



BNP PARIBAS

Merc@net

GUIDE DU PROGRAMMEUR PLUG-IN A6.15

Avertissements :

- **Le fichier Version.txt précise l'environnement dans lequel l'API a été compilée et testée. L'installation de l'API sur tout autre environnement n'est pas garantie.**
- **Les chemins et les champs présents dans les différents fichiers de l'API ne doivent en aucun cas contenir des espaces. La présence d'espaces provoque des messages d'erreur difficiles à interpréter car la valeur des champs concernés est tronquée.**

Sommaire

1. INTRODUCTION.....	3
2. LA REQUÊTE DE PAIEMENT : CALL_REQUEST	3
2.1 PARAMETRES DE LA REQUETE	3
2.2 LE FICHIER PATHFILE	4
2.3 LE MODE DEBUG	4
2.4 EXECUTABLE REQUEST	5
2.5 PARAMETRE TRANSACTION_ID.....	5
3. LA REPONSE A UN PAIEMENT : CALL_RESPONSE	6
3.1 CHAMPS DE LA REPONSE.....	6
3.2 EXECUTABLE RESPONSE	6
4. LA REPONSE AUTOMATIQUE : CALL_AUTORESPONSE	7
4.1 POURQUOI DEUX REPONSES	7
4.2 PRINCIPE DE FONCTIONNEMENT.....	7
4.3 IMPLEMENTATION DU SCRIPT CALL_AUTORESPONSE	7
5. MESSAGES D'ERREUR DE L'API	8
5.1 LORSQUE L'ON APPELLE L'API	8
5.2 LORSQUE L'API APPELLE LE SERVEUR SIPS.....	9
5.3 LA REPONSE AUTOMATIQUE NE FONCTIONNE PAS.....	9

1. INTRODUCTION

Ce document vous décrit les différents scripts et exécutables disponibles dans l'API Merc@net Plug-in, quel est leur rôle et comment les utiliser. Pour intégrer l'API, vous devrez développer 3 scripts (PHP ou ASP) pour traiter les 3 messages échangés entre le serveur de paiement, le site marchand et le navigateur de l'internaute.

L'API Merc@net Plug-in présente 2 exécutables principaux :

- request pour la création de la requête de paiement
- response pour l'interprétation la réponse du serveur

Note : ce document ne décrit pas comment vous interfacer avec votre système d'information ou votre base de données. Dans les exemples fournis, les variables sont déjà renseignées, vous devrez programmer la lecture et la mise à jour des données de votre système d'information.

Logiciels pré-requis :

- Interpréteur PHP.
- Serveur http pour exécuter vos scripts.

Conventions d'écriture :

- Les renvois à d'autres documentations seront notés en majuscules et en italique.
ex : *DICTIONNAIRE DES DONNEES*
- Les exécutables et champs de l'API seront notés en italique.
ex : *request*, *pathfile*, *transaction_id*
- Les scripts que vous devez développer seront notés en gras quel que soit leur type php ou asp
ex : **call_request**

2. LA REQUÊTE DE PAIEMENT : CALL_REQUEST

2.1 PARAMETRES DE LA REQUETE

Le *DICTIONNAIRE DES DONNEES* fournit la liste des champs de la requête, leur description et le caractère obligatoire ou facultatif de leur affectation.

Ces champs peuvent être renseignés dans les fichiers parmcom et dans le script **call_request**. De plus, l'exécutable *request* affectera des valeurs par défaut aux champs *block_align*, *block_order*, *currency_code* et *header_flag* s'ils ne sont renseignés ni dans les fichiers parmcom, ni dans le script **call_request**.

Pour renseigner un champ dans le script **call_request**, il faut créer une chaîne de caractères de la forme *mot-clé=valeur* et passer cette chaîne de caractères en paramètre à l'exécutable *request* (rappel : *valeur* ne doit pas contenir d'espace). Le fichier **call_request** comporte des exemples de chaînes de caractères pour chacun des champs de la requête. Il suffira de transmettre à l'exécutable *request* toutes les chaînes de caractères que vous souhaitez prendre en compte pour la requête.

Comment se déroulera la création de la requête ?

