Documentation SDK PHP La Presse Libre

Emis par : Bastien Caubet, Luc Raymond

Date : 26/02/2015

Etat : Brouillon

Révision	Commentaires
2.0	Ajout de l'inscription partenaire Correctifs
1.0	Document initial

Table des matières

1. Généralités

Ce document a pour but d'expliquer le fonctionnement du SDK PHP aux développeurs partenaires du projet La Presse Libre utilisant cette technologie. Ce SDK met à disposition une API pour faire communiquer le serveur partenaire et le serveur de La Presse Libre au travers de web-services.

Le SDK est décomposé en trois parties :

- Une partie API qui comprend les classes nécessaires à la communication inter-serveurs.
- Une partie Models, qui contient les objets envoyés et attendus par la plateforme La Presse Libre
- Une partie Utils, qui comprend différents utilitaires.

L'API est composée de 3 web-services :

- Un service de vérification, qui permettra à La Presse Libre de vérifier l'existence d'un compte utilisateur dans la base de données du partenaire.
- Un service de création de compte, qui permettra de propager un nouveau compte créé sur La Presse Libre dans la base de données partenaire.
- Un service de mise à jour de compte, qui permettra de mettre à jour un compte utilisateur (abonnement / reconduite de l'abonnement ou informations personnelles) sur la plateforme partenaire.

Chacun de ces web-services doit être implémenté pour assurer le bon fonctionnement de la communication entre La Presse Libre et la plateforme partenaire.

2. Informations concernant le SDK

Le SDK La Presse Libre pour PHP nécessite d'utiliser PHP 5.x

Il sera laissé au choix du développeur d'utiliser une classe d'autoload ou non pour gérer la dépendance de fichiers.

3. Configuration

IMPORTANT : Pour pouvoir communiquer avec la plateforme La Presse Libre, il est nécessaire de configurer le SDK. À la racine du dossier LPL-PHP-SDK vous trouverez un fichier nommé « Config.php » : chaque attribut de ce fichier doit être mis à jour avec des valeurs de production.

Pour obtenir ces valeurs de production, veuillez-vous connecter avec un compte gestionnaire à la plateforme de gestion des partenaires à l'adresse suivante : http://lplbackend.azurewebsites.net/ et vous rendre dans la partie « Gestion des informations partenaires ». Vous y trouverez une section pour télécharger les identifiants au format JSON.

Si vous ne disposez pas encore d'un compte gestionnaire, veuillez-vous rapprocher de l'administration La Presse Libre.

À noter qu'à tout moment, vous pouvez faire une demande de modification de ces identifiants (pour des raisons de sécurité) à l'administration La Presse Libre. Une fois les codes régénérés, rendez-vous de nouveaux dans la partie « Gestion des informations partenaires » où ils pourront être télécharger. ,modifiées . Il sera laissé au développeur le fait de changer ces codes dans le fichier « Config.php »

4. Routage

IMPORTANT : Pour accepter les requêtes venant de la plateforme La Presse Libre, il sera nécessaire de vous vous connecter avec un compte gestionnaire à la plateforme de gestion des partenaires à l'adresse suivante : http://lplbackend.azurewebsites.net/ et vous rendre dans la partie « Gestion des informations partenaires ». Vous pourrez y configurer les points de terminaisons de vos trois web-services.

Pour pouvoir tester le bon fonctionnement de vos web-services, rendez-vous sur l'environnement de test à l'adresse suivante : http://sandboxlpl.azurewebsites.net/

