ATOS ORIGIN

SIPS OFFICE

Guide du développeur

www.sips-atos.com

TABLE DES MATIERES

HISTORIQUE DES REVISIONS DOCUMENT

Ce document référence : **SIPS OFFICE / dev**

Version	Date	auteur	Motif
1	06/12/2001	FBI	Création du document

REFERENCE DOCUMENTAIRE

Référence	Description
SIPS OFFICE/services	Guide commerçant SIPS OFFICE
SIPS OFFICE/install	Documentation d'installation et paramétrage SIPS OFFICE
SIPS OFFICE/overview	présentation api SIPS OFFICE

Introduction

Dans cette documentation, vous trouverez des informations pour….

- **Comprendre le fonctionnement de l'API SIPS Office**: Un schéma explicatif, exemples, définitions, ..
- **Faciliter votre recherche** : la totalité des paramètres de l'application est répertoriée en différents chapitres et sous forme de tableaux
- **Illustrer son fonctionnement** : exemples d'appel et de réponse du serveur en XML
- **Consulter des références** : nombreuses annexes

A qui s'adresse ce manuel ?

Ce manuel s'adresse aux développeurs chargés d'intégrer l'API SIPS Office. Ce produit nécessite donc quelques connaissances dans un langage de développement couramment employé sur les serveurs Web. Des exemples en java, asp, perl, php ou C sont fournis avec cette API. Pour l'installation automatique, reportez-vous au guide d'installation du produit. SIPS Office s'installe automatiquement sur votre machine en suivant votre paramétrage.

Chapitre

1

SCHEMA d'architecture

Positionnement de l'API SIPS OFFICE …

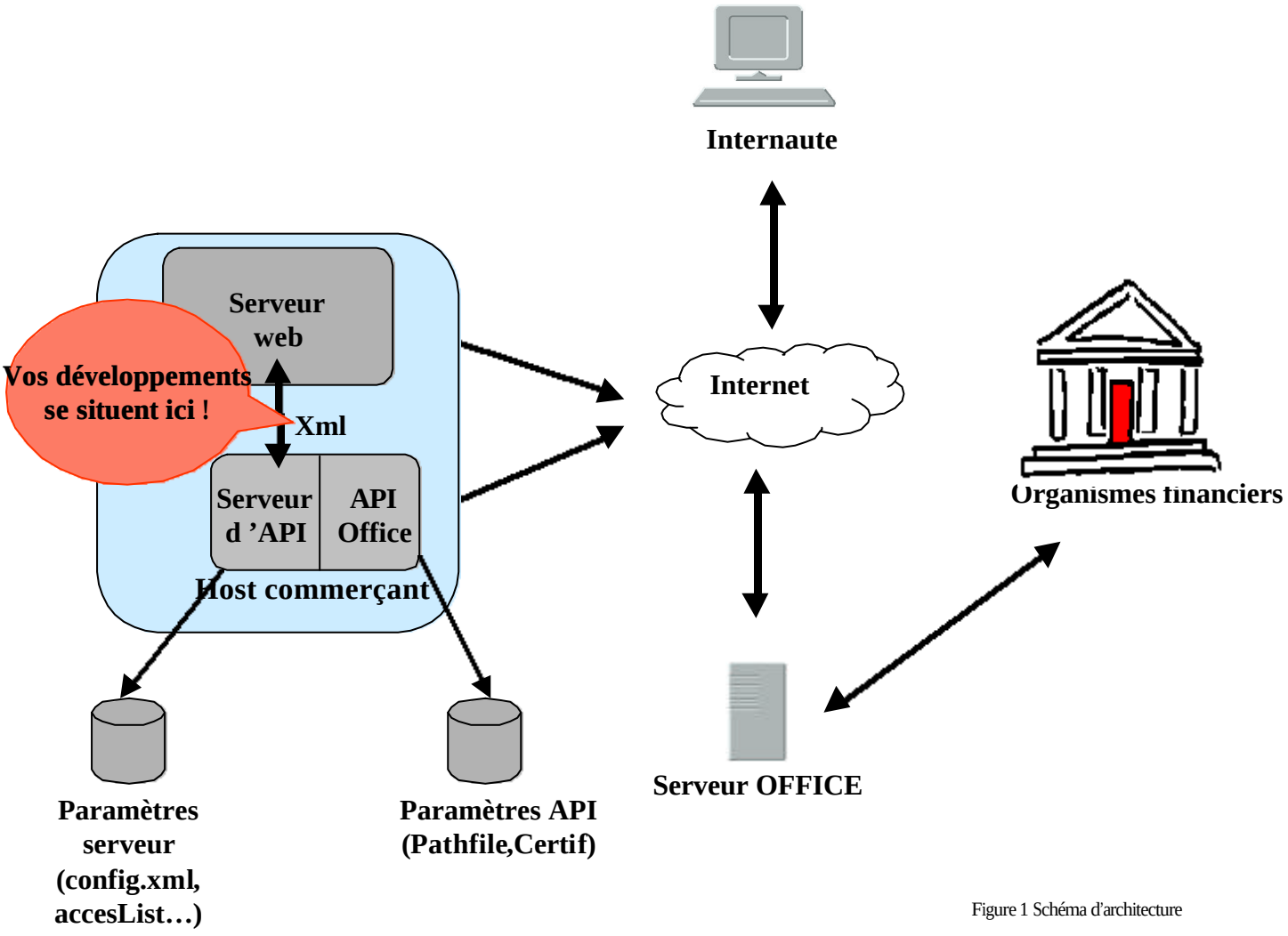


Figure 1 Schéma d'architecture

Chapitre

2

Présentation de SIPS OFFICE

Pour répondre à la demande, tout commerçant doit avoir les moyens de réagir et de gérer sa caisse dans les meilleurs délais. Dans cette optique, ATOS ORIGIN a conçu une offre complémentaire au paiement, SIPS OFFICE capable de gérer la caisse du commerçant selon les cas suivants.

Pour les paiements effectués par carte (ou par chèque) sur Internet, le commerçant peut donc effectuer les opérations suivantes :

Autorisation d'une transaction
Cette fonction permet de faire une demande d'autorisation pour une transaction par carte bancaire.

Validation d'une transaction pour envoi en remise en banque
Cette fonction permet de déclencher partiellement ou totalement l'envoi en banque d'une transaction précédemment autorisée.

Remboursement d'une transaction préalablement remisée en banque
Cette fonction permet de rembourser partiellement ou totalement une transaction déjà envoyée en banque.

Annulation d'une transaction avant la remise en banque
Cette fonction permet d'annuler totalement ou partiellement l'envoi d'une transaction précédemment autorisée. Contrairement au remboursement, l'annulation ne peut intervenir quand le paiement a été envoyé en banque.

Forçage d'une transaction
Dans le cas où la demande d'autorisation aurait été refusée, la fonction Forçage permet au commerçant, après une demande préalable à la banque du porteur, d'autoriser une transaction.

Duplication d'une transaction
Cette fonction permet de dupliquer une transaction existante. Seules les coordonnées bancaires sont conservées, les autres paramètres sont donc modifiables.

Pour avoir plus de précisions sur les fonctions décrites ci -dessus, vous pouvez vous reporter à la documentation fonctionnelle utilisateur destinée au commerçant.

Simulation d'une transaction

La fonction de simulation permet de tester le bon fonctionnement du serveur d'API et du composant appelé. Elle ne vous permet pas d'atteindre le serveur SIPS Office mais simplement de valider que la connexion a bien été établie entre l'application cliente et le module SIPS Office. Pour plus de détails, reportez-vous au chapitre 6 sur les différentes phases.

Note

La fonction **Simulation** vous permet de **tester** le serveur, **mais** ne vous retourne **en aucun cas des informations valides.** Certains champs sont volontairement renseignés à X pour éviter une utilisation réelle sur un site ouvert.
Par exemple, pour la fonction Simulation, vous recevrez en retour, la chaîne de caractères suivante :
<response response_code="XX" transaction_time="16:49:41" transaction_date="20011211" transaction_certificate="XXXX" authorization_id="XXXX" new_status="SIMULATION" new_amount="12300" credit_amount="12300" currency_code="978" data="ceci est un test" cvv_response_code="X" /> .