Pour créer une requête de paiement, vous devez appeler le script **call_request** à l'aide d'un navigateur. Ce script appelle l'exécutable *request* avec les champs que vous aurez paramétré. L'exécutable *request* prépare la requête de paiement avec les valeurs qu'il lit dans le fichier parmcom.default, dans le fichier parmcom.011223344551111 et dans les paramètres fournis par le script **call_request**.

Si un champ est renseigné dans le fichier `parmcom.default`, dans le fichier `parmcom.011223344551111` et dans les paramètres fournis par le script **call_request**, sa valeur sera écrasée au fur et à mesure des lectures. Dans cet exemple, le paramètre aura finalement la valeur transmise par le script **call_request**.

Ce type d'affectation vous permet de renseigner statiquement des valeurs dans les fichiers `parmcom` et de les modifier dynamiquement dans le script **call_request** si nécessaire.

2.2 LE FICHIER PATHFILE

Le fichier `pathfile` est le fichier de configuration principal de l'API. Il contient les chemins des autres fichiers paramètres de l'API et permet donc aux exécutables d'accéder aux fichiers `parmcom.default`, `parmcom.011223344551111` et `certificat`. Le fichier `pathfile` contient également le chemin Internet des logos des moyens de paiement ainsi que le mot clé d'activation du mode **DEBUG**.

Deux possibilités s'offrent à vous pour la localisation du fichier `pathfile` sur votre serveur :

- Dans le script **call_request**, vous précisez en paramètre la chaîne de caractères `pathfile=chemin_vers_le_pathfile`. Alors vous devez positionner le fichier `pathfile` par rapport au `chemin_vers_le_pathfile`.
- Dans le script **call_request**, vous ne précisez pas le chemin du `pathfile` en paramètre. Alors vous devez positionner le fichier `pathfile` dans le même répertoire que le script **call_request**.

Le chemin du fichier `pathfile` fourni à l'exécutable doit être un chemin physique (en aucun cas un chemin Internet). On recommande un chemin absolu, comme c'est le cas dans les exemples de scripts livrés.

La syntaxe des lignes de ce fichier est la suivante :

- Chaque ligne commençant par `#` est une ligne de commentaires.
- Les chemins sont paramétrés par des lignes « mot-clé!chemin! »

Les trois paramètres essentiels sont les trois variables `F_CERTIFICATE`, `F_PARAM`, et `F_DEFAULT` qui doivent contenir respectivement les chemins complets des fichiers `certif.fr.011223344551111.php`, `parmcom.011223344551111`, et `parmcom.default`. Seul le fichier `parmcom.default` doit être paramétré avec son extension `.default`. Les extensions des autres fichiers seront ajoutées par l'API d'après les paramètres renseignés dans le script **call_request**.

Il est recommandé de placer des chemins complets dans ces variables. De plus, ces chemins doivent être des chemins physiques du serveur, et non pas des chemins internet.

Le paramètre `F_CTYPE` doit contenir le type du certificat, celui-ci correspond à l'extension du fichier certificat qui sera ici `php` par exemple.

On renseignera alors `F_CTYPE!php!`

Le paramètre `D_LOGO` sert à l'affichage des moyens de paiement. Lui seul doit être renseigné avec une portion de chemin Internet, soit l'URL du répertoire des logos de l'API sur le site web, amputée du nom de domaine de ce dernier. (cf. *GUIDE D'INSTALLATION*).

Le dernier paramètre `DEBUG` sert à l'affichage des moyens de paiement en mode **DEBUG**.

2.3 LE MODE DEBUG

Le mode **DEBUG** permet d'afficher (en format HTML) l'environnement complet de l'api, ainsi que tous les paramètres de la requête et de la réponse d'un paiement.

Pour activer ce mode, le paramètre **DEBUG** du fichier `pathfile` doit être renseigné à **YES**.

Le mode **DEBUG** est constitué de différentes rubriques indiquant les ressources utilisées par l'api. Il permet ainsi de vérifier, par exemple, le contenu du fichier `pathfile`, l'URL du serveur de paiement ou encore la valorisation des champs de la requête de paiement.

En phase de production, ce mode devra être désactivé (renseignement de la valeur du paramètre **DEBUG** autre que **YES**).

