Descriptif Technique du Kit de paiement V4.06 W-HA

Offre	Internet+ Mobile
Version	Version 0.1-8 (Juillet 2023)

Résumé :	Ce document est à destination des éditeurs ou distributeurs souhaitant mettre en place la solution Internet+ Mobile. Il comprend la description du KIT et de ses différentes applications.

Historique du document

Version	Date	Auteur	Modifications
0.1-1	08/03/2019		Modifications des Urls d'appel à W-ha pour le passage en mode HTTPS.
0.1-2	20/06/2020		- Sécurisation des URLS générées (cf. parag 7) - Mise à jour des exemples de requêtes générées par le kit (ajout du token), achat acte / abonnement, actions Editeurs, API - Nouveaux Parcours OTP accéléré
0.1-3	21/02/2022		- Nouvelle API subscriptionTransactions (récupération des transactions d'un abonnement)
0.1-4	19/09/2022		-Plus grande précision sur les erreurs possibles lors de la consultation des transactions d'abonnements
0.1-5	12/12/2022	Kevin LAM	Correction de la requête de résiliation d'abonnement
0.1-6	15/12/2022	Kevin LAM	Correction des exemples des requêtes
0.1-7	30/01/2023	Kevin LAM	Correction de la requête fullRefund Correction des informations de la requête fullRefund Ajout de la partie Limite d'achats par transaction

0.1-8	05/07/2023	Kevin LAM	Mise à jour de la partie 5.2 Requêtes serveur à serveur entre l'éditeur et w-HA (servlet admin)

Validation

Nom	Département	Date

Liste de diffusion

Service	Nom
DRC	
W-HA	
Orange	

TABLE DES MATIERES

1 Objet du document

Ce document décrit les fonctionnements généraux du Kit de paiement associé à la solution Internet + Mobile.

Ce Kit de paiement est fourni par la société W-HA, filiale du groupe Orange.

Il suppose l'utilisation du Merchant Self Care Application (MSCA) qui est un extranet permettant d'effectuer différentes opérations comme le paramétrage des Produits. Le Merchant Self Care Application (MSCA) fait l'objet d'un Descriptif technique à part.

2 Principe de fonctionnement du système w-HA

Ce paragraphe a pour objet d'identifier les "acteurs" impliqués lors d'une transaction avec w-HA et de décrire simplement les flux qui ont lieu entre ces différents acteurs.

2.1 Vocabulaire

w-HA :

Le terme "w-HA" peut désigner, selon le contexte :

- la société w-HA,
- la plate-forme technique w-HA,

Marchand :

Le "Marchand" est un vendeur de biens téléchargeables ou services accessibles sur l'Internet Mobile, et qui utilise la solution w-HA.

Boutique :

Une boutique est définie par son MerchantID. Un marchand peut avoir plusieurs boutiques.

Opérateur Client :

L' "Opérateur Client" est une entité qui a une relation commerciale et financière avec des clients, et qui a établit un partenariat avec w-HA.
Dans le cas présent, il s'agit de l'opérateur de téléphonie mobile Orange.

Utilisateur Mobile :

L'Utilisateur Mobile désigne le client final, qui achète, sur l'Internet Mobile, un bien téléchargeable ou un service, en utilisant une des solutions w-HA

2.2 Cinématiques

La solution développée par w-HA repose sur un nœud Valista 3.5 mis à jour.

Ce document intègre les cinématiques des phases d'achat de produit à l'acte, en abonnement et de consommation d'abonnement.

2.2.1 Achat à l'acte d'un produit

Le kit propose la liste des produits « achats à l'acte ». Le kit fonctionne selon une redirection avec la servlet pos-bundle et l'action : purchaseListOffer.

2.2.2 Achat à l'abonnement d'un produit

Le kit propose la liste des produits « abonnements ». Le kit fonctionne selon une redirection avec la servlet pos-bundle et l'action : purchaseListOffer.

2.2.3 Consommation d'un abonnement

L'utilisateur possède déjà un abonnement, le kit permet à l'utilisateur de pouvoir consommer automatiquement son abonnement.

