

Manipulation de fichiers d'appels et fichiers client Hermes.Net V4.2

Auteur : VOCALCOM
Date : 08/10/2013
Révision : 08/10/2013

1. Descriptif de la note

Cette note décrit le fonctionnement du web service CallFileUtilities.asmx qui offre une collection de fonctions pour la manipulation de fichiers d'appels (importation, recyclage, exclusion, etc ...)

Cette note s'adresse aux développeurs désirant intégrer la gestion des fichiers d'appels de Hermes .Net dans leurs applications.

Les fonctionnalités décrites dans cette note nécessitent Hermes.Net 4.2 à jour au niveau des patches.

2. Généralités

Le web service décrit dans ce document est CallFileUtilities.asmx situé dans le répertoire hermes_net_v4\admin\DataManager_Web_Service\

2.1. Connexion d'un administrateur

L'utilisation du web service nécessite l'utilisation d'un cookie de sessions et une authentification avec un compte administrateur. L'authentification s'effectue avec la fonction :

```
public bool Login(string login, string sha1Password)
```

Paramètres

<i>login</i> :	Login de l'administrateur (vous pouvez utiliser indifféremment un compte administrateur root ou un compte d'administrateur de site)
<i>sha1Password</i> :	mot de passe de l'administrateur encodé en SHA1

Valeur de retour

True si le login et le mot de passe sont acceptés, False sinon.

Exemple en c# :

```
CallFileUtilities ws = new CallFileUtilities();  
ws.Url = "http://server/hermes_net_v4/admin/DataManager_Web_Service/CallFileUtilities.asmx";  
ws.CookieContainer = new System.Net.CookieContainer();  
  
bool result = ws.Login("admin",  
System.Web.Security.FormsAuthentication.HashPasswordForStoringInConfigFile("admin_pwd", "SHA1"));
```

2.2. Mode asynchrone

La grande majorité des fonctions du web service fonctionnent en mode asynchrone : l'appel de la fonction lance une tâche, et il faut ensuite vérifier l'avancement de cette tâche en appelant une seconde fonction. Ce mécanisme permet de ne jamais monopoliser le serveur web trop longtemps et donc de ne pas perturber la production sur le serveur.

La progression d'une tâche peut être consultée en utilisant la fonction suivante :

```
public TaskProgression GetTaskProgression(int taskNumber)
```

Paramètres

taskNumber : Identifiant de la tâche

Valeur de retour

Objet de type `TaskProgression`, indiquant l'état de tâche.

La classe `TaskProgression` a la structure suivante :

```
public class TaskProgression
{
    public TaskResult Result { get; set; }
    public int Progression { get; set; }
    public string Detail { get; set; }
}
```

Result : Énumération indiquant l'état de la tâche, les valeurs possibles sont :

Pending :	tâche non démarrée, en attente qu'une autre tâche se termine
InProgress :	tâche en cours de traitement
CompletedOK :	tâche complétée, sans erreur
CompletedWithError :	tâche complétée, avec des erreurs
ErrorCampaignNotFound :	tâche abandonnée car campagne demandée non trouvée
ErrorNoCallFile :	tâche abandonnée car fichier d'appel non trouvé
ErrorNoClientFile :	tâche abandonnée car fichier client non trouvé
ErrorBadQueryFormat :	tâche abandonnée car erreur dans la requête donnée
ErrorTaskNotFound :	tâche demandée non trouvée
ErrorBadParameter :	tâche abandonnée car un des paramètres donné est invalide
ErrorTaskQueueFull :	tâche abandonnée car trop de tâches en attente d'exécution
ErrorUnauthenticated :	tâche abandonnée car la fonction Login n'a pas été utilisée
ErrorCustomerNotAllowed :	tâche abandonnée car tentative de manipuler des campagnes d'un site auquel le compte administrateur utilisé n'a pas accès.
ErrorUnknown :	tâche abandonnée suite à une erreur non répertoriée

Progression : Indique le pourcentage d'actions actuellement effectuées par rapport aux nombres d'actions totales prévues. Par exemple pour une tâche de recyclage, indique le pourcentage de fiches exclues par rapport au nombre de fiches total à exclure. On obtient donc une valeur entre 0 et 100.

