|  |
| --- |
| Actuassur Conseil |
| Script Tool |
| Utilisation et modifications |

|  |
| --- |
| Faizeau Alexis  17/07/2018  Version 1.1.0 |

Sommaires

[Introduction 2](#_Toc520119133)

[A propos 2](#_Toc520119134)

[Usage 2](#_Toc520119135)

[Description des classes PHP 3](#_Toc520119136)

[DAO 3](#_Toc520119137)

[Variables Privées 3](#_Toc520119138)

[Constructeur 3](#_Toc520119139)

[Fonctions privées 3](#_Toc520119140)

[Fonctions publiques 4](#_Toc520119141)

[Opening 6](#_Toc520119142)

[Variables privées 6](#_Toc520119143)

[Constructeur 6](#_Toc520119144)

[Mutateur Magique 6](#_Toc520119145)

[Accesseur Magique 6](#_Toc520119146)

[Fonctions privées 7](#_Toc520119147)

[Fonctions publiques 7](#_Toc520119148)

[Closing 8](#_Toc520119149)

[Variables privées 8](#_Toc520119150)

[Constructeur 8](#_Toc520119151)

[Mutateur Magique 8](#_Toc520119152)

[Accesseur Magique 9](#_Toc520119153)

[Fonctions privées 9](#_Toc520119154)

[Fonctions publiques 10](#_Toc520119155)

[Description de la classe JS Script 12](#_Toc520119156)

[Constructeur 12](#_Toc520119157)

[Fonctions 12](#_Toc520119158)

[Création d’un scenario 15](#_Toc520119159)

[Arborescence 15](#_Toc520119160)

[Structure du fichier opening 15](#_Toc520119161)

[Structure du template 17](#_Toc520119162)

[Structure du script.js 21](#_Toc520119163)

[Structure du closing 22](#_Toc520119164)

# Introduction

## A propos

Le Script Tool est une bibliothèque écrite en PHP 7 et JavaScript ES6 permettant la simplification de la création d’un scenario et de sa maintenance.

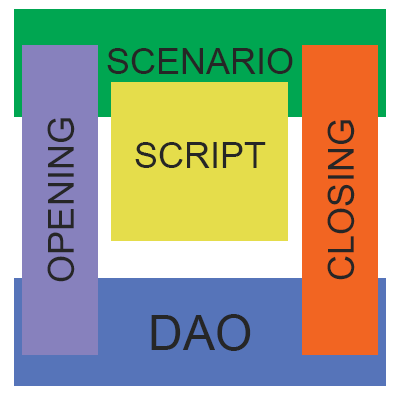
Les différentes classes permettent de gérer automatiquement la gestion du prospect, de la récupération de ses informations à sa qualification et passant par l’envoi en phase 2 si la campagne en possède.

## Usage

La bibliothèque se décompose en plusieurs parties en PHP :

* La DAO : Permet d’accéder aux différentes bases de données
* L’Opening : Traite les données reçues dans l’URL et prépare les informations nécessaires à l’affichage du scenario
* Le Closing : Traite les données envoyées par le scenario, qualifie l’appel effectué et permet d’envoyer le prospect en phase 2 si elle est disponible

En JavaScript il n’y a qu’une classe qui s’appelle Script et permet de gérer l’aspect dynamique du script ainsi que de gérer les variables de temps.



# Description des classes PHP

## DAO

### Variables Privées

#### db\_camp

Objet PDO contenant la base MS Access d’une campagne.

#### db\_campP2

Objet PDO contenant la base MS Access de phase 2 si db\_camp en possède une.

#### db\_admin

Objet PDO contenant la base MS Access adminElastix

#### db\_savprod

Objet PDO contenant la base MS Access savProd

#### db\_webphone

Objet PDO contenant la base MySQL Webphone de 1.30.0.1

### Constructeur

#### Paramètres

* Id\_server : identifiant du serveur Elastix utilisé (présent dans adminElastix dans la table SERVEURS)
* Id\_camp : identifiant de la campagne sur le serveur concerné

### Fonctions privées

#### loadAdmin()

Initialise la variable db\_admin

#### loadSavProd()

Initialise la variable db\_savprod

#### loadWebphone()

Initialise la variable db\_webphone

#### loadCamp(server, id)

Initialise la variable db\_camp et db\_campP2 si la première possède une phase 2.