2.3 Parcours clients

Le Kit V4 MPME permet de proposer aux clients Orange un parcours adapté à chaque situation : Mobile, Smartphone, PC ou tablettes, en WIFI ou en 3G.
La plateforme w-HA s'occupe de la redirection en fonction du contexte du client. Elle utilise le dépôt publique WURFL (http://wurfl.sourceforge.net) pour diriger le client sur le parcours le plus adapté.

2.3.1 Parcours classique

Le client utilise son mobile sur le réseau DATA d'Orange. Il est automatiquement reconnu et arrive directement sur le panneau de paiement mobile.

2.3.2 Parcours mobile en WIFI (OTP deux étapes)

Le client utilise son mobile connecté à Internet en WIFI. Il doit saisir son numéro de mobile puis un code de sécurité (One Time Password) envoyé par SMS à ce numéro. Il est ensuite redirigé vers

un panneau de paiement adapté aux mobiles. La mise en production de ce paramétrage mobile/Wifi/OTP marque la fin de l'authentification SSO EUI sur cinématique Internet+ mobile.

2.3.3 Parcours mobile en WIFI (OTP accéléré 1 étape)

Le client utilise son mobile connecté à Internet en WIFI. Il s'agit d'un parcours sur un seul écran de paiement. L'utilisateur achète en 1 étape, avec saisie de code de sécurité (One Time Password) envoyé par SMS à son numéro. La mise en production de ce paramétrage mobile/Wifi/OTP marque la fin de l'authentification SSO EUI sur cinématique Internet+ mobile.

2.3.4 Parcours Full Web (OTP deux étapes)

Le client utilise un PC ou une tablette connecté en WIFI ou en 3G. Dans une page web classique, il doit saisir son numéro de mobile et un code de sécurité (One Time Password) envoyé par SMS. Il est ensuite redirigé vers un panneau de paiement adapté aux PC et tablettes.

2.3.5 Parcours Full Web (OTP accéléré 1 étape)

Le client utilise un PC ou une tablette connecté en WIFI ou en 3G. Il s'agit d'un parcours sur un seul écran de paiement adapté aux PC et tablettes. L'utilisateur achète en 1 étape, avec saisie de code de sécurité (One Time Password) envoyé par SMS à son numéro.

Le panneau Full Web peut-être utilisé de différentes manières :

- en pleine page (appel classique du kit avec le paramètre format=xhtml)
- en version « small », afin de permettre une meilleure intégration
- en mode Light Box. Un script Light Box est disponible en complément du kit et son utilisation est décrite en annexe 9.6 Intégration Light Box pour MPME Full Web

3 Description technique du système w-HA (côté marchand)

3.1 Schéma descriptif

La mise en œuvre de la solution w-HA nécessite l'installation, sur le serveur Web du marchand, de l'application w-HA Kit V4.

Le schéma suivant illustre les relations entre les différents composants logiciels et matériels nécessaires au fonctionnement de l'application w-HA.