Detail : En cas d'erreur, donne un texte explicatif donnant des précisions sur l'erreur rencontrée.

Des exemples d'utilisation seront donnés dans la suite de ce document.

Note : Il ne peut pas y avoir plus de 50 tâches lancées en simultanée. La demande d'une tâche supplémentaire lorsque ce nombre est atteint provoque l'erreur « `ErrorTaskQueueFull` » décrite ci-dessus.

2.3. Syntaxe des conditions de sélection

Certaines fonctions demandent de préciser une condition permettant de ne manipuler qu'une partie d'un fichier d'appels (par exemple pour une tâche de recyclage, ou d'exclusion, etc.)

Les conditions sont au format SQL, le nom de table « **CallFileTable** » correspond à la table du fichier d'appels et « **ClientTable** » à la table principale du fichier client.

Exemples :

Sélection des fiches dont l'indice est entre 100 et 200 :

```
CallFileTable.INDICE > 100 AND CallFileTable.INDICE < 200
```

Sélection des fiches qualifiées avec le code 1 dans les départements 75 ou 92 :

```
CallFileTable.STATUS = 1 AND ClientTable.POSTAL_CODE IN (75, 92)
```

2.4. Liste des fiches modifiées par une tâche.

Pour les tâches modifiant des fiches d'un fichier d'appel (recyclage, exclusion, etc.), il est possible - une fois la tâche terminée - de consulter la liste des fiches modifiées en utilisant la fonction :

```
public long[] GetTaskModifiedIndices(int taskNumber)
```

Paramètres

taskNumber : Identifiant de la tâche

Valeur de retour

Liste des indices des fiches modifiées

2.5. Liste des fiches non importées.

Si une tâche d'importation (dans un fichier d'appel ou un fichier client) se termine avec l'état « CompletedWithError », la fonction suivante permet d'obtenir la liste des erreurs rencontrées :

```
public ImportationError[] GetImportationErrors(int taskNumber)
```

Paramètres

taskNumber : Identifiant de la tâche

Valeur de retour

Liste d'objets de type `ImportationError` donnant des informations sur les erreurs rencontrées.

La classe `ImportationError` a la structure suivante :

```
public class ImportationError
{
    public string Destination { get; set; }
    public object[] Data { get; set; }
    public string Reason { get; set; }
}
```

Destination : Indique la destination prévue des données. Peut valoir :

- « CALLFILE » : erreur d'écriture dans le fichier d'appel
- « CLIENT » : erreur d'écriture dans le fichier client
- « CALLBACK » : erreur lors de l'enregistrement d'un rappel personnel (en mode Avaya)

Data : Tableau contenant les données qui devaient être importées.

Reason : Raison de l'erreur, contient le texte de l'exception qui est survenue ou le texte de l'erreur retournée par Sql Server ou Oracle.

3. Fonctions de manipulation de fichiers d'appels

3.1. Exclusion de fiches

Pour lancer une exclusion de fiches, il faut utiliser la fonction suivante :

```
public int ExcludeCalls(int customerId, string campaignId, string condition, string reason)
```

Paramètres

<i>customerId</i> :	Identifiant du site sur lequel se trouve la campagne.
<i>campaignId</i> :	ID d'une campagne d'émission d'appels.
<i>condition</i> :	Condition de sélection des fiches à modifier. Si null ou « » alors modification de toutes les fiches.
<i>reason</i> :	Raison de l'exclusion. Sera précisée dans le champ HISTORIQUE du fichier d'appel.

Valeur de retour

Identifiant de la tâche.

Note : quelle que soit la condition précisée, la fonction n'exclue que les fiches qui peuvent encore être appelées.