Paramètres :

* Id\_server : identifiant du serveur Elastix utilisée (présent dans adminElastix dans la table SERVEURS)
* Id\_camp : identifiant de la campagne sur le serveur concerné

### Fonctions publiques

#### getCamp()

Accesseur de db\_camp.

#### getCampP2()

Accesseur de db\_campP2.

#### getAdmin()

Accesseur de db\_admin.

#### getSavProd()

Accesseur de db\_savprod.

#### getWebphone()

Accesseur de db\_webphone.

#### keysToStrPDOUpdate(tab)

Transforme les clés d'un tableau associatif en string pour les requêtes SQL UPDATE en PDO.

Paramètres :

* Tab : Tableau associatif contenant en clé le nom du champ en base de données.

#### keysToStrPDOInsert(tab)

Transforme les clés d'un tableau associatif en string pour les requêtes SQL INSERT en PDO.

Paramètres :

* Tab : Tableau associatif contenant en clé le nom du champ en base de données.

## Opening

### Variables privées

#### Infos

Tableau associatif comportant tous les champs de la table prospect (db\_prospect\_[nom du champs en minuscule] en clé), les informations passées en GET et les variables ajoutées au dictionnaire.

#### dao

Objet DAO permettant l’accès aux bases de données.

### Constructeur

#### Paramètres

* infos : tableau associatif contenant obligatoirement les clés suivantes :
  + agent\_id : identifiant de l’agent
  + serveur\_id : identifiant du serveur
  + campagne\_id : identifiant de la campagne sur le serveur
  + prospect\_id : réfférence du prospect
  + appel\_id : identifiant de l’appel
  + appel\_uniqueid : identifiant unique de l’appel
  + appel\_tel : numéro de téléphone du prospect

### Mutateur Magique

Permet de créer ou de modifier des propriétés de la classe qui sont stockées dans infos.

#### Paramètres

* name : nom de la propriété
* value : valeur à assigner

### Accesseur Magique

Permet d’accéder à des propriétés de la classe qui sont stockées dans infos.

#### Paramètres

* name : nom de la propriété

### Fonctions privées

#### checkInfos()

Vérifie si les propriétés obligatoires à l'initialisation sont bien présentes.

#### getProspect()

Récupère les informations du prospect et les stocke dans la propriété db\_prospect\_[nom du champs en minuscule].

#### getAgent()

Récupère les informations de l’agent :

* agent\_team : identifiant de l’équipe de la team
* agent\_nom : nom complet de l’agent
* agent\_lib : libellé de l’agent
* agent\_sexe : sexe de l’agent
* agent\_type : type de l’agent

#### getPseudo()

Récupère le pseudo de l’agent en fonction de son sexe et de la campagne.

### Fonctions publiques

#### checkExistP2()

Vérifie si le prospect existe en phase 2.

#### getAge(dte\_naiss)

Récupère l’âge millésimé d’une date de naissance.

Paramètres :

* dte\_naiss : date en jj/mm/aaaa

## Closing

### Variables privées

#### isP2

Boolean permettant de savoir s’il s’agit d’une campagne de phase 2 (si oui : true).

#### infos

Tableau associatif comportant tous les champs de la table prospect (db\_prospect\_[nom du champs en minuscule] en clé), les informations passées en GET et les variables ajoutées au dictionnaire.

### Constructeur

#### Paramètres

* isP2 : Permet de savoir si c’est une campagne de phase 2 ou non.
* Infos : tableau associatif contenant obligatoirement les clés suivantes :
  + agent\_id : identifiant de l’agent
  + serveur\_id : identifiant du serveur
  + campagne\_id : identifiant de la campagne sur le serveur
  + prospect\_id : référence du prospect
  + appel\_id : identifiant de l’appel
  + appel\_uniqueid : identifiant unique de l’appel
  + appel\_tel : numéro de téléphone du prospect
  + appel\_timestamp : timestamp du début de l’appel
  + appel\_date : date de l’appel
  + appel\_heure : heure de l’appel
  + resultat\_conclusion : résultat de la qualification
  + resultat\_conclusion\_motifref : motif en cas de qualification en refus
  + resultat\_conclusion\_status : status de la qualification (uniquement si c’est une phase 2)

### Mutateur Magique

Permet de créer ou de modifier des propriétés de la classe qui sont stockées dans infos.

#### Paramètres

* name : nom de la propriété
* value : valeur à assigner

### Accesseur Magique

Permet d’accéder à des propriétés de la classe qui sont stockées dans infos.

#### Paramètres

* name : nom de la propriété

### Fonctions privées

#### checkInfos()

Vérifie si les propriétés obligatoires à l'initialisation sont bien présentes.