Intégrer l'api SIPS Office sur votre serveur

Les fonctions de l'API sont implémentables sur vos serveurs selon un schéma toujours identiques :

1. **Collecte des informations** : Elles sont nécessaires lors du traitement de la requête par le serveur de signature, vous trouverez une description des ces informations dans le chapitre 3 : « **Dictionnaire des données** » (nom, format, signification, présence dans la requête)

2. **Constitution du message** : Il s'agit ici de créer un message de type XML selon les spécifications données dans le chapitre 5 « **exemples** ».

3. **Envoi de la requête au serveur d'api** : Les exemples de code dans les **annexes** montrent comment se connecter au serveur d'API puis envoyer le message XML. Lorsque l'application cliente envoie le message de requête elle se met en attente de la réponse. Les exemples de codes permettent de voir comment recevoir la réponse.

4. **Analyse de la réponse** : Le format de la réponse est également de type XML, il est décrit dans le chapitre 4 « **exemples** ». L'application cliente doit alors analyser le contenu, soit directement par recherche des mots clés manuellement, soit en utilisant un parser XML. Le parsing (analyse d'un message XML) a été volontairement simplifié afin de ne faire figurer qu'une seule occurrence pour chaque mot clé (éléments ou attributs XML) ainsi l'analyse de la réponse se fait en spécifiant uniquement le nom du mot clé

   (exemple **valeur = getElementAttribut("response",1, "new_status");** en java pour récupérer le statut de la transaction dans le buffer **"response"**

Note

Ce document ne décrit pas les principes de parsing XML, ils sont largement disponibles sur les sites de référence sur ce sujet on peut citer **à titre d'exemple**:

http://www.w3.org/XML/
http://xmlfr.org/
http://www.xml.com/

Note

Le serveur d'API et ses composants ne sont pas livrés avec des parser XML (à l'exception d'une version JAVA livrée avec le serveur d'API). Il est nécessaire d'utiliser les parsers de votre choix largement disponibles en libre téléchargement pour les environnements les plus courants.

Les exemples en ASP, PERL et PHP et JAVA montrent l'utilisation de parsers disponibles en téléchargement libre ou intégrés en standard dans l'environnement (microsoft.xmldom).Cela ne constitue, en aucun cas, une forme de prescription ni de recommandation.

Chapitre

3

Dictionnaire des données

*Toutes les données décrites ci-dessous interviennent dans le fonctionnement de l'API SIPS Office (cf norme **ISO 8583** sur les messages initiés par carte de transaction financière). Leur valeur est codée comme ci-contre : AN = Alpha numérique, N = numérique, et A = alphabétique. Attention, le chiffre correspond à la longueur maximale du champ en question et est exprimée en nombre de caractères.*

Amount	
Définition	Montant de la transaction
Format	N 12
Valeurs	L'unité la plus petite de la devise concernée Ex : 10.50 euros correspond à 1050 (centimes) Mais 100 yen (pas de décimales) se code 100 (yen)
Authorization_id	
Définition	Numéro d'autorisation donné par la banque
Format	AN 6
Valeurs	Ex : 1008
Capture_day	
Définition	Nombre de jours avant la capture de la transaction
Format	N 2
Valeurs	Se reporter à l'annexe 6
Capture_mode	
Définition	Ce champ définit les méthodes d'envoi en banque des transactions
Format	A 20
Valeurs	Se reporter à l'annexe 6

Card_number	
Définition	Numéro de carte de paiement
Format	N 21
Valeurs	Ex : 4972000000000000
Card_type	
Définition	Type de carte utilisé
Format	AN 20 (se reporter a l'annexe 1)
Valeurs	Ex : VISA
Card_validity	
Définition	Date de fin de validité de la carte
Format	N 6 ou AAAAMM (certaines cartes n'ont pas de date de fin de validité, reportez-vous à l'annexe 1)
Valeurs	Ex : 200310
Complementary_code	
Définition	Code réponse complémentaire utilisé pour le scoring
Format	N 2
Valeurs	Utilisation future
Credit_amount	
Définition	Montant total du remboursement
Format	N 12
Valeurs	l'unité la plus petite de la devise concernée Ex : 10.50 euros correspond à 1050 (centimes) Mais 100 yen (pas de décimales) se code 100 (yen)
Currency_code	
Définition	Code de la devise sélectionnée
Format	N 3 (se reporter à l' annexe 2)
	Ex : Euro : 978. Compatible ISO 4217

Cvv_flag	
Définition	Drapeau indiquant la présence ou l'absence du CVV
Format	N 1
Valeurs	**0** -> le cvv n'est pas remonté par le commerçant **1** -> le cvv est présent **2** -> le cvv est présent sur la carte du porteur mais illisible **9** -> le porteur a informé le commerçant que le cvv n'était pas imprimé sur sa carte
Cvv_key	
Définition	Valeur du code cvv
Format	N 3
Valeurs	Champ obligatoire si le champ CVV_FLAG indique la présence du CVV (CVV=1)
Cvv_response_code	
Définition	Code réponse CVV envoyé par la banque porteur
Format	AN 2
Valeurs	**4E** -> cvv incorrect **4D** -> cvv correct **50** –> cvv non traité (date de validité absente) **53** -> cvv absent de la demande d'autorisation **55** -> L'émetteur n'est pas certifié, le contrôle n'a pu être effectué
Data	
Définition	Données privées entre le serveur SIPS Office et celui du commerçant
Format	AN 1024
Valeurs	Pour certaines cartes de paiement, ce champ doit être renseigné (reportez-vous à l'annexe 1)
Error_message	
Définition	
Format	
Valeurs	

From_transaction_id	
Définition	Identifiant de la transaction
Format	N 6
Valeurs	Ex : 20
From_payment_date	
Définition	Date de la transaction à modifier
Format	N 8 (AAAAMMJJ)
Valeurs	Ex : 20011105, soit le 5 novembre 2001
Merchant_country	
Définition	Code pays du commerçant
Format	A 2
Valeurs	Ex : fr = France (voir annexe 3)
Merchant_id	
Définition	Identifiant du commerçant
Format	N 15
Valeurs	Il s'agit du numéro de SIRET précédé par 0
New_amount	
Définition	Montant de la transaction après opération
Format	A 12
Valeurs	Varie selon le montant initial
New_status	
Définition	Etat de la transaction après l'opération
Format	A 20
Valeurs	La valeur de ce champ dépend de la nature de l'opération. Reportez-vous à l' annexe 4.

Order_id	
Définition	Numéro de la commande configuré par le commerçant en fonction de son système d'information. Aucun contrôle n' est effectué par SIPS Office.
Format	N 32
Valeurs	Ex : OI_131100_8744
Order_validity	
Définition	Période de validité de la commande jusqu'à la date de paiement
Format	N2
Valeurs	Utilisation future
Origin	
Définition	Origine de la transaction
Format	AN 20
Valeurs	Identifiant du programme ou du user émetteur de la demande. Les données de ce champ sont libres.
Pathfile	
Définition	Il s'agit du chemin complet du fichier Pathfile. Pour une définition plus précise, reportez-vous à l'introduction de ce document.
Format	AN 200
Valeurs	Ex : /home/apiOffice/param/pathfile
Payment_date	
Définition	Date locale de la transaction renseignée par le serveur SIPS Office
Format	N 8 (AAAAMMJJ)
Valeurs	Ex : 20011115 : 15 Novembre 2001

Payment_time	
Définition	Heure locale de la transaction renseignée par le serveur SIPS Office
Format	N 6 (HHMMSS)
Valeurs	Ex : 130000 signifie 13h00m00s
Response_code	
Définition	Code réponse applicatif du serveur SIPS Office
Format	A2
Valeurs	Les valeurs de ce champ varient selon la nature de l'opération. Reportez-vous à l'annexe 5.
Return_context	
Définition	Ce champ est facultatif, il contient des données.
Format	A 256
Valeurs	Au libre choix du commerçant
Transaction_certificate	
Définition	Numéro du certificat de la transaction retourné par le serveur SIPS Office
Format	N 12
Valeurs	Ex :100f58l14o
Transaction_date	
Définition	Date de la transaction enregistrée et retournée par le serveur SIPS Office
Format	N 8 (AAAMMJJ)
Valeurs	Ex : 20011031 soit le 31 octobre 2001
Transaction_time	
Définition	Heure de la transaction retournée par le serveur SIPS Office
Format	N 6 (HHMMSS)
Valeurs	Ex : 102543

Transaction_id	
Définition	Cet identifiant permet de référencer de façon unique une transaction sur le serveur SIPS Office, c'est la base du contrôle d'unicité (sur une période de 24 heures). Une valeur déjà utilisée sur cette période provoquerait un rejet lors de l'accès sur le serveur de paiement
Format	N 6
Valeurs	Ex : 010000

Chapitre

4

Paramètres

Les paramètres en entrée

Les paramètres suivent la légende suivante : R = Requis , F = Facultatif et C = Conditionnel. Les paramètres conditionnels font référence à différents cas de figure selon la nature des opérations.

