# Définition des Webservices Standards Systempay 1.24.1

Version 2.6e

# Rédaction, Vérification, Approbation

| Rédaction | | Vérification | | Approbation | |
|---|---|---|---|---|---|
| **Nom** | **Date/Visa** | **Nom** | **Date/Visa** | **Nom** | **Date/Visa** |
| Lyra-Network | 21/01/2013 | Lyra-Network | 21/01/2013 | Lyra-Network | 21/01/2013 |

# Historique du document

| Version | Auteur | Date | Commentaires |
|---|---|---|---|
| 2.6e | Lyra-Network | 21/01/2013 | Ajout de l'errorCode 26.<br>Nouveau comportement lors du remboursement d'une carte expirée<br>Correction de l'encodage dans l'exemple JAVA<br>Ajout des codes renvoyés dans le champ authResult.<br>Précision apportées pour le champ cvv |
| 2.6d | Lyra-Network | 10/01/2013 | localControl : Ajout de nouvelles valeurs<br>Précision apportée concernant l'erreurCode 15.<br>create : Précisions apportées sur la valorisation de paymentMethod |
| 2.6c | Lyra-Network | 14/12/2012 | Duplicate : Précisions sur les statuts autorisés |
| 2.6b | Lyra-Network | 26/10/2012 | Maintien de la session HTTP lors de l'enchainement de plusieurs requêtes |
| 2.6a | Lyra-Network | 17/10/2012 | Ajout des caractéristiques du champ transactionId |
| 2.6 | Lyra-Network | 24/04/2012 | Ajout d'un exemple de la méthode create<br>Précisions sur les champs du transactionInfo et createPaiementInfo<br>Précision sur la valeur du champ transmissionDate |
| 2.51 | Lyra-Network | 06/04/2012 | Précisions sur dates d'expiration |
| 2.5 | Lyra-Network | 20/10/2011 | Précisions apportées pour la prise en compte des champs de type date en UTC lors du calcul de signature.<br>Précisions apportées sur les codes d'erreur |
| 2.4 | Lyra-Network | 31/12/2010 | Ajout modifyAndValidate |
| 2.3 | Lyra-Network | 29/11/2010 | Ajout champs pour paiement 3DS<br>Ajout champ impayés |
| 2.2 | Lyra-Network | 03/09/2010 | Ajout description champ date |
| 2.1 | Lyra-Network | 31/03/2010 | Modification valorisation champ PaymentMethod |
| 2.0 | Lyra-Network | 31/03/2010 | Ajout champs Infos + contrat. |
| 1.1 | Lyra-Network | 05/03/2010 | Version avec signature. |
| 1.0 | Lyra-Network | 04/09/2009 | Version initiale. |

Confidentialité

Toutes les informations contenues dans ce document sont considérées comme confidentielles.
L'utilisation de celles-ci en dehors du cadre de cette consultation ou la divulgation à des personnes extérieures est soumise à l'approbation préalable de
Lyra Network.

# SOMMAIRE

# 1. Présentation

Ce document présente les webservices standards qui permettent d'automatiser les actions réalisables manuellement depuis l'outil de gestion de caisse commerçant.

Ces webservices ont été développés suivant le protocole SOAP (Simple Object Access Protocol) et sont décrits par le fichier wsdl suivant :

https://paiement.systempay.fr/vads-ws/v3?wsdl

Afin de sécuriser les échanges, les webservices (SOAP) sont cryptés grâce au protocole HTTPS. De plus un mécanisme de signature a été mis en place afin de valider et d'authentifier l'échange des données.

Avant de commencer votre implémentation, veuillez-vous assurer que l'offre souscrite auprès de Systempay comprend l'utilisation des webservices.

# 2. Description des types

En dehors des types simples (int, string, date, …), les webservices utilisent des types plus complexes (code retour, description de la transaction, …) qui sont décrits dans ce chapitre.

## 2.1. Date

Les champs 'date' doivent suivre les recommandations W3C (http://www.w3.org/TR/NOTE-datetime). Par exemple pour une date d'expiration en décembre 2012, le format correct devrait-être 2012-12-31T00:00:00+00:00.
**Attention** :
Lors du calcul de signature, le format des champs de type 'date' est YYYYmmDD.
(cf. chapitre 4)

## 2.2. standardResponse

Ce type permet de décrire la réponse de la plupart des webservices.

| Nom du champ | Type | Description |
|---|---|---|
| errorCode | int | Code d'erreur (cf. 2.7) |
| extendedErrorCode | String | Précision sur le code d'erreur |
| transactionStatus | int | Statut de la transaction |
| timestamp | long | Timestamp permettant la génération de signature unique |
| signature | String | Signature de la transaction (cf. 3.9) |

Les différentes valeurs des codes d'erreurs et codes d'erreurs étendus seront précisées pour chacun des webservices. Les différents statuts de la transaction peuvent être :

| Valeur | Description |
|---|---|
| 0 | Initial (en traitement) |
| 1 | A valider |
| 2 | A forcer – Contacter l'émetteur |
| 3 | A valider et autoriser |
| 4 | En attente de remise |
| 5 | En attente d'autorisation |
| 6 | Remisée |
| 7 | Expirée |
| 8 | Refusée |
| 9 | Annulée |
| 10 | En attente |
| 11 | En cours de remise |
| 12 | En cours d'autorisation |
| 13 | En échec |

La signature permet de valider l'intégrité de la réponse, le calcul de cette signature se fait en prenant les paramètres dans l'ordre suivant :
errorCode, extendedErrorCode, transactionStatus, timestamp

## 2.3. localControl

Ce type permet de décrire un contrôle local (nom du contrôle et résultat).

| Nom du champ | Type | Description |
|---|---|---|
| name | String | Nom du contrôle |
| result | Bool | Résultat du contrôle |

Les différentes valeurs possibles pour 'name' sont :

| Valeur | Description |
|---|---|
| "CARD" | Carte située dans une liste grise |
| "COUNTRY" | Pays inclus dans la liste grise de la boutique ou absent de la liste blanche |
| "IPADDR" | Adresse IP située dans une liste grise |
| "AMOUNT" | Encours atteint |
| "BIN" | Le code BIN appartient à la liste grise du commerçant |
| " ECB" | Détection d'une e-carte bleue |
| "CARD_COMMERCIAL_NATIONAL" | Détection d'une carte commerciale nationale |
| "CARD_COMMERCIAL_FOREIGN" | Détection d'une carte commerciale étrangère |
| "CAS" | Détection d'une carte à autorisation systématique |
| "COUNTRY_CONSISTENCY" | Aucun pays ne correspond (pays IP, pays carte, pays client) |
| "NON_GUARANTEED_PAYMENT" | Détection d'un paiement sans garantie |
| "IPADDR_COUNTRY" | Le pays de l'adresse IP appartient à la liste grise |

Cette liste est susceptible de s'allonger, veuillez bien en tenir compte dans votre implémentation.