#### getProspect()

Récupère les informations du prospect et les stocke dans la propriété db\_prospect\_[nom du champs en minuscule].

#### getAgent()

Récupère les informations de l’agent :

* agent\_team : identifiant de l’équipe
* agent\_nom : nom complet de l’agent
* agent\_lib : libellé de l’agent
* agent\_sexe : sexe de l’agent
* agent\_type : type de l’agent

#### saveResultat(save)

Sauvegarde les données dans la table Resultats.

Paramètres :

* save : tableau associatif contenant en clé le nom du champ en base et en valeur la valeur à inséré.

#### saveProd()

Sauvegarde les données du résultat dans la table Prod.

#### saveRappel()

Sauvegarde les données dans la table Rappels.

#### saveRappelP2()

Sauvegarde les données dans la table Rappels de la phase 2.

#### saveResAppel()

Sauvegarde les données dans la table ResAppel.

### Fonctions publiques

#### save(prospect, resultat)

Sauvegarde toutes les tables.

Paramètres :

* prospect : tableau contenant en clé le nom du champ en base et en valeur la valeur à mettre à jour pour la table Prospect
* resultat : tableau contenant en clé le nom du champ dans la base et en valeur la valeur à insérer pour la table Resultats

#### saveProspect(save)

Sauvegarde les données dans la table Prospect

Paramètres :

* save : tableau associatif contenant en clé le nom du champ dans la base et en valeur la valeur à mettre à jour

#### createResultat(result)

Traitement et création/mise à jour d’un résultats.

Paramètres :

* result : tableau associatif contenant en clé le nom du champ dans la base et en valeur la valeur à insérer

Les champs RefProspect, RefAppel, HOTESSE, DATEAPPEL, HEUREAPPEL, RefQualif, RefCateg, CONCLUSION, MOTIFREFUS, UNIQUE\_ID, RecordFileName sont automatiquement renseignés ! Ils ne sont pas ajoutés à ajouter au paramètres result !

#### getRFN()

Récupère le nom des enregistrements effectués pendant l’appel.

#### checkExistProd()

Vérifie si l’unique id et la refprospect existe déjà dans la table Prod.

#### checkExistRappel()

Vérifie si le prospect existe déjà en rappel.

#### checkExistRappelP2()

Vérifie si le prospect existe déjà en rappel en phase 2.

#### checkExistResAppel()

Vérifie si le resappel existe.

#### checkExistP2()

Vérifie si le prospect est déjà présent en phase 2.

#### getResAppelNum()

Retourne le nombre d’appel dans ResAppel.

#### exportToP2()

Envoi le prospect en phase 2.

Prend toutes les informations de la table Prospect et crée une nouvelle ligne dans la table Prospect en phase 2.

# Description de la classe JS Script

## Constructeur

#### Paramètres

* p2 : vrai si le script possède une phase 2
* curdate : timestamp du début de l’appel généré par le serveur PHP

## Fonctions

#### resetMotifRef()

Désactive l’obligation de saisir un motif de refus.

#### requiredField(field)

Oblige la saisie d’un champ et le colore en rose.

Paramètres :

* field : id d’une balise HTML input

#### unRequiredField(field)

Supprime l’obligation de la saisie d’un champ et le met en blanc.

Paramètre :

* field : id d’une balise HTML input

#### checkAgeMax(field, ageMax, error)

Vérifie si la date de naissance est inférieure à son max sinon on affiche un message d’erreur dans une balise.

Paramètres :

* field : id d’une balise HTML input contenant une date en jj/mm/aaaa
* ageMax : age maximum avant erreur
* error : id d’une balise HTML ou apparaitra l’erreur

#### getAge(dteNaiss)

Récupère l’âge à partir d’une date de naissance.

Paramètre :

* dteNaiss : date au format jj/mm/aaaa

#### checkRadio()

Vérifie si une conclusion a bien été renseignée.

#### setRapTimeMan(field)

Sauvegarde la date et l’heure d’un rappel manuel.

Paramètre :

* field : balise HTML input où doit apparaitre une date heure au format jj/mm/aaaa hh:mm:ss

#### setTimestamp(vDate, vHeure)

Crée un timestamp avec une date et une heure.

Paramètres :

* vDate : date au format jj/mm/aaaa
* vHeure : heure au format hh:mm:ss

#### afficheHeure()

Affiche l’heure dans une balise HTML avec l’id hnow.