Paramètres	Autorisation	Validation	Remboursement	Annulation	Duplication	Forçage
Amount	**R**	**R**	**R**	**R**	**R**	
Authorization_id						**R**
Capture_day	**F**				**F**	
Capture_mode	**F**				**F**	
Card_number	**R**					
Card_type	**R**					
Card_validity	**C** (reportez-vous à l'annexe 1)					
Currency_code	**R**	**R**	**R**	**R**	**R**	
Cvv_flag	**F**					
Cvv_key	**F**					
Data	**F**				**F**	
From_transaction_id					**R**	
From_payment_date					**R**	
Merchant_country	**R**	**R**	**R**	**R**	**R**	**R**
Merchant_id	**R**	**R**	**R**	**R**	**R**	**R**
Order_id	**F**				**F**	
Order_validity	**F**				**F**	
Origin	**F**	**F**	**F**	**F**	**F**	**F**
Pathfile	**R**	**R**	**R**	**R**	**R**	**R**
Payment_date		**R**	**R**	**R**		**R**
Return_context	**F**				**F**	
Transaction_id	**R**	**R**	**R**	**R**	**R**	**R**

Les paramètres en sortie

Voici la liste des paramètres classés par fonction que vous retrouverez en sortie.

Response_code est la réponse du serveur. Si l'opération s'est bien déroulée, il est égal a 0, sinon –1.

Paramètres	Autorisation	Validation	Remboursement	Annulation	Duplication	Forçage
Authorization_id	**X**				**X**	
Currency_code	**X**	**X**	**X**	**X**	**X**	**X**
Credit_amount			**X**			
Complementary_code	**X**				**X**	
Cvv_response_code	**X**				**X**	
Data	Ce champ libre, est identique dans la requête et la réponse du serveur					
Error_message	**X**	**X**	**X**	**X**	**X**	**X**
New_status (reportez-vous à l'annexe 4)	**X**	**X**	**X**	**X**	**X**	**X**
New_amount	**X**	**X**	**X**	**X**	**X**	**X**
Response_code	**X**	**X**	**X**	**X**	**X**	**X**
Transaction_time	**X**	**X**	**X**	**X**	**X**	**X**
Transaction_date	**X**	**X**	**X**	**X**	**X**	**X**
Transaction_certificate	**X**	**X**	**X**	**X**	**X**	**X**

Chapitre

5

Exemples

Exemples de requêtes de services en XML

Les exemples ci dessus reprennent les fonctions d'appel définies dans le chapitre 2 au format XML. Pour obtenir des réponses valides, vous devrez respecter les formats et paramètres requis pour chaque opération. (cf chapitre 4). Vous devrez respecter la syntaxe XML, soit ***mot-clé*** *égal* ***valeur.***

Principe de construction d'un message XML

requêtes

Les exemples qui sont donnés ci dessous montrent comment coder une requête XML et obtenir la réponse également en XML.La construction de la requête doit comporter les éléments suivants :

- **Nom du composant : (tag service component**) C'est le début de toute requête XML d'appel il s'agit ici de préciser
  - le composant d'api à utiliser; il en existe deux pour cette api :
    - **RequestOffice** pour les fonctions de back office paiement

    - **Pour RequestOffice :**
      - **Author**: dema nde d'autorisation de paiement
      - **Validate** : validation de transaction en attente (si capture_mode validation)
      - **Cancel** : annulation d'une transaction
      - **Credit** : remboursement d'une transaction
      - **Duplicate** : clonage de trnascation
      - **Advice** : forçage de transaction (si code réponse paiement = "02")
      - **Simulate** : simulation d'appel au serveur distant

- **La pathfile transaction : (tag : transaction pathfile)** cette donnée permet d'indiquer où est stocké le fichier de paramétrage « **PATHFILE** »

- **Les attributs de la requête** : (voir liste des paramètres en entrée) selon le type de requête, on précise ici chaque valeur d'attributs de la requête sous la forme :

Mot-clé = "valeur"

- Le message se termine par « **\>** » pour marquer la fin de la zone « **transaction** », puis « </**service**> » pour marquer la fin de la requête de service

Réponses

Les réponses aux requêtes sont également structurées de la manière suivante.

- Entête message : (tag response)
- Les attributs contenant les résultats en sortie (Cf. tableau des paramètres en sortie)
- Chaîne de fin de message XML « **\>** »

Remarque : Pour obtenir des résultats cohérents, vous devez prendre en compte les contraintes indiquées pour chaque opération comme la valeur des champs capture_mode ou capture_day saisie préalablement dans la demande d'autorisation (cf annexes 4, 5 et 6).

Note

Afin d'alléger le code, les entêtes standard XML ne sont pas nécessaires mais sont ajoutées par le serveur de signature pour vérification lors du parsing XML (utilisation d'un DTD).

Par contre lors des réponses du serveur d'API vers l'application cliente aucune référence à un DTD n'est incluse dans le message. Il est donc important de désactiver la validation lors du parsing, afin de ne pas déclencher des erreurs pendant cette phase.

Note

Pour des raisons de lisibilité, les requêtes sont présentées ci dessous, sur plusieurs lignes, toutefois, lors de l'appel de l'api, **elles ne doivent pas être codées avec des retours chariot**, dans le cas ou un saut de ligne doit figurer (notamment dans les textes libres ou mini-contrat), il faut coder le signal « %0D0A »

Autorisation

Remarque *: pour les tests , le champ card_number doit se terminer obligatoirement par 00 (sauf pour tester le remboursement) pour obtenir un response_code egal à « 00 ».*

```
<service component = "requestOffice" name = "author">
      <author pathfile="/home/apiOffice/param/pathfile "
              merchant_id = "011223344552222"
              merchant_country = "fr"
              transaction_id = "02"
              amount = "12310"
              currency_code = "978"
              card_number = "4865123123123200"
              card_validity = "200411"
              card_type = "VISA" />
</service>
```

Validation

Remarque *:*

```
<service component = "requestOffice" name="validate">
       <validate pathfile="/home/apiOffice/param/pathfile "
                 origin="toto"
                 merchant_id="011223344552222"
                 merchant_country="fr"
                 transaction_id="20"
                 currency_code="978"
                 payment_date="20011220"
                 amount="100" />
</service>
```

Remboursement

Remarque : *Le remboursement est possible seulement si la demande d'autorisation a été faite avec une carte de type Aurore.*

```
<service component="requestOffice" name="credit">
        <credit pathfile="/home/apiOffice/param/pathfile "
                origin=" "
                merchant_id="«011223344552222»"
                merchant_country="fr"
                transaction_id="03"
                currency_code="978"
                payment_date="20010208"
                amount="100" />
</service>
```

Annulation

```
<service component="requestOffice" name="cancel">
        <cancel pathfile="/home/apiOffice/param/pathfile "
                origin="toto"
                merchant_id="011223344552222"
                merchant_country="fr"
                transaction_id="20"
                currency_code="978"
                payment_date="20010820"
                amount="100" />
</service>
```

Forçage

Remarque *: pour tester cette fonction, vous devez au préalable faire une demande d'autorisation avec un numéro de carte bancaire se terminant par 02, en retour vous obtiendrez un code_réponse 02 (soit referral) à la suite duquel vous pourrez effectuer un forçage de la transaction en question.*

```
<service component="requestOffice"  name="advice">
      <advice pathfile="/home/apiOffice/param/pathfile "
                origin="toto"
                merchant_id="011223344552222"
                merchant_country="fr"
                transaction_id="7"
                payment_date="20010820"
                authorization_id="X354LB"
                data="" />
</service>
```