2.4 EXECUTABLE REQUEST

Utilisation : `request var1 var2 var3...`

avec

`varx` une chaîne de caractère du type *mot-clé=valeur*

La liste des différents mot-clés est précisée dans le *DICTIONNAIRE DES DONNEES*.

Rappel : *valeur* ne doit pas contenir d'espace.

Retour :

L'exécutable *request* retourne une chaîne de caractères composée de trois chaînes de caractères séparées par un « ! » : `!code!error!buffer!`

avec *code*, le code retour du traitement. Deux valeurs sont possibles :

- 0 : l'exécutable génère un formulaire HTML contenue dans la variable *buffer*
- -1 : l'exécutable retourne un message d'erreur dans la variable *error*. (cf. *MESSAGES D'ERREUR DE L'API*).
-

Le formulaire HTML contenue dans la variable *buffer* comporte :

- La liste des logos des moyens de paiement.
- L' URL du serveur Sips.
- Des données cachées à transmettre au serveur Sips.

Cette page doit ensuite être affichée sur le poste de l'internaute. Les blocs d'icônes de paiement proposés par la boutique incluent chacun un en-tête en option (cf. *GUIDE DE PERSONNALISATION*).

L'exécutable *request* construit un message crypté à partir de toutes les données que vous aurez renseignées dans les fichiers *parmcom* et le script **call_request** (l'activation du mode DEBUG permet de s'assurer de la valeur finale des paramètres de la requête de paiement). Il le signe ensuite par un MAC (Message Authentication Code), un scellement calculé avec un algorithme DES. Ce message crypté apparaît (en variable cachée) dans la page HTML retourné par l'exécutable. Il sera posté sur l'URL du serveur Sips pour permettre l'identification du commerçant et par suite, le traitement de sa demande de transaction.

2.5 PARAMETRE TRANSACTION ID

Dans le script **call_request**, deux possibilités s'offrent à vous pour renseigner le champ *transaction_id* :

- si vous ne le transmettez pas en paramètre à l'exécutable *request*, une valeur sera générée automatiquement. Cette valeur est basée sur l'heure système (format hhmmss), elle est donc conforme au format du champ *transaction_id* (6 caractères numériques).
- Si vous transmettez le *transaction_id* à l'exécutable *request*, la valeur fournie sera prise en compte pour la transaction.

WARNING : Etant donné que la génération automatique du *transaction_id* s'appuie sur l'heure système pour générer un numéro de transaction, on notera deux inconvénients : tout d'abord on ne couvre que 86400 nombres (24h*60mn*60s) et non pas toute la plage possible de numéros de transaction (000000 à 999999). Ceci peut être gênant pour de très gros sites, le numéro de transaction devant être unique sur une journée.

D'autre part, il subsiste un risque de doublons, si deux internautes distincts souhaitant payer au même instant appellent le script **call_request** dans la même seconde. Une des deux transactions sera alors nécessairement rejetée comme un doublon, pour la même cause d'unicité que précédemment.

Il sera donc toujours préférable de gérer un numéro de séquence pour le champ *transaction_id* sur le site web commerçant, plutôt que d'utiliser la génération automatique.

3. LA REPONSE A UN PAIEMENT : CALL_RESPONSE

3.1 CHAMPS DE LA REPONSE

Dans la réponse du paiement, on retrouve les champs généraux de la transaction tels *merchant_id*, *amount*, et *currency_code*, mais aussi les champs renseignés par le serveur Sips suite à la demande d'autorisation : *response_code*, *payment_certificate*, etc...

Le *DICTIONNAIRE DES DONNEES* fournit la liste des champs de la réponse et leur description.

Le serveur Sips renvoie les données de la transaction en méthode POST et de façon cachée. En fait, elle « poste » une variable DATA cryptée sur les URL *normal_return_url* et *cancel_return_url* renseignées dans la requête de paiement. L'exécutable *response* décrypte cette variable et renseigne tous les champs de la réponse.

3.2 EXECUTABLE RESPONSE

Après la validation des coordonnées bancaires de l'internaute, le serveur Sips affiche en cas d'acceptation de la transaction une page de réponse. Cette page résume les données de la transaction, et présente un bouton « RETOUR A LA BOUTIQUE » qui reconnecte l'internaute à l'URL *normal_return_url*. Dans le cas d'un refus de l'autorisation par les serveurs bancaires, le serveur Sips affiche une page de refus avec un bouton « ANNULATION – RETOUR A LA BOUTIQUE » qui renvoie l'internaute vers l'URL *cancel_return_url*.