#### checkTime(i)

Ajoute un 0 devant une valeur numérique à 1 digit si inférieur à 10

Paramètre :

* i : valeur numérique

#### startTimer()

Affiche le temps de conversation dans une balise HTML possédant l’id tpsConversation.

#### formatDate(pDate)

Formate une date au format jj/mm/aaaa.

Paramètre :

* pDate : Object Date

#### formatHour(pHour)

Formate une heure au format hh:mm:ss.

Paramètre :

* pHour : Objet Date

#### dateNowToScript()

Ecrit la date/heure/timestamp du jour dans des balises input en caché du scenario.

#### getRepondeur()

Valide un script en le qualifiant en répondeur.

# Création d’un scenario

## Arborescence

Un scenario doit toujours être créé dans le répertoire « \\1.30.0.1\www\scenario ».

On crée un nouveau dossier possédant le nom de notre campagne (exemple : sdi9).

Il est composé au minimum de 3 dossiers qui contiennent chacun une partie de notre scenario :

* js :
  + script.js : classe fille de Script
* php :
  + opening.php : traitement avec un objet Opening
  + closing.php : traitement avec un objet Closing
* style :
  + closing.css : style du closing
  + print.css : style utilisé lors de l’impression du scenario
  + style.css : style du scenario

Ainsi que d’un fichier PHP possédant le nom de notre campagne (exemple : sdi9.php) qui contiendra le HTML de notre scenario.

## Structure du fichier opening

On récupère la classe Opening :

*require\_once* '../../addons/ScriptTool/php/Opening.php';

On initialise un objet Opening qui s’appelle opening :

$opening = new Opening(array(

        "agent\_id" => substr($\_GET['agent'],6),

        "serveur\_id" => $\_GET['server'],

        "campagne\_id" => $\_GET['campaign\_id'],

        "prospect\_id" => $\_GET['refprospect'],

        "appel\_id" => $\_GET['call\_id'],

        "appel\_uniqueid" => $\_GET['uniqueid'],

        "appel\_tel" => $\_GET['phone']));

On fait de la réécriture de variable pour avoir des variables plus propres dans notre template :

$opening->prospect\_titre = $opening->db\_prospect\_titre;

$opening->prospect\_nom = $opening->db\_prospect\_nom;

$opening->prospect\_prenom = $opening->db\_prospect\_prenom;

$opening->prospect\_dtenaiss = $opening->db\_prospect\_datenaiss;

$opening->prospect\_nomnaiss = $opening->db\_prospect\_nomnaiss;

$opening->prospect\_lieunaiss = $opening->db\_prospect\_lieunaiss;

$opening->prospect\_dptnaiss = $opening->db\_prospect\_dptnaiss;

$opening->prospect\_email = $opening->db\_prospect\_email;

$opening->prospect\_idpers = $opening->db\_prospect\_idpers;

…

On récupère l’âge du prospect pour pouvoir l’afficher dans le template :

$opening->prospect\_age = $opening->getAge($opening->prospect\_dtenaiss);

## Structure du template

On inclut dans le template le fichier opening :

<?php *require\_once* 'php/opening.php'; ?>

Un head basique contient :

Un titre :

<title><?= $opening->prospect\_nom." : ".$opening->prospect\_id ?></title>

Les styles :

    <link rel="stylesheet" type="text/css" href="style/style.css" />

    <link rel="stylesheet" type="text/css" href="style/print.css" media="print">

JQuery :

    <script type = "text/javascript" src="../../addons/jquery.min.js"></script>

La classe Script et son objet fille :

    <script type="text/javascript" src="/addons/ScriptTool/js/Script.js"></script>

    <script type="text/javascript" src="js/script.js"></script>

L’initialisation de l’objet Script :

    <script>

        var script = new SDI(true, <?= time() ?>);

    </script>

S’il y a besoin de faire des enregistrements il faut inclure la connexion au websocket :

    <script type="text/javascript" src="http://1.30.0.1:3030/socket.io/socket.io.js"></script>

    <script type="text/javascript">

        var socket = io('http://1.30.0.1:3030', {query:"type=0&agentId=<?= $opening->agent\_id ?>"});

    </script>

Et la bibliothèque du recorder :

    <script type = "text/javascript" src="../../addons/recorder/recorder-socket.js"></script>

    <link rel="stylesheet" type="text/css" href="../../addons/recorder/styleRecorder.css" />

Bootstrap si besoin :