Duplication

```
<service component="requestOffice" name="duplicate">
        <duplicate pathfile="/home/apiOffice/param/pathfile "
                    origin="toto"
                    merchant_id="011223344552222"
                    merchant_country="fr"
                    transaction_id="21"
                    from_transaction_id="20"
                    from_payment_date="20010820"
                    currency_code="978"
                    amount="100"
                    return_context=" ceci est un test "
                    order_id="OI_131100_8744"
                    capture_mode="VALIDATION"
                    capture_day="2"
                    data=" "
                    order_validity="" />
</service>
```

Simulation

```
<service component="requestOffice" name="simul">
<simul pathfile="/home/apiOffice/param/pathfile "
        origin="toto"
        merchant_id = "011223344552222"
        merchant_country = "fr"
        transaction_id = "22"
        amount = "12300"
        currency_code = "978"
        card_number = "498465123123123200"
        cvv_flag = "1"
        cvv_key = "000"
        card_validity = "200411"
        card_type = "VISA"
        return_context = ""
        order_id="OI_131100_8744"
        capture_mode="VALIDATION"
        capture_day="2"
       data = "ceci est un test"
      order_validity=""/>
</service>
```

Exemples de réponse du serveur

Autorisation

```
<response response_code="00"
        transaction_time="110940"
        transaction_date="20020208"
        transaction_certificate="1009361380"
        authorization_id="1009"
        new_status="TO_VALIDATE"
        new_amount="12300"
        credit_amount=""
        currency_code="978"
        data=""
        cvv_response_code="4D" />
```

Validation

```
<response response_code="00"
        transaction_time="112741"
        transaction_date="20020208"
        transaction_certificate="5ac651696cdb"
        authorization_id=""
        new_status="TO_CAPTURE"
        new_amount="100"
        credit_amount=""
        currency_code="978"
        data=""
        cvv_response_code="" />
```

Remboursement

```
<response response_code="00"
        transaction_time="100901"
        transaction_date="20020208"
        transaction_certificate="61d2766745c5"
        authorization_id=""
        new_status="CREDITED"
        new_amount="0"
        credit_amount="12310"
        currency_code="978"
        data=""
        cvv_response_code="" />
```

Annulation

Deux types de réponses selon les valeurs du capture_mode,

```
<response response_code="00"
        transaction_time="101112"
        transaction_date="20020208"
        transaction_certificate="1013159472"
        authorization_id="1013"
        new_status="TO_CAPTURE"
        new_amount="12310"
        credit_amount=""
        currency_code="978"
        data=""
        cvv_response_code="53" />
```

```
<response response_code="00"
        transaction_time="101347"
        transaction_date="20020208"
        transaction_certificate="00ae25b7e1c6"
        authorization_id=""
        new_status="CANCELLED"
        new_amount="0"
        credit_amount=""
        currency_code="978"
        data=""
        cvv_response_code="" />
```

Forçage

```
<response response_code="00"
          transaction_time="120122"
          transaction_date="20020208"
          transaction_certificate="c33a0d8cf434"
          authorization_id=""
          new_status="TO_CAPTURE"
          new_amount="12300"
          credit_amount=""
          currency_code="978"
          data=""
          cvv_response_code="" />
```

Duplication

```
<response response_code="00"
          transaction_time="114658"
          transaction_date="20011226"
          transaction_certificate="1009363618"
          authorization_id="1009"
          new_status="TO_VALIDATE"
          new_amount="100"
          credit_amount=""
          currency_code="978"
          data="ceci est un test"
          cvv_response_code="4D" />
```

Simulation

```
<response response_code="XX"
          transaction_time="10:31:02"
          transaction_date="20011217"
          transaction_certificate="XXXX"
          authorization_id="XXXX"
          new_status="SIMULATION"
          new_amount=""
          credit_amount=""
          currency_code=""
          data=""
          cvv_response_code="X" />
```

Chapitre

6

paramétrage

Dans ce chapitre vous trouverez une présentation plus précise de certains fichiers présents sur votre machine après installation. Cela va donc vous permettre de mieux comprendre les documents que vous pouvez modifier et les règles à respecter.

Configuration

Durant l'installation de l'API, un grand nombre d'informations vous ont été demandées. Une partie de ces données constituent la configuration du serveur et ont été stockées de manière automatique dans le fichier config.xml du répertoire « config » du dossier « server ». Vous pouvez donc retrouver les différentes valeurs que vous avez saisies. Vous pouvez également les modifier pour les prendre en compte au prochain démarrage du serveur. Cependant, il est conseillé de modifier ce fichier en toute connaissance des impacts sur le fonctionnement du serveur d'api.

Le fonctionnement de l'api de signature et paiement est elle-même pilotée par deux fichiers de paramétrage spécifiques : le fichier pathfile et le fichier certificat.

Les fichiers Pathfile et Certificat

- Le fichier **Pathfile** est utilisé lors de l'envoi des requêtes. Ce fichier est paramétré automatiquement lors de l'installation de l'API. Toutefois, si vous décidez de modifier l'emplacement de l'application SIPS Office, l'adresse ou le port du Proxy, vous devrez effectuer des modifications manuellement dans le ficher Pathfile et indiquer le chemin du nouveau répertoire.

  Il est important de ne pas altérer la syntaxe de ce fichier (séparateur).

```
# ------------------------------------------------------------------
#       Paramètres liés aux Proxy
# ------------------------------------------------------------------
# Adresse du Proxy
PROXY_HOST!mon_adresse_proxy!

# Numéro de port du Proxy
PROXY_PORT!mon_numero_de_port_proxy!

# ------------------------------------------------------------------
#       Paramètres liés aux Répertoires
# ------------------------------------------------------------------
#       Répertoire principal de l'application
D_APPLI!C:\home\chemin_de_mon_api\!

#       Répertoire des fichiers certificats
D_CERTIFICATE!param\!

#       Préfixe des fichiers certificats
F_CERTIFICATE!certif!
# ------------------------------------------------------------------
#       end of file
# ------------------------------------------------------------------
```

- Le fichier **certif.fr.xxxxxxxxxxxx** correspond au certificat commerçant. Lors de l'installation de SIPS Office, un certificat de test vous est fourni. Il vous permettra de faire vos tests avant de passer en pré-production, puis en production.

  Ce fichier permet d'assurer les fonctions d'authentification et d'intégrité des messages échangés avec le serveur de signature son format est spécifique à l'API de signature.

  Une fois la procédure d'enregistrement validée, vous recevrez un nouveau fichier certificat, généré pour chacun des commerçants. Ce nouveau certificat sera de la forme suivante : certif.fr.(numéro_de_siret_précédé_du_0)
  où,
  'fr' fait référence au pays du commerçant.

  En France, il s'agit du numéro d'enregistrement du commerçant, soit le numéro de SIRET préédé du 0, par exemple 011223344552222 (en test).

Chapitre

7

Des tests à la production.

La phase de test : Pendant cette période, vous devrez tester, le bon fonctionnement du serveur d'API puis vérifier si vos requêtes sont acheminées vers le serveur SIPS Office. Enfin vérifier qu'en retour la réponse du serveur est valide. Durant cette phase, vous devrez vous servir du certificat de test fourni.

La phase de pré-production : cette phase se caractérise par la préparation à la phase de production. L'utilisateur devra effectuer des modifications sur certains champs de la requête XML qu'il enverra au serveur d'API, et remplacer le certificat de test par son certificat commerçant. Les transactions ne sont pas remises en banque.

La phase de production : cette phase débute uniquement au moment où nos équipes reçoivent le PV de recette dûment rempli et complété.