Le bouton « ANNULATION – RETOUR A LA BOUTIQUE » présent sur la page de saisie des coordonnées carte bancaire a le même effet que son homonyme de la page de refus.

Dans chacun de ces cas, une variable cachée DATA contenant la chaîne de caractères cryptée des données de la transaction est postée sur l'URL *normal_return_url* ou *cancel_return_url*.

Ce mode permet au commerçant de reconnecter l'internaute à son site, puisque le développeur a une totale liberté de paramétrage de ces URL. Il faut malgré tout s'assurer que le serveur commerçant autorise la réception de données en méthode POST.

Remarque : les URL de retour devront nécessairement pointer vers des CGI. En effet, certains serveurs n'acceptent pas que l'on poste des données sur des pages HTML statiques et retournent alors le message « methode POST not allowed ». Rien ne vous empêche par contre d'écrire une page statique dans votre script **call_response**.

Le script **call_response** récupère la variable « DATA » et la transmet en paramètre à l'exécutable *response* qui va ensuite la décrypter et renseigner les différents champs de la réponse (l'activation du mode DEBUG permet d'afficher l'ensemble des paramètres de la réponse du serveur de paiement).

Utilisation : *response message*

avec

message une chaîne de caractère du type *message=DATA*

L'exécutable *response* peut prendre un ou deux paramètres. Le second paramètre qui peut être transmis est une chaîne de caractères du type *pathfile=chemin_vers_le_pathfile* pour spécifier la localisation du fichier pathfile (cf. paragraphe 2.2).

Ainsi, l'exécutable *response* peut prendre un ou deux paramètres en entrée.

Retour : L'exécutable *response* retourne une chaîne de caractères composée de 34 chaînes de caractères séparées par des « ! » :

!code!error!amount!authorisation_id!bank_response_code!caddie!card_number!capture_day!capture_model!complementary_code!complementary_info!currency_code!customer_email!customer_id!customer_ip_adress!cvv_flag!cvv_response_code!data!language!merchant_country!merchant_id!merchant_language!order_id!order_validity!payment_certificate!payment_date!payment_means!payment_time!receipt_complement!response_code!return_context!transaction_condition!transaction_id!transmission_date!

avec *code*, le code retour du traitement. Deux valeurs sont possibles :

- 0 : l'exécutable renseigne les champs de la réponse
- -1 : l'exécutable retourne un message d'erreur dans la variable *error*. (cf. *MESSAGES D'ERREUR DE L'API*).

Tous les champs de la réponse sont donc disponibles au niveau du script **call_response**.

4. LA REPONSE AUTOMATIQUE : CALL_AUTORESPONSE

4.1 POURQUOI DEUX REPONSES

Comme expliqué au paragraphe 3.2, l'appel de l'URL *normal_return_url* (ou *cancel_return_url*) est déclenché par l'internaute, lorsqu'il clique sur le bouton « RETOUR A LA BOUTIQUE » du ticket électronique ou « ANNULATION – RETOUR A LA BOUTIQUE » du ticket ou de la page de saisie des coordonnées bancaires. Par conséquent, il se peut qu'aucune de ces URL ne soit déclenchée lors d'un paiement, pour peu que l'internaute perde sa connexion, ou tout simplement ferme son navigateur sans avoir cliqué sur un de ces boutons. Le commerçant risque alors d'avoir une commande initialisée dans sa base de données, mais qui ne serait pas confirmée avant la réception de son journal des transactions le lendemain.

Pour pallier ce cas, le serveur Sips renvoie une réponse automatique de la transaction en même temps qu'il affiche la page de réponse.

Ce mode permet au commerçant d'être informé systématiquement, et en temps réel, de toutes les transactions qui sont effectuées sur son site.

4.2 PRINCIPE DE FONCTIONNEMENT

La réponse automatique ressemble en tous points à la réponse manuelle, c'est à dire que le serveur poste la même variable *DATA* cryptée que celle de la réponse manuelle sur l'URL *automatic_response_url*.