    <link rel="stylesheet" type="text/css" href="/addons/bootstrap/css/bootstrap.min.css" />

    <script type = "text/javascript" src="/addons/bootstrap/js/bootstrap.min.js"></script>

DateTime Picker si besoin :

    <link href="/addons/bootstrap-datetimepicker/css/bootstrap-datetimepicker.css" rel="stylesheet" media="screen">

    <script type="text/javascript" src="/addons/bootstrap-datetimepicker/js/bootstrap-datetimepicker.js"></script>

    <script type="text/javascript" src="/addons/bootstrap-datetimepicker/js/locales/bootstrap-datetimepicker.fr.js"></script>

Dans la balise body on initialise la mise à jour de l’heure, on démarre le timer et on récupère l’heure du début de l’appel :

<body onload="setInterval(script.afficheHeure, 500);script.startTimer();script.dateNowToScript();">

Une balise form doit être présente au début pour envoyer les informations au closing et pour vérifier la qualification :

<form action="php/closing.php" onsubmit="return script.checkRadio();" method="POST" id="form\_principal">

Le script doit obligatoirement avoir les deux balises suivantes pour afficher l’heure et le timer :

<span id="hnow"></span>

<span id="tpsConversation">00:00</span>

Pour créer un module d’enregistrement il suffit de mettre ce code HTML en modifiant les paramètre de la fonction JS :

<div id="record">

                            <table class="enregistrer">

                                 <tr>

                                     <th class="enregistrer" id="titrerecorder\_off" colspan=3> Enregistrement </th>

                                 </tr>

                                 <tr>

                                     <td class="enregistrer"><span

                                            id="recorder\_on" onclick="startRecord('<?= $opening->agent\_id ?>',

                                                                                                document.getElementById('res\_nom').value,

                                                                                                '<?= $opening->campagne\_id ?>',

                                                                                                '<?= $opening->prospect\_id ?>',

                                                                                                '<?= $opening->appel\_id ?>',

                                                                                                '<?= $opening->appel\_uniqueid ?>');disableSaveData();">

                                     </span></td>

                                     <td class="enregistrer"><span id="stoprecorder\_off" onclick=""></span></td>

                                     <td class="enregistrer"><span id="timerecorder">00:00</span></td>

                                 </tr>

                            </table>

                        </div>

A la fin du form il faut placer les inputs obligatoires (à part les 3 champs p2 qui ne sont obligatoires que si c’est une phase 1 qui possède une phase 2) au bon fonctionnement du script :

*<!-- CHAMPS CACHES POUR RENSEIGNER LA TABLE RESULTATS -->*

            <div style="display:none">

                <input type = "hidden" name="res\_refprospect" id="res\_refprospect" value="<?= $opening->prospect\_id ?>" /><br/>

                <input type = "hidden" id="res\_appel\_date" name="res\_appel\_date" value=""/><br/>

                <input type = "hidden" id="res\_appel\_heure" name="res\_appel\_heure" value=""/><br/>

                <input type = "hidden" id="res\_appel\_timestamp" name="res\_appel\_timestamp" value=""/><br/>

                <input type = "hidden" id="res\_rappel\_heure" name="res\_rappel\_heure" value=""/><br/>

                <input type = "hidden" id="res\_rappel\_date" name="res\_rappel\_date" value=""/><br/>

                <input type = "hidden" name="res\_rappel\_timestamp" id="res\_rappel\_timestamp" value="" />

                <input type = "hidden" id="res\_p2\_date" name="res\_p2\_date" value=""/><br/>

                <input type = "hidden" id="res\_p2\_heure" name="res\_p2\_heure" value=""/><br/>

                <input type = "hidden" name="res\_p2\_timestamp" id="res\_p2\_timestamp" value="" />

                <input type = "hidden" id="res\_callid" name="res\_callid" value="<?= $opening->appel\_id ?>"/><br/>

                <input type = "hidden" id="res\_uniqueid" name="res\_uniqueid" value="<?= $opening->appel\_uniqueid ?>"/><br/>

                <input type = "hidden" id="res\_serveurid" name="res\_serveurid" value="<?= $opening->serveur\_id ?>"/><br/>

                <input type = "hidden" id="res\_campagneid" name="res\_campagneid" value="<?= $opening->campagne\_id ?>"/><br/>

                <input type = "hidden" id="res\_hotesse" name="res\_hotesse" value="<?= $opening->agent\_id ?>"/><br/>