Phase de test	Phase de pré-production	Phase de production
Pour tester l'intégration de SIPS Office dans le système d'information du commerçant, ATOS ORIGIN met à disposition un **serveur de test**. Cependant, si vous débutez dans l'intégration de notre API, vous pouvez avant tout tester le fonctionnement du serveur d'API (avant même de tester que vos requêtes arrivent sur le serveur de test) avec la fonction **Simulation (cf chapitre 1)**. Ce serveur de test a les mêmes fonctionnalités que le serveur de production sauf pour la demande d'autorisation auprès du serveur de la banque, qu'il simule. Vous devez donc vous servir du **certificat de test** livré avec l'API (par exemple : certif.fr.011223344552222). Ce certificat est générique à tous les commerçants en phase de test. Pour les tests, reportez-vous à l'ensemble de ce document en suivant les exemples donnés.	Pendant cette phase, vous devrez : 1. Remplacer le champ **Merchant_id** dans la **requête XML** envoyée au serveur par le numéro de commerçant qui vous aura été attribué (en général, votre numéro de SIRET précédé de 0). **2.** Modifier le champ **Merchant_country** de la même manière **si la banque du commerçant est étrangère.** 3. Remplacer (c'est à dire au même endroit) le certificat de test par celui qui vous sera attribué. **A ce point précis, vous êtes prêt à passer en phase de production**	Pour passer en production, vous devrez nous envoyer le PV de recette fourni avec l'API. **ATTENTION !** Ce n'est qu'à réception de ce PV, que nous pourrons vous basculer vers la plate-forme de production.

Annexes

Liste des valeurs du champ card_type

Les cartes marquées d'un astérisque (*) n'ont pas de date de fin de validité. Par conséquent, le champ **card_validity** est **vide.**

- **CB**
- **VISA**
- **MASTERCARD**
- **AMEX**
- **AURORE (*) : le champ Data doit être complété par « DATE_NAISSANCE=AAAAMMJJ ;**
- **JCB**
- **DELTA**
- **CONFORAMA**
- **PASS (*) : le champ Data doit être complété par « DATE_NAISSANCE=AAAAMMJJ ; MODE_REGLEMENT_PASS=COMPTANT ou CREDIT ou 3FOIS »**
- **FNAC**
- **PRINTEMPS**
- **SURCOUF (*)**
- **FINAREF (*)**
- **VOYAGEUR**
- **PEUGEOT**
- **PLURIEL**
- **NUITEA**
- **POCKETCARD**
- **KANGOUROU (*)**
- **CYRILLUS (*)**
- **SWITCH**
- **SOLO**

Liste des valeurs du champ currency_code

Vous trouverez ci-dessous les codes devises supportés par le serveur d'API. Cette liste est extraite de la norme ISO 4217 sur la représentation des monnaies et type de fonds, mais ne constitue en rien le document officiel de l'Organisation Internationale de Normalisation que vous trouverez sur le site officiel : **www.iso.ch**.

Remarque : les monnaies marquées d'un astérisque (*) sont indivisibles.

Code devise alpha	Code devise numérique	Nombre de décimales	Nom de la devise
ARS	032	2	PESO ARGENTIN
ATS	040	2	CHILLING
AUD	036	2	DOLLAR AUSTRALIEN
BEF	056	0	FRANC BELGE (*)
BRL	986	2	REAL
CAD	124	2	DOLLAR CANADIEN
CHF	756	2	FRANC SUISSE
DEM	280	2	DEUTSCHE MARK
DKK	208	2	COURONNE DANOISE
ESP	724	0	PESETA ESPAGNOL (*)
EUR	978	2	EURO
FIM	246	2	MARK FINLANDAIS
FRF	250	2	FRANC
GBP	826	2	LIVRE STERLING
GRD	300	0	DRACHME GREC (*)
IEP	372	2	LIVRE IRLANDAISE
ITL	380	0	LIRE (*)
JPY	392	0	YEN (*)
KHR	116	2	RIEL
KRW	410	2	WON
LUF	442	0	FRANC LUXEMBOURGEOIS (*)
MXN	484	0	PESO MEXICAIN
NLG	528	2	FLORIN NEERLANDAIS
NOK	578	2	COURONNE NORVEGIENNE
NZD	554	2	DOLLAR NEO-ZELANDAIS
PTE	620	2	ESCUDO PORTUGAIS
SEK	752	2	COURONNE SUEDOISE
SGD	702	2	DOLLAR DE SINGAPOUR
TRL	792	2	LIVRE TURQUE
TWD	901	2	NOUVEAU DOLLAR DE TAIWAN
USD	840	2	DOLLAR DES ETATS-UNIS

Liste des valeurs du champ merchant_country

Cette liste est extraite de la norme ISO 3166 sur la représentation des codes pays (sur 2 caractères).

en	England	da	Denmark
fr	France	fi	Finland
ge	Germany	sw	Sweden
it	Italy	po	Portugal
sp	Spain	lu	Luxembourg
be	Belgium	gr	Greece

Liste des valeurs du champ new_status

Opération SIPS Office	Status de la transaction avant opération	Status de la transaction après opération
Validation	TO_VALIDATE	TO_CAPTURE CAPTURED si carte JCB
	TO_REPLAY	TO_CAPTURE si response_code = 0 TO_REPLAY si response_code = 90 REFERRAL si response_code = 02 REFUSED si response_code <> 0
Annulation (totale ou partielle)	TO_AUTHORIZE	TO_AUTHORIZE si annulation partielle sur transaction à to_authorize (New_amount <> 0)
	TO_CAPTURE	TO_CAPTURE si annulation partielle sur transaction à TO_CAPTURE (New_amount <> 0) CANCELLED si annulation totale (New_amount = 0)
Remboursement	CAPTURED	TO_CREDIT CAPTURED si remboursement partiel pour carte Aurore, JCB ou PASS CREDITED si remboursement total pour carte Aurore, JCB ou PASS
	TO_CREDIT	TO_CREDIT
Forçage	REFERRAL	TO_CAPTURE TO_VALIDATE si CAPTURE_MODE = VALIDATION CAPTURED carte JCB
Clonage	*	*

Liste des valeurs du champ New_status (suite)

Valeurs de Status pour une demande d'autorisation :

Le champ Status varie selon les valeurs de Response_code, capture_mode et capture_day de la demande d'autorisation.

Valeur des champs de la demande d'autor			**Valeur de Status**
response_code	*capture_mode*	*capture_day*	
0	AUTHOR_ CAPTURE Ou non renseigné	<= 6 jours	**TO_CAPTURE**
			ENDED pour certaines transactions CyberComm (voir remarque)
			CAPTURED si carte non remisée par SIPS mais remboursement possible (PASS, JCB et Aurore).
		> 6 jours	**TO_AUTHORIZE**
	VALIDATI ON	<= 6 jours	**TO_VALIDATE**
		> 6 jours	**TO_REPLAY**
<> 0	*	*	**REFUSED**
02			**REFERRAL**

Liste des valeurs du champ response_code

Il s'agit des valeurs du champ Response_code selon les opérations réalisées.

ANNULATION	
00	Annulation OK
12	Transaction invalide : un des champs est invalide (ex : le code devise est inconnu)
24	Opération impossible (erreur statut) : l'état de la transaction ne permet pas une annulation
25	Transaction non trouvée : la référence de la transaction n'existe pas dans la base. L'un des paramètres suivants n'est pas valide : merchant_country, merchant_id, payment_date, transaction_id
40	Fonction non supportée : les droits à l'opération d'annulation n'ont pas été donnés au commerçant
51	Montant trop important : montant annulé supérieur au solde de la transaction
63	Erreur de sécurité : erreur détectée lors du déchiffrement des données. L'opération est annulée.
90	Problème temporaire sur serveur banque (uniquement dans le cas d'une carte PASS)
99	Problème temporaire du serveur SIPS Office

VALIDATION	
00	Validation OK
02	« referral » (uniquement si la validation a donné lieu à une nouvelle demande d'autorisation)
05	Validation refusée (uniquement si la validation a donné lieu à une nouvelle demande d'autorisation)
12	Transaction invalide : un des champs est invalide (ex : le code devise est inconnu)
24	Opération impossible (erreur statut) : l'état de la transaction ne permet pas une validation
25	Transaction non trouvée : la référence de la transaction n'existe pas dans la base. L'un des paramètres suivants n'est pas valide : merchant_country, merchant_id, payment_date, transaction_id
40	Fonction non supportée : les droits à l'opération de validation n'ont pas été donnés au commerçant
51	Montant trop important : montant validé supérieur au montant de la transaction
63	Erreur de sécurité : erreur détectée lors du déchiffrement des données
90	Problème temporaire sur serveur banque (uniquement si la validation a donné lieu à une nouvelle demande d'autorisation)
99	Problème temporaire du serveur SIPS Office

