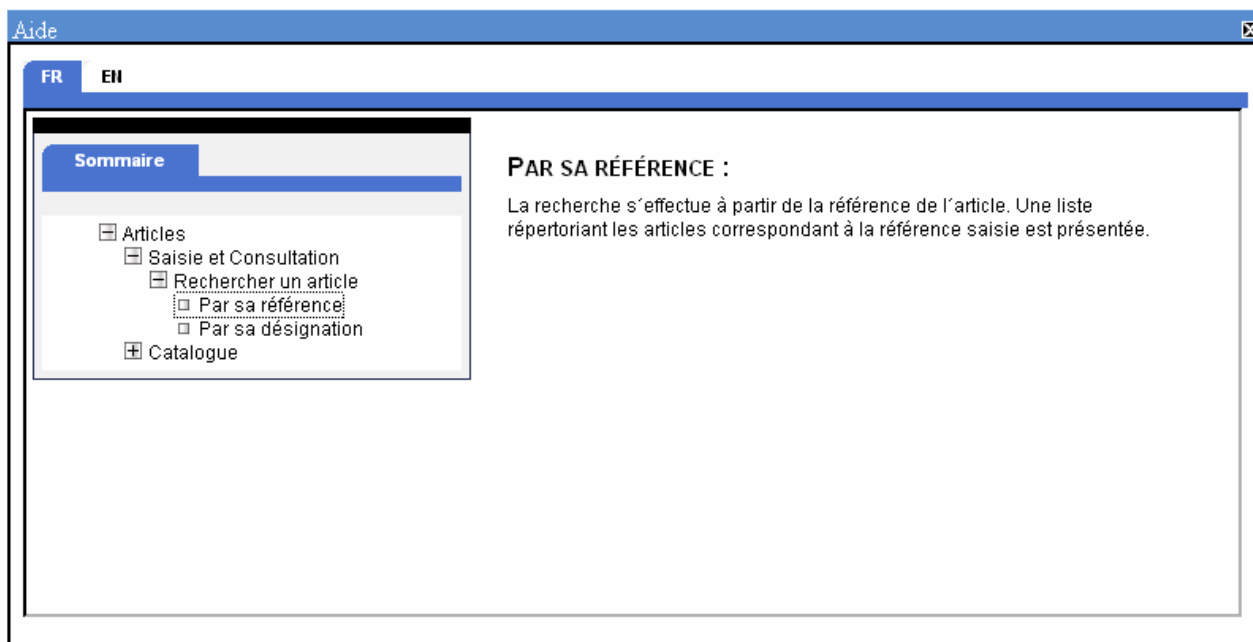
Pour chaque onglet, une langue dans le fichier aide.html :

```
<iframe id="onglet 0" class="ongletProjet ongletcontenu"
src="@URL HTML HtmlAideAideFR?lang=FR&" width="98%" height="310" scrolling="auto"
></iframe>
<iframe id="onglet 1" class="ongletProjet ongletcontenu" style="display:none;"
src="@URL HTML HtmlAideAideEN?lang=EN&" width="98%" height="310" scrolling="auto"
></iframe>
```

Les fonctionnalités du composant aide reste inchangés.

## 2.10.3 EXEMPLE

- l'aide en français



-l'aide en anglais :