La différence majeure est que l'envoi de cette réponse automatique est fait par le serveur Sips, et non pas par le navigateur de l'internaute.

L'absence de navigateur implique des différences du point de vue du traitement de cet appel. Le script **call_autoreponse** ne peut pas faire d'affichage, ni de redirection (pas de navigateur à l'écoute). On ne peut y effectuer que des traitements automatiques, tels une mise à jour de base de données, un envoi de mail, ou tout simplement une sauvegarde des données dans un fichier comme cela est présenté dans l'exemple.

4.3 IMPLEMENTATION DU SCRIPT CALL_AUTORESPONSE

Le script **call_autoreponse** est très proche du script **call_response**. Ce script récupère tout d'abord la variable cryptée *DATA* et la transmet à l'exécutable *response*. Le chemin vers le pathfile peut également être transmis en paramètre à l'exécutable *response* (cf. paragraphe 2.2). L'exécutable *response* renverra un message d'erreur ou la liste des champs composant la réponse.

A ce stade, vous vous trouvez en possession de toutes les données de la transaction.

Le traitement qui suit est au libre choix du commerçant, en fonction de son système d'information. (mise à jour de base données, envoi de mail, ...). Dans le script d'exemple `call_autoreponse`, les messages d'erreur et les champs de la réponse sont sauvegardés dans un fichier.

Si votre script **call_autoreponse** ne fonctionne pas, consultez le chapitre *MESSAGES D'ERREUR DE L'API*.

5. MESSAGE D'ERREUR DE L'API

Vous trouverez ci-dessous les messages d'erreur qui apparaissent le plus souvent durant la phase d'installation, et la manière de les résoudre.

5.1 LORSQUE L'ON APPELLE L'API

Le message s'affiche sur la page sur laquelle on devrait voir les logos des cartes dans ce cas, les logos ne s'affichent pas.

Un message d'erreur (Ex : Error in call parameters structure (merchant_id not filled)) est constitué de deux parties :

- le message d'erreur (ex : Error in call parameters structure)
- le code diagnostic (ex : merchant_id not filled)

Ci-dessous vous trouverez les principaux messages renvoyés par l'API.

Message	Cause	Solution
erreur appel request executable request non trouve	Le script <code>call_request</code> ne peut pas lancer l'exécutable request	Vérifiez le chemin et les droits d'exécution du fichier request.
Invalid Keyword in parameter (toto=12)	Le paramètre toto n'est pas une clé valide de l'API	Les seuls mots clés utilisables sont ceux décrits dans le <i>DICTIONNAIRE DES DONNEES</i>
Error parameter (transaction_id=1234567) too long	Le paramètre passé à l'API est trop long	Contrôlez la taille du paramètre grâce au <i>DICTIONNAIRE DES DONNEES</i>
Error reading pathfile (chemin du fichier pathfile)	L'API ne peut pas ouvrir le fichier «chemin du fichier pathfile»	Vérifiez le chemin et les droits de lecture du fichier pathfile.
Error reading pathfile (no keyword F_CERTIFICATE)	Le fichier pathfile ne contient pas le mot clé <code>F_CERTIFICATE</code>	Vérifier la présence du mot clé <code>F_CERTIFICATE</code> dans le fichier pathfile
Error reading pathfile (no keyword F_PARAM)	Le fichier pathfile ne contient pas le mot clé <code>F_PARAM</code>	Vérifier la présence du mot clé <code>F_PARAM</code> dans le fichier pathfile
Error reading pathfile (no keyword F_DEFAULT)	Le fichier pathfile ne contient pas le mot clé <code>F_DEFAULT</code>	Vérifier la présence du mot clé <code>F_DEFAULT</code> dans le fichier pathfile
Error reading pathfile (no keyword D_LOGO)	Le fichier pathfile ne contient pas le mot clé <code>D_LOGO</code>	Vérifier la présence du mot clé <code>D_LOGO</code> dans le fichier pathfile
Error reading default parameters definition (chemin du fichier parmcom.default)	L'API ne peut ouvrir le fichier « chemin du fichier parmcom.default »	Vérifiez le chemin et les droits de lecture du fichier.
Error reading merchant parameters definition (chemin du fichier parmcom.011223344551111)	L' API ne peut ouvrir le fichier « chemin du fichier parmcom.011223344551111 »	Vérifiez le chemin et les droits de lecture du fichier.
Error open certificate file (chemin du fichier certif.fr.011223344551111.php)	L'API ne peut pas ouvrir le fichier certificat « chemin du fichier certif.fr.011223344551111.php »	Vérifiez le chemin et les droits de lecture du fichier. Vérifiez également la cohérence entre les paramètres