Les différentes valeurs possibles pour 'result' sont :

| Valeur | Description |
|---|---|
| "0" | Indique que le contrôle est correct |
| "1" | Indique que le contrôle est en erreur |

## 2.4. transactionInfo

Ce type permet de décrire une transaction.

| Nom du champ | Type | Description |
|---|---|---|
| | | Réponse générale |
| errorCode | Int | Code d'erreur (cf. 2.7) |
| extendedErrorCode | String | Précision sur le code d'erreur |
| transactionStatus | Int | Statut de la transaction (cf. 2.2 pour le détail) |
| | | Détail transaction |
| shopId | String | Identifiant de la boutique |
| paymentMethod | String | Source du paiement :<br>- "E_COMMERCE" : e-Commerce<br>- "MAIL_OR_TELEPHONE" : mail ou téléphone<br>- "CALL_CENTER" : centre d'appel<br>- "OTHER" : autres |
| contractNumber | String | N° de contrat commerçant |
| orderId | String | Référence de la commande |
| orderInfo | String | Description libre de la commande |
| orderInfo2 | String | Description libre de la commande |
| orderInfo3 | String | Description libre de la commande |
| transmissionDate | Date | Date et heure de la transaction |
| transactionId | String | Identifiant de transaction |
| sequenceNb | Int | Numéro de séquence de la transaction |
| amount | Long | Montant actuel de la transaction en plus petite unité monétaire |
| initialAmount | Long | Montant initial (avant modification éventuelle) en plus petite unité monétaire |
| devise | Int | Devise (Code monnaie ISO 4217, Euro : 978) |
| cvAmount | Long | Montant en contre-valeur en plus petite unité monétaire |
| cvDevise | Int | Devise en contre-valeur (Code monnaie ISO 4217, Euro : 978) |
| presentationDate | Date | Date de remise demandée |
| type | Int | 0 = DEBIT, 1 = CREDIT |
| multiplePaiement | Int | Paiement en plusieurs fois (Non = 0, Oui = 1) |
| ctxMode | String | Contexte de sollicitation de la plateforme de paiement ("TEST", "PRODUCTION") |
| | | Détails carte |
| cardNumber | String | Numéro de carte |
| cardNetwork | String | Réseau de carte |
| cardType | String | Type de carte |
| cardCountry | Int | Pays d'émission de la carte (Code numérique ISO 3166-1 ex : France=250) |
| cardExpirationDate | Date | Date expiration de la carte |
| | | Détails porteur |
| customerId | String | Code client |
| customerTitle | String | Civilité |
| customerName | String | Nom client |
| customerPhone | String | Téléphone client |
| customerMail | String | Mail client |
| customerAddress | String | Adresse client |
| customerZipCode | String | Code postal client |
| customerCity | String | Ville client |
| customerCountry | String | Pays client |
| customerLanguage | String | Langue client composée du code langue (ISO 639-1) et du code pays (ISO-3166) séparés par le caractère « _ » . Exemple : fr_FR |
| customerIP | String | Adresse IP |

| Détails Authentification 3D-Secure | | |
|---|---|---|
| transactionCondition | String | " COND_3D_SUCCESS", "COND_3D_FAILURE", "COND_3D_ERROR", "COND_3D_NOTENROLLED", "COND_3D_ATTEMPT", "COND_SSL" (voir detail ci-dessous) |
| vadsEnrolled | String | Enrôlement porteur 3DS |
| vadsStatus | String | Authentification porteur |
| vadsECI | String | Indicateur de commerce électronique |
| vadsXID | String | Identifiant transaction 3DS |
| vadsCAVV | String | Informations relatives au traitement du cryptogramme de commerce électronique |
| vadsCAVVAlgorithm | String | Méthode de calcul du cryptogramme de commerce électronique |
| vadsSignatureValid | String | Signature de l'authentification |
| directoryServer | String | Réseau dont le Directory Server a été contacté |
| Détails de l'autorisation | | |
| authMode | String | Empreinte ou non<br>"MARK" : avec empreinte<br>"FULL" : sans empreinte |
| markAmount | Long | Montant de l'empreinte en plus petite unité monétaire |
| markDevise | Int | Devise de l'empreinte (Code monnaie ISO 4217, Euro : 978) |
| markDate | Date | Date de l'empreinte |
| markNb | String | Numéro d'auto de l'empreinte |
| markResult | Int | Résultat de l'empreinte |
| markCVV2_CVC2 | String | *Information relative au traitement du cryptogramme visuel de l'empreinte* |
| authAmount | Long | Montant de l'autorisation en plus petite unité monétaire |
| authDevise | Int | Devise de l'autorisation (Code monnaie ISO 4217, Euro : 978) |
| authDate | Date | Date de l'autorisation |
| authNb | String | Numéro de l'autorisation |
| authResult | Int | Résultat de l'autorisation |
| authCVV2_CVC2 | String | *Information relative au traitement du cryptogramme visuel de l'autorisation* |
| Détails Garantie & Contrôles locaux | | |
| warrantlyResult | String | Garantie de paiement |
| localControl | Array <LocalControl> | Tableau des résultats des différents contrôles locaux (cf. 2.3) |
| Détails Remise (renseignés uniquement si la transaction a été remisée) | | |
| captureDate | Date | Date et heure de remise |
| captureNumber | Int | Numéro de remise |
| rapprochementStatut | Int | Statut de rapprochement bancaire de la transaction. |
| refundAmount | Long | Montant ayant déjà fait l'objet d'un remboursement en plus petite unité monétaire |
| refundDevise | Int | Devise du montant ayant déjà fait l'objet d'un remboursement (Code monnaie ISO 4217, Euro : 978) |
| litige | Bool | Litige |
| timestamp | long | Timestamp permettant la génération de signature unique |
| signature | String | Signature de la transaction (cf. 3.9) |

Liste des valeurs possibles pour le champ **authResult et markResult**.

| Valeur | Description |
|---|---|
| 00 | transaction approuvée ou traitée avec succès |
| 02 | contacter l'émetteur de carte |
| 03 | accepteur invalide |
| 04 | conserver la carte |
| 05 | ne pas honorer |
| 07 | conserver la carte, conditions spéciales |
| 08 | approuver après identification |
| 12 | transaction invalide |
| 13 | montant invalide |
| 14 | numéro de porteur invalide |
| 30 | erreur de format |
| 31 | identifiant de l'organisme acquéreur inconnu |
| 33 | date de validité de la carte dépassée |
| 34 | suspicion de fraude |
| 41 | carte perdue |
| 43 | carte volée |
| 51 | provision insuffisante ou crédit dépassé |
| 54 | date de validité de la carte dépassée |
| 56 | carte absente du fichier |
| 57 | transaction non permise à ce porteur |
| 58 | transaction interdite au terminal |
| 59 | suspicion de fraude |
| 60 | l'accepteur de carte doit contacter l'acquéreur |
| 61 | montant de retrait hors limite |
| 63 | règles de sécurité non respectées |
| 68 | réponse non parvenue ou reçue trop tard |
| 90 | arrêt momentané du système |
| 91 | émetteur de cartes inaccessible |
| 96 | mauvais fonctionnement du système |
| 94 | transaction dupliquée |
| 97 | échéance de la temporisation de surveillance globale |
| 98 | serveur indisponible routage réseau demandé à nouveau |
| 99 | incident domaine initiateur |

Détails transactionCondition:

| Valeur | Description |
|---|---|
| "COND_3D_SUCCESS" | Le commerçant et le porteur de la carte sont inscrits au programme 3-D Secure et le porteur s'est authentifié correctement. |
| " COND_3D_FAILURE" | Le commerçant et le porteur de la carte sont inscrits au programme 3-D Secure mais l'acheteur n'a pas réussi à s'authentifier (mauvais mot de passe) |
| " COND_3D_ERROR" | Le commerçant participe au programme 3-D Secure mais le serveur Systempay a rencontré un problème technique durant le processus d'authentification (lors de la vérification de l'inscription de la carte au programme 3D ou de l'authentification du porteur). |
| " COND_3D_NOTENROLLED" | Le commerçant participe au programme 3-D Secure mais la carte du porteur n'est pas enrôlée. |
| " COND_3D_ATTEMPT" | Le commerçant et le porteur de la carte sont inscrits au programme 3-D Secure mais l'acheteur n'a pas eu à s'authentifier (le serveur de contrôle d'accès de la banque qui a émis la carte n'implémente que la génération d'une preuve de tentative d'authentification). |
| "COND_SSL" | Le commerçant n'est pas enrôlé à 3D-Secure ou le canal de vente n'est pas couvert par cette garantie. |