C'est-à-dire les fiches respectant la condition : `CallFileTable.PRIORITE >= 0`

Exemple en c# :

```
// voir chapitre 2.1 concernant la connexion au compte d'administration

int taskNumber = ws.RecycleCalls(1, "myCampaignId", "ClientTable.POSTAL_CODE IN (75, 92)", "");

CallFileUtilities.TaskProgression taskResult;
do
{
    System.Threading.Thread.Sleep(500);
    taskResult = ws.GetTaskProgression(taskNumber);
}
while (taskResult.Result == CallFileUtilities.TaskResult.Pending
    || taskResult.Result == CallFileUtilities.TaskResult.InProgress);
```

3.2. Inclusion de fiches

Pour lancer une exclusion de fiches, il faut utiliser la fonction suivante :

```
public int IncludeCalls(int customerId, string campaignId, string condition)
```

Paramètres

<i>customerId</i> :	Identifiant du site sur lequel se trouve la campagne.
<i>campaignId</i> :	ID d'une campagne d'émission d'appels.
<i>condition</i> :	Condition de sélection des fiches à modifier. Si null ou « » alors modification de toutes les fiches.

Valeur de retour

Identifiant de la tâche.

Note : quelle que soit la condition précisée, la fonction n'inclue que les fiches précédemment exclues. C'est-à-dire les fiches respectant la condition : `CallFileTable.PRIORITE IN (-10, -11)`

3.3. Mélange de fiches

L'option de mélanges de fiches permet de changer la valeur du champ `MIXUP` du fichier d'appel qui détermine l'ordre dans lequel seront numérotées les fiches. Il est possible de modifier la valeur de ce champ `MIXUP` en utilisant la fonction :

```
public int ChangeCallsOrder(int customerId, string campaignId, string condition, int order)
```

Paramètres

<i>customerId</i> :	Identifiant du site sur lequel se trouve la campagne.
<i>campaignId</i> :	ID d'une campagne d'émission d'appels.
<i>condition</i> :	Condition de sélection des fiches à modifier. Si null ou « » alors modification de toutes les fiches.
<i>order</i> :	Valeur du champ MIXUP.

Valeur de retour

Identifiant de la tâche.

Note : quelle que soit la condition précisée, la fonction ne permet de modifier que les fiches qui peuvent encore être appelées et pour lesquelles aucun rappel n'est programmé. C'est-à-dire les fiches respectant la condition : `CallFileTable.PRIORITE > 0`

3.4. Recyclage de fiches

Pour lancer un recyclage de fiches, il faut utiliser la fonction suivante :

```
public int RecycleCalls(int customerId, string campaignId, string condition, bool cleanHisto)
```

Paramètres

<i>customerId</i> :	Identifiant du site sur lequel se trouve la campagne.
<i>campaignId</i> :	ID d'une campagne d'émission d'appels.
<i>condition</i> :	Condition de sélection des fiches à modifier. Si null ou « » alors modification de toutes les fiches.
<i>cleanHisto</i> :	Indique si on veut supprimer l'historique des appels, c'est-à-dire si on veut remettre à zéro les champs HISTORIQUE , NIVABS et NBAPPELS

Valeur de retour

Identifiant de la tâche.

Note : quelle que soit la condition précisée, la fonction ne permet de recycler que les fiches qui ne sont ni exclues, ni black-listées.

C'est-à-dire les fiches respectant la condition : **CallFileTable.PRIORITE NOT IN (-10,-11,-12) AND LIB_STATUS<>'DNCL'**

4. Fonctions de manipulation des données

4.1. Importation dans un fichier client

Pour importer des données dans un fichier client, il faut utiliser la fonction suivante :

```
public int AddClients(int customerId, string campaignId, string[] fields, object[][] values,
    bool addToCallFile, string[] phoneFields, string memoField, bool excludeCalls,
    string excludeReason, int order, int agentCode, DateTime callbackHour)
```

Paramètres

customerId : Identifiant du site sur lequel se trouve la campagne.
campaignId : ID d'une campagne d'émission d'appels.
fields : Liste des colonnes du fichier client que l'on veut renseigner.
values : Liste des lignes à écrire dans le fichier client. Pas plus de 500 lignes.
addToCallFile : Indique si on veut également ajouter ces fiches dans le fichier d'appels.