		merchant_country, merchant_id, et le nom de votre fichier certificat. Vérifier dans le pathfile le paramétrage du mot clé F_CTYPE à php
Error invalid separator in file (chemin du fichier pathfile)	Le fichier concerné a une syntaxe incorrecte (ici, le fichier pathfile)	Vérifiez les lignes du fichier cité dans le message. Il manque un ou plusieurs séparateurs. « ! »
Error in call parameters structure (champ en erreur)	Le champ indiqué dans le message est invalide	Agir en fonction du message d'erreur et du format de la donnée. (Se référer au <i>DICTIONNAIRE DES DONNEES</i>)
Autres messages		Contactez le Centre d'Assistance Technique

Si le message d'erreur contient un chemin de fichier, ce dernier dépend de votre système d'exploitation :

- Windows : "c:\\repertoire\\pathfile"
- Unix : "/home/repertoire/pathfile"

5.2 LORSQUE L'API APPELLE LE SERVEUR SIPS

Le serveur Sips affiche ces messages d'erreur. Les erreurs graves sont affichées en rouge sur un fond jaune. Les autres erreurs sont affichées sur la page HTML standard de paiement. En mode « démonstration », les messages d'erreur peuvent s'afficher sur fond vert. Le message affiché fournit le paramètre en erreur.

Message	Cause	Solution
Security error	MAC reçu incorrect	Vérifiez votre certificat. Le serveur commerçant et le serveur de paiement ne partagent pas le même certificat.
Invalid Transaction	Un des paramètres de paiement n'est pas valide (mauvais format)	Vérifiez le format des paramètres de la transaction
	message modifié	fraude suspectée, Appelez l'équipe technique
	Un des paramètres de paiement ne correspond pas à votre configuration bancaire	Vérifiez la cohérence entre votre contrat vente à distance et votre transaction (devise acceptée par exemple)
Transaction already processed	L'identifiant de transaction a déjà été utilisé dans la journée	Configurer un transaction_id unique sur une journée pour chaque paiement
Autres messages		Contactez le Centre d'Assistance Technique

5.3 LA REPONSE AUTOMATIQUE NE FONCTIONNE PAS

- L'URL doit être accessible depuis un accès Internet extérieur. Un login/password, ou un firewall sont susceptibles de bloquer l'accès à votre serveur.
- Vous ne devez pas utiliser de protocole https.
- Contrôlez les log d'accès de votre serveur (historique).
- Si vous utilisez un port non standard, ce dernier doit se situer dans la plage de 80 à 9999.

- Vous ne devez pas passer de variables de session avec l'URL, pour transférer des informations utiliser les champs CADDIE (2048 caractères) et RETURN_CONTEXT (256 caractères).
- Si vous avez validé le fait que le serveur Sips appelle bien votre URL *automatic_response_url*, il y a sans doute une erreur dans votre script **call_autoresponse**. La similitude entre les réponses manuelle et automatique permet de déboguer facilement le script **call_autoresponse**. Il suffit pour cela de renseigner l'URL *normal_return_url* avec l'URL du script **call_autoresponse**. Ainsi, ce dernier sera déclenché par le bouton « RETOUR A LA BOUTIQUE » du ticket électronique, et ce dans un navigateur. La présence de ce navigateur vous permettra d'ajouter des affichages tout au long du traitement pour tracer les commandes défectueuses. Bien entendu, ces affichages seront à supprimer quand vous paramètrerez le script **call_autoresponse** dans l'URL *automatic_response_url*.
- Enfin, si votre script de réponse automatique fonctionne quand il est appelé via l'URL de réponse manuelle, mais ne fonctionne pas via l'URL de réponse automatique, il y a probablement un cookie, une redirection, ou toute autre chose nécessitant la présence d'un navigateur dans le code de votre script **call_autoresponse**.