La signature permet de valider l'intégrité de la réponse, le calcul de cette signature se fait en prenant les paramètres dans l'ordre suivant :

errorCode, extendedErrorCode, transactionStatus, shopId, paymentMethod, contractNumber, orderId, orderInfo, orderInfo2, orderInfo3, transmissionDate, transactionId, sequenceNb, amount, initialAmount, devise, cvAmount, cvDevise, presentationDate, type, multiplePaiement, ctxMode, cardNumber, cardNetwork, cardType, cardCountry, cardExpirationDate, customerId, customerTitle, customerName, customerPhone, customerMail, customerAddress, customerZipCode, customerCity, customerCountry, customerLanguage, customerIP, transactionCondition, vadsEnrolled, vadsStatus, vadsECI, vadsXID, vadsCAVVAlgorithm, vadsCAVV, vadsSignatureValid, directoryServer, authMode, markAmount, markDevise, markDate, markNb, markResult, markCVV2_CVC2, authAmount, authDevise, authDate, authNb, authResult, authCVV2_CVC2, warrantlyResult, captureDate, captureNumber, rapprochementStatut, refoundAmount, refundDevise, litige, timestamp

## 2.5. createPaiementInfo

Ce type permet de décrire les paramètres pour une création de transaction.

| Nom du champ | Type | Description | Obligatoire |
|---|---|---|---|
| Détail transaction | | | |
| shopId | String | Identifiant de la boutique | ✓ |
| transmissionDate | Date | **Date et heure** de transmission de la transaction | ✓ |
| transactionId | String – longueur 6 | Identifiant de transaction.<br>La longueur est 6 impérativement et le format numérique est obligatoire.<br>L'identifiant doit être unique sur une même journée. | ✓ |
| paymentMethod | String | Source du paiement :<br>- "EC" : e-Commerce<br>- "MOTO" : mail ou téléphone<br>- "CC" : centre d'appel<br>- "OTHER" : autres<br>**Remarque :**<br>La valeur **EC** impose qu'un contrat E-commerce **VADS (ERT 24)** soit associé à votre boutique.<br>Toutes les autres valeurs nécessitent un contrat VAD (ERT 20). Dans le cas contraire, la requête sera rejetée avec un code d'erreur 15 (cf. 2.7) | ✓ |
| orderId | String | Référence de la commande | ✓ |
| orderInfo | String | Description libre de la commande | |
| orderInfo2 | String | Description libre de la commande | |
| orderInfo3 | String | Description libre de la commande | |
| amount | Long | Montant de la transaction en plus petite unité monétaire | ✓ |
| devise | int | Devise (Code monnaie ISO 4217, Euro : 978) | ✓ |
| presentationDate | Date | Date de remise demandée | |
| validationMode | int | 0 = Automatique, 1 = Manuelle | |
| Détail carte | | | |
| cardNumber | String | Numéro de carte | ✓ |
| cardNetwork | String | Réseau de carte ("AMEX", "CB", "MASTERCARD", "VISA", "MAESTRO", "E-CARTEBLEUE") | ✓ |
| cardExpirationDate | Date | Date expiration de la carte.<br>Etant donné que seuls l'année et le mois sont pris en compte, il est conseillé de positionner la valeur au dernier jour du mois à 00:00:00 UTC. | ✓ |
| cvv | String | Cryptogramme visuel.<br>**Ce champ est obligatoire lorsque la carte dispose d'un cryptogramme visuel et que l'internaute l'a saisi.**<br><br>**Remarque :**<br>**Certaines cartes ne possédant pas de CVV, le champ cvv est optionnel dans la méthode create.** | |
| contractNumber | String | Numéro de contrat commerçant.<br>Si ce champ est valorisé, merci de faire attention à fournir le bon numéro de contrat en fonction du réseau (par exemple, le contrat CB ne peut pas être utilisé pour les transactions AMEX) | |
| Paramètres avancés | | | |
| threeDsResult | ThreeDsResult | Cf. 2.6 | |

| | | | |
|---|---|---|---|
| subPaymentType | Integer | Ne pas renseigner | |
| subReference | String | | |
| subPaymentNumber | Integer | | |
| Détail porteur | | | |
| customerId | String | Code client | |
| customerTitle | String | Civilité | |
| customerName | String | Nom client | |
| customerPhone | String | Téléphone client | |
| customerMail | String | Mail client | |
| customerAddress | String | Adresse client | |
| customerZipCode | String | Code postal client | |
| customerCity | String | Ville client | |
| customerCountry | String | Pays client | |
| customerLanguage | String | Langue client (Code ISO 639-1, sur 2 caractères) | |
| customerIP | String | Adresse IP | |
| customerSendEmail | bool | Envoi e-mail client souhaité. | |
| Divers | | | |
| ctxMode | String | Contexte de sollicitation de la plateforme de paiement ("TEST", "PRODUCTION") | ✓ |
| comment | String | Commentaire « libre » | |

## 2.6. threeDsResult

Ce type permet de décrire les paramètres de retour 3DS.

| Nom du champ | Type | Description | Obligatoire |
|---|---|---|---|
| brand | String | Brand de la carte ("VISA" ou "MASTERCARD") | ✓ |
| enrolled | String | Statut enrôlement porteur :<br>• "Y" : Enrôlé<br>• "N" : Non enrôlé<br>• "U" : Inconnu | ✓ |
| authStatus | String | Statut authentification :<br>• "Y" : Authentifié 3DS<br>• "N" : Erreur Authentification<br>• "U" : Authentification impossible<br>• "A" : Essai d'authentification | ✓ |
| eci | String | ECI | |
| xid | String | XID | ✓ |
| cavv | String | CAVV | |
| cavvAlgorithm | String | Algorithme CAVV :<br>• "0" : HMAC<br>• "1" : CVV<br>• "2" : CVV_ATN<br>• "3" : Mastercard SPA | |

## 2.7. errorCode

Les codes d'erreurs **en gras** sont détaillés sur la page suivante.

| Code d'erreur | Description | Code d'erreur | Description |
|---|---|---|---|
| **0** | **Action réalisée avec succès** | 61 | Paramètre 'orderInfo1' invalide |
| **1** | **Action non autorisée** | 62 | Paramètre 'orderInfo2' invalide |
| 2 | Transaction non trouvée | 63 | Paramètre 'orderInfo3' invalide |
| 3 | Transaction pas dans le bon état | 64 | Paramètre 'PaymentMethod' invalide |
| 4 | Transaction existe déjà | 65 | Paramètre 'CardNumber' invalide |
| 5 | Mauvaise signature | 66 | Paramètre 'ContractNumber' invalide |
| 6 | Mauvaise date | 67 | Paramètre 'customerId' invalide |
| 10 | Mauvais montant | 68 | Paramètre 'customerTitle' invalide |
| 11 | Mauvaise devise | 69 | Paramètre 'customerName' invalide |
| 12 | Type de carte inconnu | 70 | Paramètre 'customerPhone' invalide |
| 13 | Paramètre 'date d'expiration' invalide | 71 | Paramètre 'customerMail' invalide |
| 14 | Paramètre 'cvv' invalide | 72 | Paramètre 'customerAddress' invalide |
| **15** | **Contrat inconnu** | 73 | Paramètre 'customerZipCode' invalide |
| 16 | Paramètre 'Numéro de carte' invalide | 74 | Paramètre 'customerCity' invalide |
| 17 | Identifiant non trouvé | 75 | Paramètre 'customerCountry' invalide |
| 18 | Identifiant non valide (Résilié, …) | 76 | Paramètre 'customerLanguage' invalide |
| 19 | Subscription non trouvée | 77 | Paramètre 'customerIp' invalide |
| 20 | Subscription non valide | 78 | Paramètre 'customerSendMail' invalide |
| 21 | Identifiant déjà existant | 79 | Paramètre 'customerMobilePhone' invalide |
| 22 | Création d'identifiant refusé | 80 | Paramètre 'subPaiementType' invalide |
| 23 | Identifiant purgé | 81 | Paramètre 'subReference' invalide |
| 26 | Pas de changement | 82 | Paramètre 'initialAmount' invalide |
| 40 | Plage non trouvée | 83 | Paramètre 'occInitialAMount' invalide |
| 50 | Paramètre 'shopId' invalide | 84 | Paramètre 'effectDate' invalide |
| 51 | Paramètre 'transmissionDate' invalide | 85 | Paramètre 'state' invalide |
| 52 | Paramètre 'transactionId' invalide | 90 | Paramètre 'enrolled' invalide |
| 53 | Paramètre 'ctxMode' invalide | 91 | Paramètre 'authStatus' invalide |
| 54 | Paramètre 'comment' invalide | 92 | Paramètre 'eci' invalide |
| 55 | Paramètre 'AutoNb' invalide | 93 | Paramètre 'xid' invalide |
| 56 | Paramètre 'AutoDate' invalide | 94 | Paramètre 'cavv' invalide |
| 57 | Paramètre 'presentationDate' invalide | 95 | Paramètre 'cavvAlgo' invalide |
| 58 | Paramètre 'newTransactionId' invalide | 96 | Paramètre 'brand' invalide |
| 59 | Paramètre 'validationMode' invalide | 98 | Paramètre 'requestId' invalide |
| 60 | Paramètre 'orderId' invalide | 99 | Autre erreur |