5. Recevoir les requêtes de vérification de compte

Exemple d'implémentation

```php
<?php

require_once "LPL-PHP-SDK/api/VerificationService.php";
require_once "LPL-PHP-SDK/models/UserInfoModel.php";

try {
    $service = new VerificationService($_SERVER);
    $verificationModel = $service->getResult();

    // Vérification en base de données à partir de l'objet $verificationModel

    $model = new UserInfoModel();
    $model->Mail = "jean.dupont@gmail.com";
    $model->CodeUtilisateur = "e5016836-dfbe-49e1-82d7-b8ac300da6aa";
    $model->TypeAbonnement = "mensuel";
    $model->PartenaireID = 2;
    $model->DateExpiration = "14-02-2012";
    $model->DateSouscription = "14-02-2012";
    $model->AccountExist = TRUE;

    echo $service->createResponse($model, 200);
} catch(Exception $e) {
    header('HTTP/1.1 401 Unauthorized', true, 401);
    echo json_encode(Array('error' => $e->getMessage()));
}
```

Détails

La première chose à faire est d'ajouter les références nécessaires au bon fonctionnement de l'API.

```php
require_once "LPL-PHP-SDK/api/VerificationService.php";
require_once "LPL-PHP-SDK/models/UserInfoModel.php";
```

Le fichier VerificationService.php représente le service, il va effectuer les traitements suivants :

- Récupère les paramètres de la requête
- Déchiffre les paramètres Json en objet
- Créer la réponse HTTP

Le fichier UserInfoModel.php représente le modèle de données de retour attendu par LPL.

Le service, après avoir été instancié avec la variable globale $_SERVER, est exécuté grâce à la méthode getResult. Cette méthode se charge de déchiffrer les données envoyées par LPL, et renvoie un objet de type VerificationModel.

```php
$service = new VerificationService($_SERVER);
$verificationModel = $service->getResult();
```

Une fois l'objet créé, le partenaire devra interroger sa base de données afin de rechercher l'adresse mail transmise. Cette étape n'est pas implémentée, elle dépendra de la manière dont les données sont traitées et stockées côté partenaire.

L'étape suivante consiste à créer le modèle de réponse attendu par LPL.

```php
$model = new UserInfoModel();
$model->Mail = "jean.dupont@gmail.com";
$model->CodeUtilisateur = "e5016836-dfbe-49e1-82d7-b8ac300da6aa";
$model->TypeAbonnement = "mensuel";
$model->PartenaireID = 2;
$model->DateExpiration = "14-02-2012";
$model->DateSouscription = "14-02-2012";
$model->AccountExist = TRUE;
```

Après la vérification en base de données, pour avertir LPL de la validation ou non, il faut envoyer une réponse. Il suffira d'appeler la méthode createResponse, avec comme paramètres un objet de type UserInfoModel et le statut HTTP de la requête.

```php
$service->createResponse($model, 200);
```

En cas d'erreur, une exception sera levée et transmise dans la réponse HTTP.

```php
} catch(Exception $e) {
    header('HTTP/1.1 401 Unauthorized', true, 401);
    json_encode(Array('error' => $e->getMessage()));
}
```

6. Recevoir les requêtes de création de compte

Exemple d'implémentation

```php
<?php

require_once "LPL-PHP-SDK/api/CreationCompteService.php";
require_once "LPL-PHP-SDK/models/ValidationReponseModel.php";

try {
    $service = new CreationCompteService($_SERVER);
    $createModel = $service->getResult($service->file);

     // Traitements en base de données à partir de l'objet $createModel

    $model = new ValidationReponseModel();
    $model->CodeUtilisateur = "e5016836-dfbe-49e1-82d7-b8ac300da6aa";
    $model->IsValid = TRUE;
    $model->PartenaireID = 2;

    echo $service->createResponse($model, 200);
} catch(Exception $e) {
    header('HTTP/1.1 401 Unauthorized', true, 401);
    echo json_encode(Array('error' => $e->getMessage()));
}
```

Détails

La première chose à faire est d'ajouter les références nécessaires au bon fonctionnement de l'API.

```php
require_once "LPL-PHP-SDK/api/CreationCompteService.php";
require_once "LPL-PHP-SDK/models/ValidationReponseModel.php";
```

Le fichier CreationCompteService.php représente le service, il va effectuer les traitements suivants :

- Récupère le message de la requête
- Déchiffre le message Json en objet
- Créer la réponse HTTP

Le fichier ValidationReponseModel.php représente le modèle de données de retour attendu par LPL.