Les paramètres suivants sont ignorés si addToCallFile=false :

phoneFields : Liste des colonnes du fichier client contenant des numéros de téléphone. La première colonne sera utilisée pour TEL1, la seconde pour TEL2, etc.
memoField : Colonne du fichier client contenant le mémo à écrire dans le fichier d'appels.
excludeCalls : Indique si on veut exclure les fiches importées dans le fichier d'appels.
excludeReason : Indique la raison de l'exclusion. (Sera précisée dans le champ HISTORIQUE)
order : Valeur du champ MIXUP
agentCode : Permet d'affecter la fiche à un agent. Mettre 0 pour ne pas affecter la fiche.
callbackHour : Permet de programmer un rappel pour les fiches ajoutées. Mettre DateTime.MinValue pour ne pas fixer de rappels.

Valeur de retour

Identifiant de la tâche.

Exemple en c# :

```
// champs du fichier client :
string[] clientFields = new string[] { "NAME", "AGE", "CITY", "PHONE", "MOBILE" };
// champs contenant des numéros de téléphone :
string[] phoneFields = new string[] { "PHONE", "MOBILE" };

// données du fichier client
List<object[]> data = new List<object[]>();
data.Add(new object[] { "Dupont", 54, "Paris", "0155373050", "06xxxxxxx" });
data.Add(new object[] { "Durand", 27, "Rouen", "xxxxxxxxxx", "" });
data.Add(new object[] { "Lefevre", 33, "Marseille", "xxxxxxxxxx", "06xxxxxxx" });

// importation dans fichier d'appel + fichier client, sans exclusion ni programmation de rappel
int taskNumber = ws.AddClients(1, "mycampaignId", clientFields, data.ToArray(), true, phoneFields,
"CITY", false, "", 0, 0, DateTime.MinValue);
```

Note : Si vous avez beaucoup de lignes à importer, il faut faire plusieurs appels à la fonction AddClients en ne donnant qu'un maximum de 500 lignes à chaque appel. Vous n'êtes pas obligé d'attendre la fin d'une tâche d'importation avant d'en demander une autre, mais il ne faut pas perdre de vue la limitation de 50 tâches simultanées.

4.2. Mise à jour d'un fichier client et/ou d'un fichier d'appels

Pour mettre à jour des données dans un fichier client, il faut utiliser la fonction suivante :

```
public int UpdateClient(int customerId, string campaignId, long indice,
    string[] callFileFields, string[] callFileValues, string[] clientFields,
    object[] clientValues)
```

Paramètres

<i>customerId</i> :	Identifiant du site sur lequel se trouve la campagne.
<i>campaignId</i> :	ID d'une campagne d'émission d'appels.
<i>indice</i> :	indice de la fiche à modifier
<i>callFileFields</i> :	Liste des colonnes à modifier dans le fichier d'appels.
<i>callFileValues</i> :	Liste des valeurs à modifier dans le fichier d'appels.
<i>clientFields</i> :	Liste des colonnes à modifier dans le fichier client.
<i>clientValues</i> :	Liste des valeurs à modifier dans le fichier client.

Valeur de retour

Identifiant de la tâche.

Notes :

- Les listes `callFileFields` et `callFileValues` doivent contenir le même nombre d'éléments.
- Les listes `clientFields` et `clientValues` doivent contenir le même nombre d'éléments.
- Si le fichier d'appel modifié n'est pas associé à un fichier client, ou si vous ne voulez pas modifier le fichier client, mettre la valeur null pour les paramètres `clientFields` et `clientValues`.
- Si vous ne voulez pas modifier les données du fichier d'appels, mettre la valeur null pour les paramètres `callFileFields` et `callFileValues`.
- Seuls les champs **DATAMEMO** et de **TEL1** à **TEL10** peuvent être modifiés dans le fichier d'appels.