Précisions sur les codes d'erreurs

- **ErrorCode 0 :**

Indique que l'action demandée a été réalisée avec succès, traduisant ainsi que le format de la requête est correct.

Remarque :
Dans le cas d'une création de paiement (méthode create) ce code d'erreur ne doit pas être confondu avec le champ transactionStatus qui est le seul à donner le résultat du paiement.

Ainsi on pourra avoir un errorCode à 0 et un transactionStatus à 8, correspondant à la création d'une transaction dont la demande d'autorisation a été refusée.

- **ErrorCode 1 :**

Indique que vous n'avez pas souscrit une offre Systempay permettant d'utiliser les webservices.

- **ErrorCode 15 :**

Indique un défaut au niveau du contrat commerçant.
Plusieurs cas possibles :

- La valeur transmise dans la requête ne correspond à aucun contrat enregistré sur la boutique (shopId),
- Il n'y a pas de contrat enregistré sur la boutique,
- Le contrat spécifié est clôturé,
- Aucun contrat ne correspond au type de contrat nécessaire pour effectuer le paiement. C'est le cas si vous ne possédez pas de contrat VAD (ERT 20) et que **paymentMethod** est valorisé à **MOTO**, **CC** ou **OTHER** dans votre requête.

# 3. Description des méthodes

## 3.1. cancel

Cette fonction permet d'annuler définitivement une transaction, non encore remisée, disposant d'un des statuts suivants :

- A valider
- A valider et autoriser
- En attente
- En attente d'auto
- En attente de remise

Cette fonction prend en entrée les paramètres suivants :

| Nom du champ | Type | Description | Obligatoire |
|---|---|---|---|
| shopId | String | Identifiant de la boutique | ✓ |
| transmissionDate | Date | Date de transaction | ✓ |
| transactionId | String | Identifiant de transaction | ✓ |
| sequenceNb | Int | Numéro de séquence de la transaction | ✓ |
| ctxMode | String | Contexte de sollicitation de la plateforme de paiement ("TEST", "PRODUCTION") | ✓ |
| comment | String | Commentaire « libre » | |
| wsSignature | String | Signature (cf. 4) | ✓ |

Le calcul de la signature se fait en prenant les paramètres dans l'ordre suivant :
shopId, transmissionDate, transactionId, sequenceNb, ctxMode, comment

Cette fonction retourne une réponse du type **standardResponse** (cf. 2.2).

**NB** : le code d'erreur étendu (extendedErrorCode) est renseigné uniquement lorsque l'errorCode est 5 (Mauvaise signature) et que le champ ctxMode est « TEST ».
Il prend alors comme valeur la chaine qui a été utilisé pour le calcul de signature.

**Remarque** :
Le champ transmissionDate représente la date et l'heure de l'appel de la méthode.
Si la valeur de ce champ est trop éloignée de l'heure réelle, la requête sera refusée (errorCode 6).
Le champ extendedErrorCode prendra alors la valeur du champ erroné

### 3.2. validate

Cette fonction permet d'autoriser la remise en banque d'une transaction à la date de présentation demandée dans le paiement original. Les transactions pouvant faire l'objet d'une validation possèdent l'un des statuts suivants :

- A valider
- A valider et autoriser

Cette fonction prend en entrée les paramètres suivants :

| Nom du champ | Type | Description | Obligatoire |
|---|---|---|---|
| shopId | String | Identifiant de la boutique | ✓ |
| transmissionDate | Date | Date de transaction | ✓ |
| transactionId | String | Identifiant de transaction | ✓ |
| sequenceNb | Int | Numéro de séquence de la transaction | ✓ |
| ctxMode | String | Contexte de sollicitation de la plateforme de paiement ("TEST", "PRODUCTION") | ✓ |
| comment | String | Commentaire « libre » | |
| wsSignature | String | Signature (cf. 4) | ✓ |

Le calcul de la signature se fait en prenant les paramètres dans l'ordre suivant :
shopId, transmissionDate, transactionId, sequenceNb, ctxMode, comment

Cette fonction retourne une réponse du type **standardResponse** (cf. 2.2).

**NB** : le code d'erreur étendu (extendedErrorCode) est renseigné uniquement lorsque l'errorCode est 5 (Mauvaise signature) et que le champ ctxMode est « TEST », il prend alors comme valeur la chaine qui a été utilisée pour le calcul de signature.

**Remarque** :
Le champ transmissionDate représente la date et l'heure de l'appel de la méthode.
Si la valeur de ce champ est trop éloignée de l'heure réelle, la requête sera refusée (errorCode 6).
Le champ extendedErrorCode prendra alors la valeur du champ erroné

### 3.3. force

Cette fonction permet aux commerçants de transmettre le n° d'autorisation d'une transaction suite à appel phonie.
Seules les transactions de statut « à forcer » peuvent bénéficier de cette fonction.

Cette fonction prend en entrée les paramètres suivants :

| Nom du champ | Type | Description | Obligatoire |
|---|---|---|---|
| shopId | String | Identifiant de la boutique | ✓ |
| transmissionDate | Date | Date de transaction | ✓ |
| transactionId | String | Identifiant de transaction | ✓ |
| sequenceNb | Int | Numéro de séquence de la transaction | ✓ |
| ctxMode | String | Contexte de sollicitation de la plateforme de paiement ("TEST", "PRODUCTION") | ✓ |
| autorisationNb | String | Numéro d'autorisation | ✓ |
| autorisationDate | Date | Date d'autorisation | ✓ |
| comment | String | Commentaire « libre » | |
| wsSignature | String | Signature (cf. 4) | ✓ |

Le calcul de la signature se fait en prenant les paramètres dans l'ordre suivant :
shopId, transmissionDate, transactionId, sequenceNb, ctxMode, autorisationNb, autorisationDate, comment.

Cette fonction retourne une réponse du type **standardResponse** (cf. 2.2).

**NB** : le code d'erreur étendu (extendedErrorCode) est renseigné uniquement lorsque l'errorCode est 5 (Mauvaise signature) et que le champ ctxMode est « TEST », il prend alors comme valeur la chaine qui a été utilisée pour le calcul de signature.