Le service, après avoir été instancié avec la variable globale $_SERVER, est exécuté grâce à la méthode getResult. Cette méthode se charge de déchiffrer les données envoyées par LPL, et renverra un objet de type CreationCompteModel.

```php
$service = new CreationCompteService($_SERVER);
$createModel = $service->getResult($service->file);
```

Une fois l'objet créé, le partenaire devra envoyer les données en base pour créer un nouveau compte utilisateur. Cette étape n'est pas implémentée, elle dépendra de la manière dont les données sont traitées et stockées côté partenaire.

L'étape suivante consiste à créer le modèle de réponse attendu par LPL.

```
$model = new ValidationReponseModel();
$model->CodeUtilisateur = "e5016836-dfbe-49e1-82d7-b8ac300da6aa";
$model->IsValid = TRUE;
$model->PartenaireID = 2;
```

Après les traitements en base de données, pour avertir LPL de la validation ou non, il faut envoyer une réponse. Il suffira d'appeler la méthode createResponse, avec comme paramètres un objet de type ValidationReponseModel et le statut HTTP de la requête.

```
$service->createResponse($model, 200);
```

En cas d'erreur, une exception sera levée et transmise dans la réponse HTTP.

```
} catch(Exception $e) {
    header('HTTP/1.1 401 Unauthorized', true, 401);
    json_encode(Array('error' => $e->getMessage()));
}
```

7. Recevoir les requêtes de mise à jour de compte

Exemple d'implémentation

```php
<?php

require_once "LPL-PHP-SDK/api/MajCompteService.php";
require_once "LPL-PHP-SDK/models/ValidationReponseModel.php";

try {
    $service = new MajCompteService($_SERVER);
    $majModel = $service->getResult($service->file);

    // Traitements en base de données à partir de l'objet $majModel

    $model = new ValidationReponseModel();
    $model->CodeUtilisateur = "e5016836-dfbe-49e1-82d7-b8ac300da6aa";
    $model->IsValid = TRUE;
    $model->PartenaireID = 2;

    echo $service->createResponse($model, 200);
} catch(Exception $e) {
    header('HTTP/1.1 401 Unauthorized', true, 401);
    echo json_encode(Array('error' => $e->getMessage()));
}
```

Détails

Pour recevoir et traiter les données envoyées par La Presse Libre pour le service de mise à jour d'un compte utilisateur, voici les méthodes à utiliser :

```php
$service = new MajCompteService($_SERVER);
$majModel = $service->getResult($service->file);
```

Le service, après avoir été instancié avec la variable globale $_SERVER, est exécuté grâce à la méthode getResult. Cette méthode se charge de déchiffrer les données envoyées par LPL, et renverra un objet de type MajCompteModel. Cet objet sera ensuite utilisé par le partenaire pour gérer les informations utilisateur dans sa base de données.

```php
$service->createResponse($model, 200);
```

Après les traitements en base de données, pour avertir LPL de la validation ou non, il faut envoyer une réponse. Il suffira d'appeler la méthode createResponse, avec comme paramètres un objet de type ValidationReponseModel et le statut HTTP de la requête.

8. Inscription à La Presse Libre depuis un partenaire

Cette fonctionnalité a pour but de faciliter l'inscription de vos utilisateurs historiques à la plateforme La Presse Libre depuis votre plateforme. Son fonctionnement est basique : à partir des informations de compte de l'utilisateur (son Email et son Identifiant utilisateur), une Url est générée sous la forme suivante :

https://lapresselibre.fr/inscription-partenaire?user=<infos>&partId=<partId>

L'URL sera donc composée de deux paramètres, le premier « user » contiendra les informations relatives à l'utilisateur, le second « partId » permettra d'identifier la plateforme partenaire de provenance.

Pour des raisons de sécurité des informations utilisateurs, les données sont chiffrés à l'aide d'AES256 puis encodés en base64.