            </div>

*<!-- FIN DES CHAMPS CACHES -->*

A la fin du body le datetime picker doit être chargée :

*<!-- Démarrage du DateTime Picker -->*

    <script type="text/javascript" src="/addons/bootstrap-datetimepicker/js/bootstrap-datetimepicker-functions.js"></script>

Voici la configuration d’un datetime picker :

*<!-- DatePicker date effet -->*

    <script type="text/javascript">

*// Set des options du datetime picker*

        $('.dteeffet').datetimepicker({

            language: 'fr',

            format: 'dd/mm/yyyy',

            weekStart: 1,

            autoclose: 1,

            todayHighlight: 1,

            pickerPosition: "top-left",

            startView: 2,

            minView:2,

            maxView: 2,

            startDate: <?= '"'.$opening->prospect\_dtefingrat.'"' ?>,

            forceParse: 0,

            showMeridian: 0

         });

    </script>

## Structure du script.js

Ce fichier est une classe fille de la classe Script.js, voici la structure basique :

class Sdi extends Script{

constructor(p2,curdte){

*super*(p2,curdte);

}

}

S’il y a des fonctions à créer c’est dans cette classe qu’elles devront être créées.

## Structure du closing

Dans un premier temps on récupère la classe closing :

*require\_once* '../../../addons/ScriptTool/php/Closing.php';

Puis on initialise l’objet Closing en l’appelant closing :

*/\**

*\* Initialisation du closing*

*\**

*\* Premier paramètre à true si c'est une campagne de phase 2 sinon false*

*\**

*\* Paramètres obligatoires :*

*\* agent\_id,*

*\* serveur\_id,*

*\* campagne\_id,*

*\* prospect\_id,*

*\* appel\_id, appel\_uniqueid, appel\_tel, appel\_timestamp, appel\_date, appel\_heure,*

*\* resultat\_conclusion, resultat\_conclusion\_motifref*

*\*/*

    $closing = new Closing(false, array(

        "agent\_id" => $\_POST['res\_hotesse'],

        "serveur\_id" => $\_POST['res\_serveurid'],

        "campagne\_id" => $\_POST['res\_campagneid'],

        "prospect\_id" => $\_POST['res\_refprospect'],

        "appel\_id" => $\_POST['res\_callid'],

        "appel\_uniqueid" => $\_POST['res\_uniqueid'],

        …

    ));

A ce niveau deux options sont proposés :

* Mettre à jour la table Prospect et Resultats grâce à une seule requête :
  + save(prospect, resultat)
* Mettre à jour la table Prospect et Resultats séparément grâce à deux requêtes :
  + saveProspect(tab)
  + createResultat(tab)

Le résultat sera le même, la fonction save a été séparée en deux pour faciliter certains traitements mais dans la plupart des cas la fonction save sera utilisée (attention les scripts actuels utilisent la syntaxe des deux fonctions ceci n’est pas une marche à suivre dans les nouveaux scripts).

Exemple d’utilisation de la double fonction :

*/\*\**

*\* Met à jour des informations du prospect*

*\*/*

    $closing->saveProspect(array(

        "NumAppel1" => $closing->appel\_tel,

        "lockAgent" => $closing->agent\_id,

        "CONTACT" => $closing->resultat\_contact,

        "CONSENT1" => $closing->resultat\_consent,

        "TITRE" => $closing->prospect\_titre,

        "NOM" => $closing->prospect\_nom,

…

    ));

*/\*\**

*\* Ajout un résultat dans la table Resultats*

*\**

*\* Les champs suivant sont rempli automatiquement :*

*\* RefProspect, RefAppel, HOTESSE, DATEAPPEL, HEUREAPPEL, RefQualif,*

*\* RefCateg, CONCLUSION, MOTIFREFUS, UNIQUE\_ID*

*\*/*

    $closing->createResultat(array(

        "CONTACT" => $closing->resultat\_contact,

        "CHOIXOPTION" => $closing->resultat\_accord\_option,

        "RecordFileName" => $closing->getRFN()

    ));

Enfin si cela est une campagne en 2 temps, on va basculer le prospect en phase 2, mais attention uniquement s’il s’agit d’un accord :

*// On vérifie si c'est bien un accord*

*if*($closing->resultat\_conclusion == "ACCORD"){

*/\*\**

*\* On export tous les champs de Prospect et la date accord,*

*\* hotesse accord et ETACOM vers la base de Phase 2*

*\*/*

        $closing->exportToP2();

    }

Voilà la base de notre script est prête !