**Remarque** :
Le champ transmissionDate représente la date et l'heure de l'appel de la méthode.
Si la valeur de ce champ est trop éloignée de l'heure réelle, la requête sera refusée (errorCode 6).
Le champ extendedErrorCode prendra alors la valeur du champ erroné

### 3.4. modify

Cette fonction permet de modifier le montant d'une transaction (à la baisse) ou d'en modifier la date de remise souhaitée.
Les transactions pouvant faire l'objet d'une modification possèdent l'un des statuts suivant :

- A valider
- A valider et autoriser
- En attente
- En attente d'auto
- En attente de remise

Cette fonction prend en entrée les paramètres suivants :

| Nom du champ | Type | Description | Obligatoire |
|---|---|---|---|
| shopId | String | Identifiant de la boutique | ✓ |
| transmissionDate | Date | Date de transaction | ✓ |
| transactionId | String | Identifiant de transaction | ✓ |
| sequenceNb | Int | Numéro de séquence de la transaction | ✓ |
| ctxMode | String | Contexte de sollicitation de la plateforme de paiement ("TEST", "PRODUCTION") | ✓ |
| amount | Long | Montant demandé en plus petite unité monétaire | ✓ |
| devise | Int | Devise (Code monnaie ISO 4217, Euro : 978) | ✓ |
| remiseDate | Date | Date remise demandée | |
| comment | String | Commentaire « libre » | |
| wsSignature | String | Signature (cf. 4) | ✓ |

Le calcul de la signature se fait en prenant les paramètres dans l'ordre suivant :
shopId, transmissionDate, transactionId, sequenceNb, ctxMode, amount, devise, remiseDate, comment.

Cette fonction retourne une réponse du type **standardResponse** (cf. 2.2).

**NB** : le code d'erreur étendu (extendedErrorCode) est renseigné uniquement lorsque l'errorCode est 5 (Mauvaise signature) et que le champ ctxMode est « TEST », il prend alors comme valeur la chaine qui a été utilisée pour le calcul de signature.

**Remarque** :
Le champ transmissionDate représente la date et l'heure de l'appel de la méthode.
Si la valeur de ce champ est trop éloignée de l'heure réelle, la requête sera refusée (errorCode 6).
Le champ extendedErrorCode prendra alors la valeur du champ erroné.

Si aucune information n'est modifiée, la requête sera rejetée avec un code erreur 26 : aucun changement.

### 3.5. refund

Cette fonction permet de rembourser le porteur.
Les transactions pouvant faire l'objet d'un remboursement possèdent le statut suivant :

- Remisé

Cette fonction prend en entrée les paramètres suivants :

| Nom du champ | Type | Description | Obligatoire |
|---|---|---|---|
| shopId | String | Identifiant de la boutique | ✓ |
| transmissionDate | Date | Date de transaction | ✓ |
| transactionId | String | Identifiant de transaction | ✓ |
| sequenceNb | Int | Numéro de séquence de la transaction | ✓ |
| ctxMode | String | Contexte de sollicitation de la plateforme de paiement ("TEST", "PRODUCTION") | ✓ |
| newTransactionId | String | Identifiant de la transaction créée | ✓ |
| amount | Long | Montant à rembourser en plus petite unité monétaire | ✓ |
| devise | Int | Devise (Code monnaie ISO 4217, Euro : 978) | ✓ |
| presentationDate | Date | Date de remise demandée | |
| validationMode | int | 0 = Automatique, 1 = Manuelle | |
| comment | String | Commentaire « libre » | |
| wsSignature | String | Signature (cf. 4) | ✓ |

Le calcul de la signature se fait en prenant les paramètres dans l'ordre suivant :
shopId, transmissionDate, transactionId, sequenceNb, ctxMode, newTransactionId, amount, devise, presentationDate, validationMode, comment

Cette fonction retourne une réponse du type **transactionInfo** (cf. 2.4).

**NB** : le code d'erreur étendu (extendedErrorCode) est renseigné uniquement lorsque l'errorCode est 5 (Mauvaise signature) et que le champ ctxMode est « TEST ».
Il prend alors comme valeur la chaine qui a été utilisé pour le calcul de signature.

**Remarque** :
Le champ transmissionDate représente la date et l'heure de l'appel de la méthode.
Si la valeur de ce champ est trop éloignée de l'heure réelle, la requête sera refusée (errorCode 6).
Le champ extendedErrorCode prendra alors la valeur du champ erroné

**Remarque importante**
Si la carte est expirée lors de la demande de remboursement, la requête ne sera plus rejetée en errorCode 13.
Une transaction refusée pour motif carte expirée sera créée. La réponse contiendra les valeurs suivantes :
errorCode : 0
transactionStatus : 8

### 3.6. duplicate

Cette fonction permet de créer une nouvelle transaction ayant exactement les mêmes caractéristiques que la transaction qui a servi de base à la duplication.

Les transactions pouvant faire l'objet d'un remboursement possèdent le statut suivant :

- Remisé
- Expiré
- Annulé
- Refusé

Cette fonction prend en entrée les paramètres suivants :

| Nom du champ | Type | Description | Obligatoire |
|---|---|---|---|
| shopId | String | Identifiant de la boutique | ✓ |
| transmissionDate | Date | Date de transaction | ✓ |
| transactionId | String | Identifiant de transaction | ✓ |
| sequenceNb | Int | Numéro de séquence de la transaction | ✓ |
| ctxMode | String | Contexte de sollicitation de la plateforme de paiement ("TEST", "PRODUCTION") | ✓ |
| orderId | String | Référence de la commande | ✓ |
| orderInfo | String | Description libre de la commande | ✓ |
| orderInfo2 | String | Description libre de la commande | ✓ |
| orderInfo3 | String | Description libre de la commande | ✓ |
| amount | Long | Montant en plus petite unité monétaire | ✓ |
| devise | Int | Devise (Code monnaie ISO 4217, Euro : 978) | ✓ |
| newTransactionId | String | Identifiant de la transaction crée | ✓ |
| presentationDate | Date | Date de remise demandée | ✓ |
| validationMode | int | 0 = Automatique, 1 = Manuelle | ✓ |
| comment | String | Commentaire « libre » | ✓ |
| wsSignature | String | Signature (cf. 4) | ✓ |

Le calcul de la signature se fait en prenant les paramètres dans l'ordre suivant :
shopId, transmissionDate, transactionId, sequenceNb, ctxMode, orderId, orderInfo, orderInfo2, orderInfo3, amount, devise, newTransactionId, presentationDate, validationMode, comment

Cette fonction retourne une réponse du type **transactionInfo** (cf. 2.4).

**NB** : le code d'erreur étendu (extendedErrorCode) est renseigné uniquement lorsque l'errorCode est 5 (Mauvaise signature) et que le champ ctxMode est « TEST ».
Il prend alors comme valeur la chaine qui a été utilisé pour le calcul de signature.

**Remarque** :
Le champ transmissionDate représente la date et l'heure de l'appel de la méthode.
Si la valeur de ce champ est trop éloignée de l'heure réelle, la requête sera refusée (errorCode 6).
Le champ extendedErrorCode prendra alors la valeur du champ erroné

## 3.7. create

Cette fonction permet :

- les paiements « manuels » (de type VAD – ERT 20), issus des différents canaux ;
- les paiements « automatiques » pour lesquels l'acquisition des données cartes est réalisée par le commerçant lui-même.

Cette fonction prend en entrée les paramètres suivants :

| Nom du champ | Type | Description | Obligatoire |
|---|---|---|---|
| createInfo | CreatePaiementInfo | cf. 2.5 | ✓ |
| wsSignature | String | Signature (cf. 3.9) | ✓ |

Le calcul de la signature se fait en prenant les paramètres dans l'ordre suivant :

shopId, transmissionDate, transactionId, paymentMethod, orderId, orderInfo, orderInfo2, orderInfo3, amount, devise, presentationDate, validationMode, cardNumber, cardNetwork, cardExpirationDate, cvv, contractNumber,
threeDsSig, subPaymentType, subReference, subPaymentNumber,
customerId, customerTitle, customerName, customerPhone, customerMail, customerAddress, customerZipCode, customerCity, customerCountry, customerLanguage, customerIP, customerSendEmail, ctxMode, comment

**Note** : si le paramètre threeDsSig est non renseigné, la valeur prise en compte dans le calcul de signature est vide (comme tous les champs non renseignés), par contre si il est renseigné il est calculé de la sorte : brand+enrolled+authStatus+eci+xid+cavv+cavvAlgorithm soit par exemple :
VISA+Y+Y++XidXidXidXidXidXidXidXidX+CavvCavvCavvCavvCavvCavvCavv+2

Cette fonction retourne une réponse du type **transactionInfo** (cf. 2.4).