Exemple en c# :

```
// modification pour la fiche d'indice 18952 du mémo dans le fichier d'appel
// et des colonnes AGE et CITY du fichier client

ws.UpdateClient(1, "myCampaignId", 18952,
    new string[] { "DATAMEMO" }, new string[] { "new memo" },
    new string[] { "AGE", "CITY" }, new object[] { 12, "Paris" });
```

4.3. Suppression de clients

Pour supprimer des lignes du fichier client, il faut utiliser la fonction suivante :

```
public int DeleteClients(int customerId, string campaignId, string condition)
```

Paramètres

customerId : Identifiant du site sur lequel se trouve la campagne.
campaignId : ID d'une campagne d'émission d'appels.
condition : Condition de sélection des fiches à supprimer.
Si null ou « » alors suppression de toutes les fiches.

Valeur de retour

Identifiant de la tâche.

Note : cette fonction supprime également les fiches correspondantes dans le fichier d'appel.
Il est possible de consulter la liste des fiches supprimées en appelant la fonction `GetTaskModifiedIndices`.

4.4. Importation dans un fichier d'appels.

Pour importer des données dans un fichier d'appels à partir d'un fichier client, il faut utiliser la fonction suivante :

```
public int AddCalls(int customerId, string campaignId, string condition, string[] phoneFields,
string memoField, bool excludeCalls, string excludeReason, int order, int agentCode,
DateTime callbackHour)
```

Paramètres

customerId : Identifiant du site sur lequel se trouve la campagne.
campaignId : ID d'une campagne d'émission d'appels.
condition : Condition de sélection des fiches à importer.
Si null ou « » alors importation de toutes les fiches.
phoneFields : Liste des colonnes du fichier client contenant des numéros de téléphone.
La première colonne sera utilisée pour **TEL1**, la seconde pour **TEL2**, etc.
memoField : Colonne du fichier client contenant le mémo à écrire dans le fichier d'appels.
excludeCalls : Indique si on veut exclure les fiches importées dans le fichier d'appels.
excludeReason : Indique la raison de l'exclusion.(Sera précisée dans le champ **HISTORIQUE**)
order : Valeur du champ **MIXUP**
agentCode : Permet d'affecter la fiche à un agent. Mettre 0 pour ne pas affecter la fiche.
callbackHour : Permet de programmer un rappel pour les fiches ajoutées. Mettre **DateTime.MinValue** pour ne pas fixer de rappels.

Valeur de retour

Identifiant de la tâche.

Note : quelle que soit la condition précisée, la fonction ne pourra importer que les fiches qui ne sont pas déjà présentes dans le fichier d'appel.
C'est-à-dire les fiches respectant la condition : **ClientTable.Indice NOT IN (SELECT Indice FROM CallFileTable)**

Il est possible de consulter la liste des fiches importées en appelant la fonction `GetTaskModifiedIndices`.
Les paramètres de cette fonction sont sensiblement les mêmes que ceux de la fonction `AddClients`, vous pouvez donc consulter l'exemple donné dans le chapitre 4.1

4.5. Suppression de lignes d'un fichier d'appels.

Pour supprimer des lignes d'un fichier d'appels, il faut utiliser la fonction suivante :

```
public int DeleteCalls(int customerId, string campaignId, string condition)
```

Paramètres

customerId : Identifiant du site sur lequel se trouve la campagne.
campaignId : ID d'une campagne d'émission d'appels.
condition : Condition de sélection des fiches à supprimer.
 Si null ou « » alors suppression de toutes les fiches.

Valeur de retour

Identifiant de la tâche.

4.6. Enregistrement ou suppression de rappels.

Pour enregistrer ou supprimer des rappels, il faut utiliser la fonction suivante :

```
public int UpdateCallbacks(int customerId, string campaignId, string condition,
    DateTime callbackHour, string phoneField, int agentCode)
```

Paramètres

customerId : Identifiant du site sur lequel se trouve la campagne.
campaignId : ID d'une campagne d'émission d'appels.
condition : Condition de sélection des fiches à modifier.
 Si null ou « » alors modification de toutes les fiches.
callbackHour : Heure du rappel ou `DateTime.MinValue` pour supprimer un rappel existant.
phoneField : Colonne du fichier d'appel à utiliser pour programmer le rappel.
agentCode : Permet d'affecter les rappels à un agent. Mettre 0 pour ne pas affecter le rappel, mettre -1 pour ne pas modifier l'affectation actuelle.