REMBOURSEMENT	
00	Remboursement OK
05	Remboursement refusé (uniquement carte PASS et AURORE)
12	Transaction invalide (un des champs est invalide) ex : le code devise est inconnu)
24	Opération impossible (erreur statut) l'état de la transaction ne permet pas un remboursement
25	Transaction non trouvée : la référence de la transaction n'existe pas dans la base. L'un des paramètres suivants n'est pas valide : merchant_country, merchant_id, payment_date, transaction_id
40	Fonction non supportée : les droits à l'opération de remboursement n'ont pas été donnés au commerçant
63	Erreur de sécurité : lors du déchiffrement du message, le serveur a détecté une erreur
90	Problème temporaire sur le serveur banque (uniquement pour les cartes PASS et AURORE)
99	Problème temporaire du serveur SIPS Office

AUTORISATION	
00	Autorisation OK
02	« referral »
05	Autorisation refusée
12	Transaction invalide (un des champs est invalide) ex : le code devise est inconnu)
14	Numéro de carte invalide
40	Fonction non supportée : les droits à l'opération autorisation n'ont pas été donnés au commerçant
63	Erreur de sécurité : lors du déchiffrement du message, le serveur a détecté une erreur
90	Problème temporaire sur le serveur banque
94	Transaction dupliquée : une transaction avec les mêmes références existe déjà dans la base
99	Problème temporaire du serveur SIPS Office

FORCAGE	
00	Forçage OK
12	Transaction invalide (un des champs est invalide) Ex : le code devise est inconnu)
24	Opération impossible (erreur statut) l'état de la transaction ne permet pas un forçage
25	Transaction non trouvée : la référence de la transaction n'existe pas dans la base. L'un des paramètres suivants n'est pas valide : merchant_country, merchant_id, payment_date, transaction_id
40	Fonction non supportée : les droits à l'opération forçage n'ont pas été donné au commerçant
63	Erreur de sécurité : lors du déchiffrement du message, le serveur a détecté une erreur
99	Problème temporaire du serveur SIPS Office

DUPLICATION	
00	Duplication OK
02	« referral »
05	Duplication refusée
12	Transaction invalide (un des champs est invalide) ex : le code devise est inconnu)
14	Numéro de carte invalide
24	Opération impossible. L'état de la transaction ne permet pas une duplication
40	Fonction non supportée : les droits à l'opération de duplication n'ont pas été donnés au commerçant
54	La carte est expirée
63	Erreur de sécurité : lors du déchiffrement du message, le serveur a détecté une erreur
90	Problème temporaire sur le serveur banque
94	Transaction dupliquée : une transaction avec les mêmes références existe déjà dans la base
99	Problème temporaire du serveur SIPS Office

Liste des valeurs du champ capture_mode

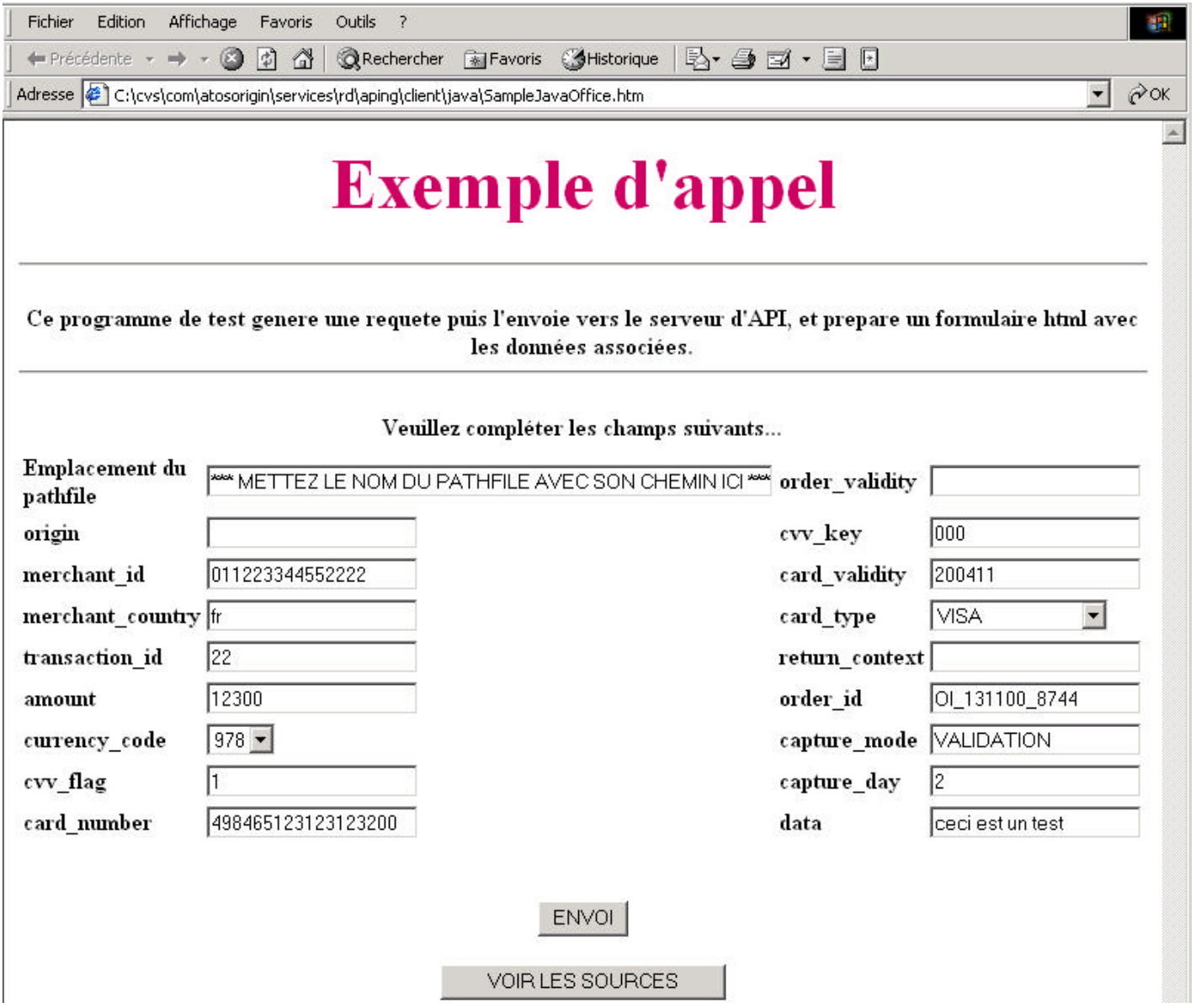
Capture_mode	Capture_day	Description
AUTHOR_CAPTURE	0	C'est le mode par défaut. Le serveur SIPS Office fait une demande d'autorisation en ligne du montant réel. La transaction est envoyée en banque le jour même.
AUTHOR_CAPTURE	n	C'est le mode de paiement différé. **Si n<=6 :** le serveur SIPS Office fait une demande d'autorisation en ligne du montant réel et la transaction est envoyée en banque à jour +n. **Si n>6 :** le serveur fait une demande d'autorisation d'un petit montant (ex : 2 euros). Lors de l'envoi en banque à jour +n, le serveur SIPS Office refait une demande d'autorisation du montant réel.
VALIDATION	n	La transaction est envoyée en banque après validation du commerçant. **Si n<=6** : le serveur fait une demande d'autorisation en ligne du montant réel et le commerçant a n jours pour valider la transaction. La transaction est envoyée en banque le jour de la validation. **Si n>6** : le serveur fait une demande d'autorisation d'un petit montant (ex : 2 euros).Le commerçant a n jours pour valider la transaction. Au moment de la validation, le serveur envoie une demande d'autorisation du montant validé, puis la transaction est envoyée en banque.

Les programmes d'exemple

Chaque fonction correspond à une action précise qui doit être formatée en **XML** selon une syntaxe particulière avant d'être envoyée au serveur d'API. Vous devrez la respecter afin d'appeler correctement le serveur d'API et afin de recevoir une réponse valide.

Ou trouver les programmes d'exemple d'appel en java, perl, asp, php, et C après l' installation ?