**NB** : le code d'erreur étendu (extendedErrorCode) est renseigné uniquement lorsque l'errorCode est 5 (Mauvaise signature) et que le champ ctxMode est « TEST ».
Il prend alors comme valeur la chaine qui a été utilisé pour le calcul de signature.

**Remarque** :
Le champ transmissionDate représente la date et l'heure de l'appel de la méthode.
Si la valeur de ce champ est trop éloignée de l'heure réelle, la requête sera refusée (errorCode 6).
Le champ extendedErrorCode prendra alors la valeur du champ erroné

Ci-dessous un exemple de fichier xml généré lors de l'appel de la méthode **create** le 24/04/2012 à 09h33 (heure de Paris):

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV=http://schemas.xmlsoap.org/soap/envelope/
 xmlns:ns1="http://v3.ws.vads.lyra.com/">
<SOAP-ENV:Body>
   <ns1:create>
      <createInfo>
        <shopId>48337286</shopId>
        <transmissionDate>2012-04-24T07:33:18+00:00</transmissionDate>
        <transactionId>073318</transactionId>
        <paymentMethod>EC</paymentMethod>
        <orderId>orderId</orderId>
        <amount>3000</amount>
        <devise>978</devise>
        <cardNumber>4970100000000003</cardNumber>
        <cardNetwork>CB</cardNetwork>
        <cardExpirationDate>2012-12-30T23:00:00+00:00</cardExpirationDate>
        <cvv>123</cvv>
        <ctxMode>TEST</ctxMode>
      </createInfo>
     <wsSignature>79b39d396539d78dfa799c6d8794aa3a8851f0cb</wsSignature>
   </ns1:create>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Et le calcul de signature correspondant :

48337286+20120424+073318+EC+orderId++++3000+978+++4970100000000003+CB+20121230+123+++++
++++++++++++++TEST++*certificat*

### 3.8. getInfo

Cette fonction permet d'interroger une transaction pour en connaître ses différents attributs.

Cette fonction prend en entrée les paramètres suivants :

| Nom du champ | Type | Description | Obligatoire |
|---|---|---|---|
| shopId | String | Identifiant de la boutique | ✓ |
| transmissionDate | Date | Date de transaction | ✓ |
| transactionId | String | Identifiant de transaction | ✓ |
| sequenceNb | Int | Numéro de séquence de la transaction | ✓ |
| ctxMode | String | Contexte de sollicitation de la plateforme de paiement ("TEST", "PRODUCTION") | ✓ |
| wsSignature | String | Signature (cf. 4) | ✓ |

Le calcul de la signature se fait en prenant les paramètres dans l'ordre suivant :
shopId, transmissionDate, transactionId, sequenceNb, ctxMode

Cette fonction retourne une réponse du type **transactionInfo** (cf. 2.4).

**NB** : le code d'erreur étendu (extendedErrorCode) est renseigné uniquement lorsque l'errorCode est 5 (Mauvaise signature) et que le champ ctxMode est « TEST ».
Il prend alors comme valeur la chaine qui a été utilisé pour le calcul de signature.

**Remarque** :
Le champ transmissionDate représente la date et l'heure de l'appel de la méthode.
Si la valeur de ce champ est trop éloignée de l'heure réelle, la requête sera refusée (errorCode 6).
Le champ extendedErrorCode prendra alors la valeur du champ erroné

### 3.9. modifyAndValidate

Cette fonction permet de modifier le montant d'une transaction (à la baisse) ou d'en modifier la date de remise souhaitée et de valider la transaction si besoin.
Les transactions pouvant faire l'objet d'une modification possèdent l'un des statuts suivant :

- A valider
- A valider et autoriser
- En attente
- En attente d'auto
- En attente de remise

Cette fonction prend en entrée les paramètres suivants :

| Nom du champ | Type | Description | Obligatoire |
|---|---|---|---|
| shopId | String | Identifiant de la boutique | ✓ |
| transmissionDate | Date | Date de transaction | ✓ |
| transactionId | String | Identifiant de transaction | ✓ |
| sequenceNb | Int | Numéro de séquence de la transaction | ✓ |
| ctxMode | String | Contexte de sollicitation de la plateforme de paiement ("TEST", "PRODUCTION") | ✓ |
| amount | Long | Montant demandé en plus petite unité monétaire | ✓ |
| devise | Int | Devise (Code monnaie ISO 4217, Euro : 978) | ✓ |
| remiseDate | Date | Date remise demandée | ✓ |
| comment | String | Commentaire « libre » | |
| wsSignature | String | Signature (cf. 4) | ✓ |

Le calcul de la signature se fait en prenant les paramètres dans l'ordre suivant :
shopId, transmissionDate, transactionId, sequenceNb, ctxMode, amount, devise, remiseDate, comment

Cette fonction retourne une réponse du type **standardResponse** (cf. 2.2).

**NB** : le code d'erreur étendu (extendedErrorCode) est renseigné uniquement lorsque l'errorCode est 5 (Mauvaise signature) et que le champ ctxMode est « TEST ».
Il prend alors comme valeur la chaine qui a été utilisé pour le calcul de signature.

**Remarque** :
Le champ transmissionDate représente la date et l'heure de l'appel de la méthode.
Si la valeur de ce champ est trop éloignée de l'heure réelle, la requête sera refusée (errorCode 6).
Le champ extendedErrorCode prendra alors la valeur du champ erroné

Si aucune information n'est modifiée, la requête sera rejetée avec un code erreur 26 : aucun changement.

# 4. Signature

Un certificat est nécessaire pour dialoguer avec la plateforme de paiement. Il est mis à disposition de toutes les personnes habilitées à la consultation des certificats dans votre outil de gestion de caisse à l'emplacement suivant : Paramètres / Boutique / Certificat. Il existe deux certificats différents : un pour la plateforme de test et un pour la plateforme de production.

La signature sera générée comme suit :

- Création d'une chaîne de caractère représentant la concaténation des paramètres, séparés par le caractère "+".
- Ajout à cette chaîne d'un "certificat " numérique (de test ou de production selon le contexte).
- Hachage de la chaîne résultante avec l'algorithme SHA1.

La plateforme de paiement effectuera obligatoirement la vérification de la signature. Il est de la responsabilité du commerçant de vérifier à son tour la signature transmise en retour.

L'ordre des champs doit être respecté.

**Les champs de type 'date' doivent être formatés de la manière suivante : YYYYMMDD en UTC soit pour le 1er Février 2009 : 20090201. ATTENTION aux heures, par exemple le 23/02/2010 00:30:00 heure française (GMT+1 ou GMT+2) donne 20100222 !**

Les champs de type numérique ne doivent pas avoir de 0 à gauche du digit le plus significatif.

Les champs de type bool prennent les valeurs suivantes :

- 1 pour vrai (true)
- 0 pour faux (false)

Les champs de type String non renseignés seront vides.

Exemple : Pour un appel Cancel si les paramètres de la requête sont les suivants :

- shopId = 12345678
- transmissionDate = 7 Mars 2010
- transactionId = 654321
- sequenceNb = 1
- ctxMode = TEST
- comment = *non renseigné*

Si la valeur du certificat de test est 1122334455667788, alors la chaîne à utiliser pour le hachage à l'aide de l'algorithme SHA1 est la suivante :

12345678+20100307+654321+1+TEST++1122334455667788

Ce qui donne après hachage (signature) :

88ff8fc1897ac4edf34c5e2327be5adde76e17d6

En mode TEST, en cas de mauvais calcul de signature, le code erreur de la fonction renvoie 5, la chaine de caractère utilisée pour la signature côté serveur est alors renvoyée dans le champ extendedErrorCode.

# 5. Maintien de la session HTTP entre 2 requêtes

**Important :**

L'architecture de la plateforme de paiement reposant sur un ensemble de serveurs avec répartition de charge, il est nécessaire que chaque requête concernant un même paiement dans un laps de temps très court, soient réalisées avec la même session HTTP afin d'assurer la continuité du processus.