Valeur de retour

Identifiant de la tâche.

Pour supprimer des rappels : mettre `callbackHour=DateTime.MinValue`, `phoneField=null` et `agentCode=0`.

Seules les fiches (exclues ou non) pour lesquelles il y a un rappel de programmé seront modifiées, c'est-à-dire les fiches respectant la condition : `CallFileTable.PRIORITE IN (0, -11)`

Pour modifier la date de rappels existants : renseigner `callbackHour` avec une date valide et mettre `phoneField=null`.

Les valeurs possibles de `agentCode` sont : 0 pour supprimer une affectation, -1 pour ne pas modifier l'affectation existante, ou une valeur de code agent pour affecter les rappels à un agent donné.

Seules les fiches (exclues ou non) pour lesquelles il y a un rappel de programmé seront modifiées, c'est-à-dire les fiches respectant la condition : `CallFileTable.PRIORITE IN (0, -11)`

Pour programmer des rappels (et modifier des rappels existants) : renseigner `callbackHour` avec une date valide et renseigner `phoneField` avec le nom d'une colonne du fichier d'appel contenant un numéro de téléphone (de `TEL1` à `TEL10`).

Les valeurs possibles de `agentCode` sont : 0 pour ne pas affecter les rappels, ou une valeur de code agent pour affecter les rappels à un agent donné.

Toutes les fiches pouvant encore être appelées (exclues ou non, avec ou sans rappels), ainsi que les fiches déjà qualifiées, seront modifiées, c'est-à-dire les fiches respectant la condition :

`CallFileTable.PRIORITE >= -1 OR CallFileTable.PRIORITE IN (-10, -11)`

4.7. Modifier l'affectation d'une fiche ou déplacer une fiche.

Pour modifier l'affectation d'une fiche ou déplacer une fiche, il faut utiliser la fonction suivante :

```
public int ChangeCallsAffectation(int customerId, string campaignId, string destCampaignId,
    string condition, int agentCode)
```

Paramètres

<i>customerId</i> :	Identifiant du site sur lequel se trouve la campagne.
<i>campaignId</i> :	ID d'une campagne d'émission d'appels.
<i>destCampaignId</i> :	ID de la campagne dans laquelle on veut déplacer la fiche. Mettre null pour ne pas déplacer la fiche.
<i>condition</i> :	Condition de sélection des fiches à modifier. Si null ou « » alors modification de toutes les fiches.
<i>agentCode</i> :	Permet d'affecter les fiches à un agent. Mettre 0 pour ne pas affecter la fiche, mettre -1 pour ne pas modifier l'affectation actuelle.

Valeur de retour

Identifiant de la tâche.

Pour modifier l'affectation d'une fiche : mettre destCampaignId=null.

Les valeurs possibles pour agentCode sont : 0 pour supprimer une affectation, ou une valeur de code agent pour affecter les rappels à un agent donné.

Pour déplacer une fiche d'une campagne à une autre : mettre une valeur valide dans destCampaignId.

Les valeurs possibles pour agentCode sont : 0 pour supprimer une affectation, -1 pour ne pas modifier les affectations actuelles, ou une valeur de code agent pour affecter les rappels à un agent donné.

Note : Pour pouvoir déplacer une fiche d'une campagne à une autre, il faut au choix que :

- Les campagnes utilisent le même fichier client, ou aucun fichier client. Dans ce cas seuls les fichiers d'appels des campagnes sont impactés.
- Les campagnes utilisent des fichiers clients ayant exactement la même structure (même colonnes de mêmes types). Dans ce cas les données des fichiers clients et des fichiers d'appels sont déplacées.

5. Lecture d'informations

5.1. Informations sur une campagne

Il est possible d'obtenir des informations de configuration sur une campagne en utilisant la fonction suivante :

```
public CallFileInfo GetCallFileInfos(int customerId, string campaignId)
```

Paramètres

customerId : Identifiant du site sur lequel se trouve la campagne.
campaignId : ID d'une campagne d'émission d'appels.