Dans le dossier **exemples du CD-ROM**, vous trouverez pour chaque langage un **formulaire HTML** (nom du fichier :… htm) contenant tous les paramètres cités dans le dictionnaire des données, ainsi que les codes sources correspondant au langage utilisé (soit java, asp, perl, php ou C), pour générer la requête XML d'appel au serveur. Pour obtenir une réponse du serveur d'API, il vous suffira de compléter les champs, puis valider. Ce formulaire HTML reprend tous les champs disponibles sur le serveur d'API, et requis (pour le commerçant) pour effectuer une opération sur une ou plusieurs transactions.



Exemple de code java

```
/*-------------------------------------------------
*                - ATOS SERVICES -
* -------------------------------------------------
* (c) Copyright 2001 by Atos Origin. All rights reserved.
*
* This software is the confidential and proprietary
* information of Atos Origin ("Confidential Information").
* You shall not disclose such Confidential Information and
* shall use it only in accordance with the terms of the
* license agreement you entered into with Atos Origin.
* -------------------------------------------------
* Projet     : APING
* Class      : com.atosorigin.services.rd.aping.http.DemoCallOffice
* File       : DemoCallOffice.java
* -------------------------------------------------
* Follow-up  :
* Created by : F Bigot on November 2001
* -------------------------------------------------*/

package com.atosorigin.services.rd.aping.http;

// Imports
import javax.servlet.http.HttpServlet;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;

import org.w3c.dom.*;
import java.io.*;
import java.util.*;
import java.net.*;
import java.util.*;
import java.lang.*;

import com.atosorigin.services.rd.aping.common.Dom;

public class DemoCallOffice extends HttpServlet {

    BufferedReader in;
    PrintWriter out;
    Dom res;
    String nom = "";
    String valeur = "";
    String FinalResult = "";
    String codeHtml =  "<body><center>"+"<td width=50><img src=\"images/blank.gif\" width=50
height=1></td>"+ "<td><center><font face=\"Century Gothic,Century Gothic MT,Century Gothic
MS,Arial,Helvetica\">"+"<font size=+3><i></i><b></b></font></font><hr><center><font
size=+2><strong>"+"REPONSE DU SERVEUR </strong></font></ center></ center><br
clear=all><center>"+ "<P><font face=\"Century Gothic,Century Gothic MT,Century Gothic
```

```
MS,Arial,Helvetica\">"+"<font size=+2></font></font><dir><P><P><table border=0><tr><td
valign=top>";


    public void service(HttpServletRequest  request,
            HttpServletResponse response)
                    throws ServletException, IOException{

      try{
          /*ouverture du  socket vers le serveur d'API*/

          Socket sock = new Socket("localhost",83);


          BufferedReader in = new BufferedReader(new InputStreamReader(sock.getInputStream()));
          PrintWriter serverStream = new PrintWriter(sock.getOutputStream(), true);


          /*on envoie la requete xml avec les parametres recupérés dans le formulaire HTML*/

          serverStream.println("<service component=\"requestOffice\" name=\"author\">" +
                    "<author pathfile=\""        +request.getParameter("pathfile")+"\""+
                    "origin=\""                  +request.getParameter("origin")+"\" " +
                    "merchant_id=\""             +request.getParameter("merchant_id")+"\" " +
                    "merchant_country=\""        +request.getParameter("merchant_country")+"\" "+
                    "transaction_id=\""          +request.getParameter("transaction_id")+"\" " +
                    "amount=\""                  +request.getParameter("amount")+"\" "+
                    "currency_code=\""           +request.getParameter("currency_code")+"\" " +
                    "card_number=\""             +request.getParameter("card_number")+"\" "+
                    "cvv_flag=\""                +request.getParameter("cvv_flag")+"\" " +
                    "cvv_key=\""                 +request.getParameter("cvv_key")+"\" "+
                    "card_validity=\""           +request.getParameter("card_validity")+"\" "+
                    "card_type=\""               +request.getParameter("card_type")+"\" " +
                    "return_context=\""          +request.getParameter("return_context")+"\" "+
                    "order_id=\""                +request.getParameter("order_id")+"\" " +
                    "capture_mode=\""            +request.getParameter("capture_mode")+"\" "+
                    "capture_day=\""             +request.getParameter("capture_day")+"\" " +
                    "data=\""                    +request.getParameter("data")+"\" "+
                    "order_validity=\""          +request.getParameter("order_validity")+"\" />" +
                    "</service>");


          /*on lit la réponse du serveur, et on la parse*/

          String result = in.readLine();
          res = new Dom(result,false);
          int nb = res.getNbAttributs("response");

          /*on parse la reponse du serveur, puis on construit la page html*/

          for(int i=0; i < nb; i++)
          {
           nom = res.getElementAttributName("response",i+1);
           valeur = res.getElementAttribut("response",1,nom);
           FinalResult = FinalResult +"<tr><td><strong>"+ nom + "</strong>" +
            ":\n</ td>" +"<td>"+ valeur+"</ td></tr>";
```

```
            }

    sock.close();

    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.println("<head>");
    out.println("<title>Reponse du serveur</ title>");
    out.println("</head>");
    out.println("<body><center>");
    out.println(codeHtml);
    out.println(result);
    out.println("<table border=0>");
    out.println(FinalResult);
    out.println("</table><br><br>");
    out.println("</ center></body>");
    out.close();
    FinalResult = "";

    }catch (Exception e){}
  }
}
```

Exemple de code perl

#ATTENTION : certains paramètres tels que PeerAddr, PeerPort, le chemin d'accès à l'interpréteur Perl sont à modifier selon votre environnement. Ce programme est un CGI. Il devra être situé dans le dossier cgi-bin de votre serveur Web.

```
#!c:/perl/bin/perl.exe

use XML::Parser;
use IO::Socket;
$EOL = "\015\012";
$BLANK = $EOL x 2;


 # récupère l'entrée standard STDIN et la place dans la variable $in
   read(STDIN, $in, $ENV{CONTENT_LENGTH});

 # la chaîne $in est coupée suivant le caractère & et crée la liste @champs
   @champs = split(/&/,$in);

 # traitement de chaque élément $e de la liste @champs
   foreach $e (@champs) {

# dissocie chaque élément, de la forme nom=valeur,
# en une paire de variable (nom,valeur)
   ($nom, $valeur) = split(/=/,$e);

 # On reconvertit les caractères codes %ab
 $valeur =~ s/%(..)/pack("C", hex($1))/eg;

 # puis des espaces codés (+ en espace)
 $valeur =~ tr/+/ /;

 # crée à partir du tableau @champs,
 # une liste associative %champs
   $champs{$nom}=$valeur;
   }

# génère l'en-tête du document HTML renvoyé
   print("Content-Type: text/html\n\n");

 # puis le document HTML
   print "<body><center><td width=50></td>";
   print "<td><center><font face=\"Century Gothic,Century Gothic MT,Century Gothic
   MS,Arial,Helvetica\">";
   print "<font size=+3><i></i><b></b></font></font><hr><center><font
   size=+2><strong>";
   print "REPONSE DU SERVEUR </strong></font></center></center><br
   clear=all><center>";
```

```
print "<P><font face=\"Century Gothic,Century Gothic MT,Century Gothic
MS,Arial,Helvetica\">";
print "<font size=+2></font></font><dir><P><P><table border=0><tr><td
valign=top>";
```

#création du socket et envoi des données vers le serveur d'API

```
$remote = IO::Socket::INET->new(PeerAddr => '104.3.2.5',
                  PeerPort  => '7180',
                  Proto     => 'tcp');

die "Could not create socket: $!\n" unless $remote;

print $remote "<service component=\"requestOffice\" name=\"author\"><author
pathfile=\"$champs{'pathfile'}\" origin=\"$champs{'origin'}\" merchant_id
=\"$champs{'merchant_id'}\" merchant_country =\"$champs{'merchant_country'}\"
transaction_id =\"$champs{'transaction_id'}\" amount=\"$champs{'amount'}\"
currency_code=\"$champs{'currency_code'}\" card_number=\"$champs{'card_number'}\"
cvv_flag=\"$champs{'cvv_flag'}\" cvv_key=\"$champs{'cvv_key'}\"
card_validity=\"$champs{'card_validity'}\" card_type=\"$champs{'card_type'}\"
return_context=\"$champs{'return_context'}\" order_id=\"$champs{'order_id'}\"
capture_mode=\"$champs{'capture_mode'}\" capture_day=\"$champs{'capture_day'}\"
data=\"$champs{'data'}\" order_validity=\"$champs{'order_validity'}\"/></service>" .
$BLANK;
```