Pour cela, à chaque requête, une session est créée coté serveur.
L'ID de la session est renvoyé dans l'entête HTTP de la réponse. Il devra être retourné dans les requêtes suivantes afin que la requête soit traitée par le même serveur, évitant ainsi à votre requête d'être rejetée car la transaction ne serait pas encore disponible sur les autres serveurs.

## 5.1. Exemple d'application

Vous souhaitez créer un paiement à remettre dans 30 jours en mode de validation manuelle.
Une fois le paiement accepté vous décidez de changer la date de remise pour le lendemain et de valider la transaction.

- Vous appelez le web service de création de paiement (create).
- La plateforme vérifie la présence d'un ID de session dans l'en-tête HTTP de votre requête.
  Comme rien n'a été précisé, une nouvelle session et un nouvel ID son créés.
  Systempay procède ensuite au traitement de votre requête et envoie sa réponse en indiquant dans l'en-tête HTTP l'identifiant de session attribué ainsi que le nom du serveur ayant traité la requête:

*Exemple d'entête de requête :*

```
POST /vads-ws/v3 HTTP/1.1
Host: paiement.systempay.fr
Connection: Keep-Alive
User-Agent: PHP-SOAP/5.2.10
Content-Type: text/xml; charset=utf-8
SOAPAction: ""
Content-Length: 482
```

*Exemple d'entête de réponse :*

```
HTTP/1.1 200 OK
Date: Thu, 25 Oct 2012 09:48:06 GMT
Server: Apache
Set-Cookie: JSESSIONID=00A19F16AD158A3C4862EBB84896E9FE.sirisvad2; Path=/vads-ws;
Secure; HttpOnly
Access-Control-Allow-Origin: *
Vary: Accept-Encoding
Connection: close
Transfer-Encoding: chunked
Content-Type: text/xml;charset=UTF-8
```

- Dans les en-têtes HTTP de la réponse, vous récupérez le cookie **JSESSIONID** :
- Vous initialisez le cookie JSESSIONID de votre en-tête HTTP.
- Vous appelez le web service de modification et de validation du paiement (modifyAndValidate).

*Exemple d'entête de requête :*

```
POST /vads-ws/v3 HTTP/1.1
Host: paiement.systempay.fr
Connection: Keep-Alive
User-Agent: PHP-SOAP/5.2.10
Content-Type: text/xml; charset=utf-8
SOAPAction: ""
Content-Length: 482
Cookie: JSESSIONID=00A19F16AD158A3C4862EBB84896E9FE.sirisvad2;
```

- La plateforme vérifie la présence d'un ID de session dans l'en-tête HTTP de votre requête.
  La requête est ensuite envoyée au serveur ayant généré la session.
  Si la session existe, elle est réutilisée, sinon une nouvelle session et un nouvel identifiant seront créés.
  Systempay procède au traitement de la requête et envoie sa réponse.

**Remarque :**
L'ID de session sera renvoyé dans l'en-tête HTTP de la réponse uniquement dans le cas où une nouvelle session a été créée.
Il est donc conseillé de tester systématiquement la présence du cookie dans l'en-tête HTTP de la réponse et d'utiliser l'ID de session présent avec d'enchainer un autre appel (getInfo par exemple).

*Exemple d'en-tête HTTP de réponse utilisant la même session:*

```
HTTP/1.1 200 OK
Date: Thu, 25 Oct 2012 09:48:07 GMT
Server: Apache
Secure; HttpOnly
Access-Control-Allow-Origin: *
Vary: Accept-Encoding
Connection: close
Transfer-Encoding: chunked
Content-Type: text/xml;charset=UTF-8
```

*(Pas d'information sur l'identifiant de session)*

*Exemple d'en-tête HTTP de réponse avec une nouvelle session:*

```
HTTP/1.1 200 OK
Date: Thu, 25 Oct 2012 09:48:06 GMT
Server: Apache
Set-Cookie: JSESSIONID=F6D2CCC8B075077843724ADDDE887EE2.sirisvad2; Path=/vads-ws;
Secure; HttpOnly
Access-Control-Allow-Origin: *
Vary: Accept-Encoding
Connection: close
Transfer-Encoding: chunked
Content-Type: text/xml;charset=UTF-8
```

*(Présence du Set-Cookie précisant l'identifiant de la nouvelle session)*

## 5.2. Exemple d'implémentation en PHP

Pour récupérer l'en-tête HTTP de la réponse vous devez utiliser la fonction __getLastResponseHeaders()

```
/* La méthode ci-dessous permet de récupérer l'entête HTTP de la réponse */
/* $client étant une instance du client SOAP utilisé pour appeler les WS */
$header = $client->__getLastResponseHeaders();
```

Pour récupérer ensuite l'identifiant de session vous pouvez utiliser le code suivant :

```
/* Dans la chaîne de caractère obtenue, nous recherchons la présence de l'ID de la session
HTTP, stockée dans l'élément "JSESSIONID" : */

if(preg_match("#JSESSIONID=([A-Za-z0-9\.]+)#",$header, $matches)){
      $JSESSIONID = $matches[1];;
}
```

Pour initialiser le cookie vous devez utiliser la fonction __setCookie()

```
$cookie= $JSESSIONID;
$client->__setCookie('JSESSIONID', $cookie);
```

# 6. Migration de la v2 à la v3

## 6.1. Changement de signature pour la méthode create

Suite à l'ajout de paramètres optionnels, voici les champs (en rouge) à ajouter dans le calcul de signature :

shopId, transmissionDate, transactionId, paymentMethod, orderId, orderInfo, orderInfo2, orderInfo3, amount, devise, presentationDate, validationMode, cardNumber, cardNetwork, cardExpirationDate, cvv, contractNumber,
threeDsSig, subPaymentType, subReference, subPaymentNumber,
customerId, customerTitle, customerName, customerPhone, customerMail, customerAddress, customerZipCode, customerCity, customerCountry, customerLanguage, customerIP, customerSendEmail, ctxMode, comment

Note : Dans le cadre de la migration, le paramètre threeDsSig est sans doute non renseigné, la valeur prise en compte dans le calcul de signature est vide (comme tous les champs non renseignés), par contre si il est renseigné il sera calculé de la sorte :
brand+enrolled+authStatus+eci+xid+cavv+cavvAlgorithm soit par exemple :
VISA+Y+Y++XidXidXidXidXidXidXidXidX+CavvCavvCavvCavvCavvCavvCavv+2

## 6.2. Changement de signature pour le retour de transactionInfo

Suite à l'ajout d'un nouveau paramètre dans le type TransactionInfo, voici le champ (en rouge) à ajouter dans le calcul de signature :

errorCode, extendedErrorCode, transactionStatus, shopId, paymentMethod, contractNumber, orderId, orderInfo, orderInfo2, orderInfo3, transmissionDate, transactionId, sequenceNb, amount, initialAmount, devise, cvAmount, cvDevise, presentationDate, type, multiplePaiement, ctxMode, cardNumber, cardNetwork, cardType, cardCountry, cardExpirationDate, customerId, customerTitle, customerName, customerPhone, customerMail, customerAddress, customerZipCode, customerCity, customerCountry, customerLanguage, customerIP, transactionCondition, vadsEnrolled, vadsStatus, vadsECI, vadsXID, vadsCAVVAlgorithm, vadsCAVV, vadsSignatureValid, directoryServer, authMode, markAmount, markDevise, markDate, markNb, markResult, markCVV2_CVC2, authAmount, authDevise, authDate, authNb, authResult, authCVV2_CVC2, warrantlyResult, captureDate, captureNumber, rapprochementStatut, refoundAmount, refundDevise, litige, timestamp

## 6.3. Note aux utilisateurs PHP

Les paramètres de retour ne sont plus ambigus et ne sont plus retournés en double dans un tableau, il faut modifier votre code d'interprétation du retour en accédant directement aux champs et non plus en passant par un tableau intermédiaire.