Valeur de retour

Informations sur le fichier d'appel de la campagne.

La classe `CallFileInfo` a la structure suivante :

```
public class CallFileInfo
{
    public TaskResult Result { get; set; }
    public string LastError { get; set; }

    public string CallFileConnectionString { get; set; }
    public string CallFileTable { get; set; }
    public string ClientFileConnectionString { get; set; }
    public string ClientFileTable { get; set; }
}
```

Result : Résultat de la demande, les valeurs possibles sont :

CompletedOK :	demande complétée, sans erreur
ErrorCampaignNotFound :	campagne demandée non trouvée
ErrorNoCallFile :	fichier d'appel non trouvé
ErrorUnauthenticated :	utilisateur non authentifié
ErrorCustomerNotAllowed :	tentative de manipuler la campagnes d'un site auquel le compte administrateur utilisé n'a pas accès.
ErrorUnknown :	erreur non répertoriée

LastError : En cas d'erreur, contient le texte de l'exception qui est survenue ou une précision sur l'erreur.

<i>CallFileConnectionString</i> :	Chaine de connexion OleDb vers la base du fichier d'appel.
<i>CallFileTable</i> :	Nom de la table du fichier d'appel.
<i>ClientFileConnectionString</i> :	Chaine de connexion OleDb vers la base du fichier client.
<i>ClientFileTable</i> :	Nom de la table du fichier client.

Note : Cette fonction n'est pas asynchrone car elle n'est pas susceptible d'être consommatrice en temps et ressource.

5.2. Contenu d'un fichier d'appels.

Pour obtenir des informations sur le contenu d'un fichier d'appels, il faut utiliser la fonction suivante :

```
public int GetCallsInformations(int customerId, string campaignId, string condition,
                                int phoneNumberCount)
```

Paramètres

customerId : Identifiant du site sur lequel se trouve la campagne.
campaignId : ID d'une campagne d'émission d'appels.
condition : Condition de sélection des fiches à modifier.
 Si null ou « » alors lecture de toutes les fiches.
phoneNumberCount : Nombre de numéros de téléphone que l'on veut récupérer (entre 0 et 10)

Valeur de retour

Identifiant de la tâche.

Il s'agit d'une tâche asynchrone, et il faut donc utiliser la fonction `GetTaskProgression` pour connaître l'état de la tâche. Une fois la tâche terminée, on peut consulter le résultat en appelant la fonction :

```
public CallInfo[] GetCallsInformationsResult(int taskNumber)
```

Paramètres

taskNumber : Identifiant de la tâche

Valeur de retour

Liste de fiches du fichier d'appel

La classe `CallInfo` a la structure suivante :

```
public class CallInfo
{
    public long Indice { get; set; } // Indice de la fiche
    public bool Excluded { get; set; } // Fiche exclue (true / false)
    public string ExcludedReason { get; set; } // Raison de l'exclusion ou ""
    public DateTime CallBackHour { get; set; } // Heure du rappel programmé ou
                                                // DateTime.MinValue si pas de rappel
    public int Priority { get; set; } // Priorité de la fiche ("PRIORITE")
    public int SortOrder { get; set; } // Valeur du champ MIXUP
    public int NbCalls { get; set; } // Valeur du champ NBAPPELS
    public int NivAbs { get; set; } // Valeur du champ NIVABS
    public int AgentCode { get; set; } // Agent affecté à la fiche (ou 0)
    public int StatusGroup { get; set; } // Groupe de qualification utilisé
    public int StatusCode { get; set; } // Code de la qualification utilisée
    public int StatusDetail { get; set; } // Code détaillé de la qualification
    public string StatusLib { get; set; } // Description de la qualification
    public string StatusSubLib { get; set; } // Description détaillée de la qualification
    public string[] PhoneNumbers { get; set; } // Numéros de téléphone (champs TEL1 à TEL10)
}
```

Note : cette fonction ne retournera en aucun cas plus de 500 fiches. Si la condition de sélection de fiches retourne plus de 500 fiches, seules les 500 premières fiches trouvées seront retournées par la fonction.