#instantiation du parseur afin de lire la réponse

```
my $parser = new XML::Parser(ErrorContext => 2);

$parser->setHandlers(Start => \&start_handler,
      End   => \&end_handler,
            Char  => \&char_handler);

$parser->parse(<$remote>);


close $remote;

# ----------------------------------------
#
# The handlers for the XML Parser.
# ----------------------------------------

sub start_handler
 {
  my $expat = shift; my $element = shift;
  print ;
```

```
# Handle the attributes
while (@_) {
        my $att = shift;
         my $val = shift;
         print "<tr><td><strong>$att:</strong></td><td>$val</td></tr>";

} }

sub end_handler
{
my $expat = shift; my $element = shift;
print ;
print "</center></body>".$BLANK;

}
sub char_handler
{
my ($p, $data) = @_;
print ;
}
```

Exemple de code ASP

Cet exemple utilise un composant ActiveX (W3 socket http://www.dimac.net) pour effectuer une connexion IP. Le parser XML utilisé est celui fourni en standard par Microsoft (MSXML)

```
<%@ LANGUAGE = VBScript%>

<HTML><HEAD><TITLE>Test d'API</TITLE>
<BODY bgColor=#ffffff>
<CENTER>
<FONT color=#009900 size=6>resultat appel API SIPS</FONT></CENTER>
<HR>
<%
dim XMLInput
dim HTMLInput
dim printbuffer

function XML2HTML(XMLInput)
        intLongueur = len(XMLInput)
        for intcptChar = 1 to intLongueur
        testChar = mid(XMLInput, intcptChar, 1)
                select case testChar
                        case ">"        XML2HTML = XML2HTML & "<br>"
                        case "<"        XML2HTML = XML2HTML
                        case else       XML2HTML = XML2HTML + testChar
                end select
        next
end function


pathfile  = Request.form("pathfile")
order_validity    = Request.form("order_validity")
origin            = Request.form("origin")
cvv_key           = Request.form("cvv_key")
merchant_id       = Request.form("merchant_id")
card_validity     = Request.form("card_validity")
merchant_country = Request.form("merchant_country")
card_type         = Request.form("card_type")
transaction_id    = Request.form("transaction_id")
return_context    = Request.form("return_context")
amount            = Request.form("amount")
order_id          = Request.form("order_id")
currency_code     = Request.form("currency_code")
capture_mode      = Request.form("capture_mode")
cvv_flag          = Request.form("cvv_flag")
capture_day       = Request.form("capture_day")
card_number       = Request.form("card_number")
data              = Request.form("data")
```

```
XMLInput = "<service component=""requestOffice"" name = ""simul"">"
XMLInput = XMLInput & "<author pathfile="""       & pathfile & """"
XMLInput = XMLInput & " origin="""                & origin & """"
XMLInput = XMLInput & " merchant_id="""           & merchant_id & """"
XMLInput = XMLInput & " merchant_country="""      & merchant_country & """"
XMLInput = XMLInput & " transaction_id="""        & transaction_id & """"
XMLInput = XMLInput & " amount="""                & amount & """"
XMLInput = XMLInput & " currency_code="""         & currency_code & """"
XMLInput = XMLInput & " card_number="""           & card_number & """"
XMLInput = XMLInput & " cvv_flag="""              & cvv_flag & """"
XMLInput = XMLInput & " cvv_key="""               & cvv_key & """"
XMLInput = XMLInput & " card_validity="""         & card_validity & """"
XMLInput = XMLInput & " card_type="""             & card_type & """"
XMLInput = XMLInput & " return_context="""        & return_context & """"
XMLInput = XMLInput & " order_id="""              & order_id & """"
XMLInput = XMLInput & " capture_mode="""          & capture_mode & """"
XMLInput = XMLInput & " capture_day="""           & capture_day & """"
XMLInput = XMLInput & " data="""                  & data & """"
XMLInput = XMLInput & " order_validity="""        & order_validity & """"
XMLInput = XMLInput & "/></service>"


HTMLInput = XML2HTML(XMLInput)
response.write ("requete = <br>" & HTMLInput)

REM -- XMLInput = "<service component =""requestOffice" name =""author"">"
REM -- Partie connection serveur IP pour génération de la reponse API
dim socket
dim buffer

response.write ("<br><b>connexion serveur</b>")
set socket = server.createObject("Socket.TCP")
socket.timeout = 1000000
socket.host = "127.0.0.1:7180"
socket.open()

socket.Sendline( XMLInput )
socket.WaitForDisconnect()

buffer = socket.Buffer
response.write("<br>")

HTMLInput = XML2HTML(buffer)
response.write ("reponse = <br>" & HTMLInput)

socket.close()
response.write ("<br><b>fin communication</b><br><br>")
set objXML = server.createObject("microsoft.xmldom")
objXML.loadXML(buffer)
```

```
Response.write("code retour analyse buffer : " & objXML.parseError.errorCode &"<br><br>")

if objXML.parseError.errorCode = 0 then
        Response.Write(" <b>date de transaction</b>="&
objXML.documentElement.getAttribute("transaction_date") &"<br>")
        Response.Write(" <b>heure de transaction</b>=" &
objXML.documentElement.getAttribute("transaction_time") &"<br>")
        Response.Write(" <b>code réponse</b>= " &
objXML.documentElement.getAttribute("response_code") &"<br>")
        Response.Write(" <b>montant</b>= " &
objXML.documentElement.getAttribute("new_amount") &"<br>")
        Response.Write(" <b>code devise</b>= " &
objXML.documentElement.getAttribute("currency_code") &"<br>")
        Response.Write(" <b>numéro autorisation</b>= " &
objXML.documentElement.getAttribute("authorization_id") &"<br>")
        Response.Write(" <b>certificat transaction</b>= " &
objXML.documentElement.getAttribute("transaction_certificate") &"<br>")
        Response.Write(" <b>nouveau status</b>= " &
objXML.documentElement.getAttribute("new_status") &"<br>")
end if
%>
<HR>
</HEAD>

</HTML>
```

Exemple de code en php

```
<?

 // a paramétrer
  $port = 7180;
  $adress = "localhost";

 // on récupère les données du formulaire
  $buffer = "<service component="."\""."requestOffice"."\""." name="."\""."simul"."\""."">
                <simul pathfile="."\"".$pathfile."\""." origin="."\"".$origin."\""."
                merchant_id="."\"".$merchant_id."\""."
                merchant_country="."\"".$merchant_country."\""."
                transaction_id="."\"".$transaction_id."\""." amount="."\"".$amount."\""."
                currency_code="."\"".$currency_code."\""."
                card_number="."\"".$card_number."\""." cvv_flag="."\"".$cvv_flag."\""."
                cvv_key="."\"".$cvv_key."\""." card_validity="."\"".$card_validity."\""."
                card_type="."\"".$card_type."\""." return_context="."\"".$retur n_context."\""."
                order_id="."\"".$order_id."\""." capture_mode="."\"".$capture_mode."\""."
                capture_day="."\"".$capture_day."\""." data="."\"".$data."\""."
                order_validity="."\"".$order_validity."\""."/></service>\n";

  echo "<center><strong><font size=+2>ceci est la réponse du serveur
  \n<br><br></font></strong></center>".$buffer;


 // on ouvre un socket vers le serveur et envoie les données
  $sock = fsockopen($adress, $port, &$errno, &$errstr, 10);

  if(!$sock) {
                echo "Connexion failed\n<br>";
                echo "erreur.$errstr ($errno)<br>\n";


   } else {

                // on écrit dans le socket
                fputs($sock,$buffer);

                // on lit dans le socket
                $result = fgets($sock,1024);

                //puis, on convertit les caractères spéciaux en HTML
                echo HTMLSpecialChars($result);
                fclose($sock);
     }

?>
```

INDEX

T

V

X