# 7. Exemple d'implémentation

Cet exemple utilise les webservices client de Jboss.

## 7.1. Générer les stubs depuis le wsdl

Afin d'utiliser les Webservices, il faut générer le code qui va communiquer avec les webservices :
`wsconsume.bat -k -p com.lyra.vads.ws.stubs https://paiement.systempay.fr/vads-ws/v3?wsdl`

## 7.2. Exemple de code pour générer la signature

Voici un exemple de classe pour générer la signature.

```
import java.security.MessageDigest;
import java.text.SimpleDateFormat;
import java.util.Date;

public class Signature {

	static public final String SEPARATOR = "+";
	static final String key = "1122334455667788";

	public static String createSignature(Object... params) {

		StringBuilder builder = new StringBuilder();

		for (int i = 0; i < params.length; i++) {
			if (i != 0) {
				builder.append(SEPARATOR);
			}


			if (params[i] != null) {
				if (params[i] instanceof Date || params[i] instanceof
XMLGregorianCalendar) {
					Date date = null;
					if (params[i] instanceof XMLGregorianCalendar) {
						XMLGregorianCalendar xmlDate = (XMLGregorianCalendar)
params[i];
						date = xmlDate.toGregorianCalendar().getTime();
					} else {
						date = (Date) params[i];
					}
					SimpleDateFormat myDateFormat = new SimpleDateFormat("yyyyMMdd");
					myDateFormat.setTimeZone(TimeZone.getTimeZone("UTC"));
					builder.append(myDateFormat.format(date));
				} else if (params[i] instanceof Boolean) {
					if ((Boolean) params[i]) {
						builder.append(1);
					} else {
						builder.append(0);
					}
				} else {
					builder.append(params[i]);
				}
			}
		}

		String toSign = builder.toString() + SEPARATOR + key;

		return encode(toSign);
	}

	public static String encode(String src) {
		try {
			MessageDigest md;
```

```
                        md = MessageDigest.getInstance("SHA-1");

                        byte bytes[] = src.getBytes("UTF-8");

                        md.update(bytes, 0, bytes.length);
                        byte[] sha1hash = md.digest();

                        return convertToHex(sha1hash);
                } catch (Exception e) {
                        throw new RuntimeException(e);
                }
        }

        private static String convertToHex(byte[] sha1hash) {
                StringBuilder builder = new StringBuilder();
                for (int i = 0; i < sha1hash.length; i++) {
                        byte c = sha1hash[i];

                        addHex(builder, (c >> 4) & 0xf);
                        addHex(builder, c & 0xf);

                }
                return builder.toString();
        }

        private static void addHex(StringBuilder builder, int c) {
                if (c < 10)
                        builder.append((char) (c + '0'));
                else
                        builder.append((char) (c + 'a' - 10));
        }

}
```

## 7.3. Exemple de code pour annuler un paiement

Remplacer XXXXXXXX par votre numéro commerçant et 123456 par votre numéro de transaction unique par jour.

```
        public static void main(String[] args) throws Exception {
                URL wsdlURL = new URL(
                                "https://paiement.systempay.fr/vads-ws/v3?wsdl");

                QName qname = new QName("http://v3.ws.vads.lyra.com/", "StandardWS");
                Service service = Service.create(wsdlURL, qname);
                StandardWs port = service.getPort(StandardWs.class);

                GregorianCalendar gCalendar = new GregorianCalendar();
                gCalendar.setTime(new Date());
                XMLGregorianCalendar xmlCalendar = DatatypeFactory.newInstance()
                                .newXMLGregorianCalendar(gCalendar);

                String signature = createSignature("XXXXXXXX", xmlCalendar, "123456",
                                1, "TEST", "Cancel comment");

                StandardResponse response = port.cancel("XXXXXXXX", xmlCalendar,
                                "123456", 1, "TEST", "Cancel comment", signature);

                System.out.println("WS Cancel Result : " + response.getErrorCode()
                                + " / " + response.getExtendedErrorCode() + " / "
                                + response.getTransactionStatus());
        }
```

## 7.4. Exemple de code pour créer un paiement

Remplacer XXXXXXXX par votre numéro commerçant et 123456 par votre numéro de transaction unique par jour.

```
public static void main(String[] args) throws Exception {
                URL wsdlURL = new URL(
                                "https://paiement.systempay.fr/vads-ws/v3?wsdl");

                QName qname = new QName("http://v3.ws.vads.lyra.com/", "StandardWS");
                Service service = Service.create(wsdlURL, qname);
                StandardWs port = service.getPort(StandardWs.class);

                GregorianCalendar gCalendar = new GregorianCalendar();
                gCalendar.setTime(new Date());
                XMLGregorianCalendar xmlCalendar = DatatypeFactory.newInstance()
                                .newXMLGregorianCalendar(gCalendar);
                XMLGregorianCalendar expDate = DatatypeFactory.newInstance()
                                .newXMLGregorianCalendar(2010, 12, 31, 0, 0, 0, 0, 0);

                CreatePaiementInfo myPayment = new CreatePaiementInfo();

                // Numero commercant
                myPayment.setShopId("XXXXXXXX");
                // Date de transmission
                myPayment.setTransmissionDate(xmlCalendar);
                // Numero de transaction
                myPayment.setTransactionId("123456");
                // VPC
                myPayment.setPaymentMethod("EC");

                // Id de la commande ...
                myPayment.setOrderId("maCommande");
                // Info sur la commande => Facultatif !
                myPayment.setOrderInfo("info commande");
                // Montant en centimes
                myPayment.setAmount(3000);
                // Devise => 978 Euros
                myPayment.setDevise(978);
                // Date de la remise
                myPayment.setPresentationDate(xmlCalendar);
                // Validation automatique
                myPayment.setValidationMode(0);

                // Numero de la carte bleue
                myPayment.setCardNumber("4970100000000000");
                // Reseau de la carte
                myPayment.setCardNetwork("CB");
                // Date expiratiration carte
                myPayment.setCardExpirationDate(expDate);
                // CVV
                myPayment.setCvv("000");

                // Mode de connexion ...
                myPayment.setCtxMode("TEST");
                // Un commentaire eventuel
                myPayment.setComment("Creation Par WebService");

                String signature = Signature.createSignature(myPayment.getShopId(),
                                myPayment.getTransmissionDate(),myPayment.getTransactionId(),
                                myPayment.getPaymentMethod(), myPayment.getOrderId(),
                                myPayment.getOrderInfo(), myPayment.getOrderInfo2(),
                                myPayment.getOrderInfo3(), myPayment.getAmount(),
                                myPayment.getDevise(),myPayment.getPresentationDate(),
                                myPayment.getValidationMode(), myPayment.getCardNumber(),
                                myPayment.getCardNetwork(), myPayment.getCardExpirationDate(),
                                myPayment.getCvv(), myPayment.getContractNumber(), null,
                                myPayment.getSubPaymentType(), myPayment.getSubReference(),
                                myPayment.getSubPaymentNumber(), myPayment.getCustomerId(),
                                myPayment.getCustomerTitle(), myPayment.getCustomerName(),
                                myPayment.getCustomerPhone(), myPayment.getCustomerMail(),
```

```
                    myPayment.getCustomerAddress(), myPayment.getCustomerZipCode(),
                    myPayment.getCustomerCity(), myPayment.getCustomerCountry(),
                    myPayment.getCustomerLanguage(), myPayment.getCustomerIP(),
                    myPayment.isCustomerSendEmail(), myPayment.getCtxMode(),
                    myPayment.getComment());

            TransactionInfo respTransactionInfo = port.create(myPayment,
                            signature);

            System.out.println("WS Result : " + respTransactionInfo.getErrorCode()
                            + " / " + respTransactionInfo.getExtendedErrorCode() + " / "
                            + respTransactionInfo.getTransactionStatus());

    }
```