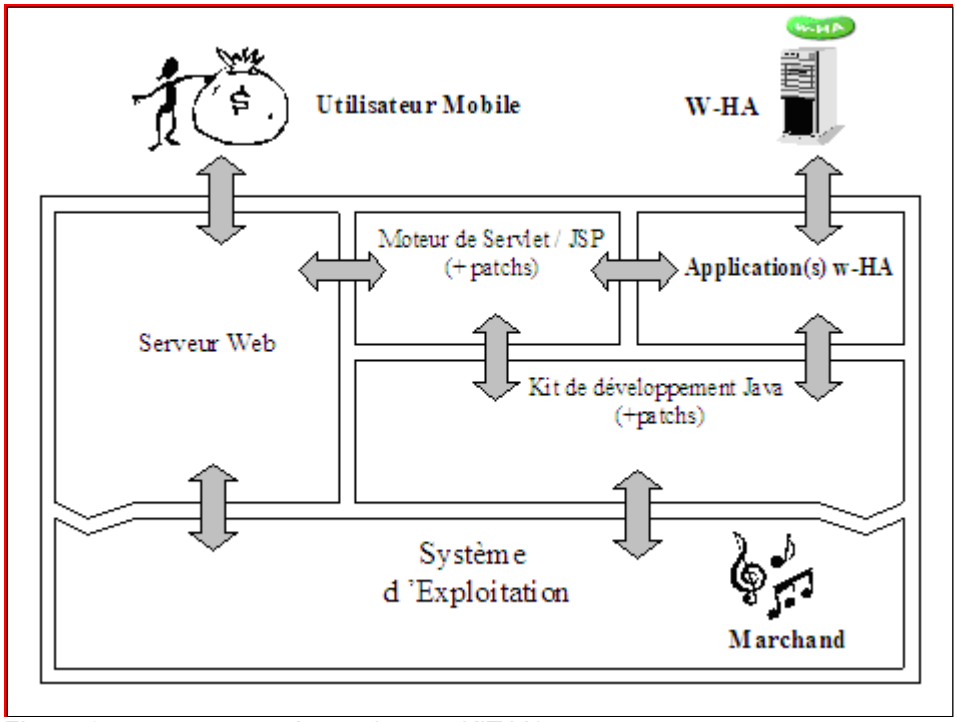


Figure 6 : composants nécessaires au KIT V4

3.2 Pré-requis techniques pour le fonctionnement du système w-HA

Le fonctionnement du système de paiement w-HA nécessite l'installation d'une application Java sur la plate-forme d'hébergement du marchand.

3.2.1 Plate-forme d'hébergement

L'environnement technique (système d'exploitation et serveur web) de la plate-forme d'hébergement du site Web du marchand doit être compatible avec l'application « KIT V4».

Celle-ci repose sur l'utilisation de servlets Java, ce qui permet une compatibilité avec la majorité des plates-formes du marché.

3.2.2 Système d'exploitation

w-HA garantit le bon fonctionnement du logiciel marchand et en assure le support technique, pour les environnements décrits ci-dessous.

Si le système d'exploitation est :

Sun Solaris version 2.6, version 2.7, ou versions supérieures, w-HA assure le support technique pour les serveurs Web :

- Apache Web Server version 1.4.2_03 ou versions supérieures (avec support des modules DSO).
  Si Apache n'est pas installé, installer une version récente.

Linux Kernel version 2.2.x (glibc/libc6 doit être installé), w-HA assure le support technique pour les serveurs Web :

- Apache Web Server version 1.4.2_03 ou versions supérieures (avec support des modules DSO).
  Si Apache n'est pas installé, installer une version récente.

Windows 2000, w-HA assure le support technique pour les serveurs Web :

- Apache Web Server version 1.4.2_03 ou versions supérieures (avec support des modules DSO).
  Si Apache n'est pas installé, installer une version récente.

Windows Server 2003 ou ultérieur, w-HA assure le support technique pour les serveurs Web :

- Apache Web Server version 1.4.2_03 ou versions supérieures (avec support des modules DSO).
  Si Apache n'est pas installé, installer une version récente.
- Internet Information Server 6

Pour les autres environnements (autres couples Système d'Exploitation / Serveur Web), le fonctionnement des applications « KIT V4 » est tout à fait envisageable, mais dans ce cas, l'installation du J2SDK (Java) et la réalisation du « bridge » entre le serveur Web et le moteur de Servlet seront effectués par le marchand, sous son entière responsabilité.

3.2.2.1 Machine Virtuelle Java

Le JDK 5 ou version supérieure doit être installé, avant ou pendant l'intégration.

w-HA préconise l'installation de la version « JDK 8».

Remarque :

Si la plate-forme est de type Linux/UNIX, il faut que les patchs requis pour chaque composant soient installés.

3.2.2.2 Moteur de Servlet

Un moteur de Servlets/JSP respectant les spécifications SUN suivantes doit être installé, avant ou pendant l'intégration :

- JavaServlet 2.4
- JavaServlets Pages (JSP) 2.0

w-HA recommande le moteur de Servlets/JSP « Tomcat 9.0.X»**.**

Si le moteur de Servlets/JSP est Tomcat, il faut que le serveur Web soit l'un des suivants :

- Apache Web Server version 1.4.2_03 ou versions supérieures (avec support des modules DSO). Si Apache n'est pas installé, installer une version récente.
- Internet Information Server 6 – IIS6 ou ultérieur

Pour les autres Serveurs Web, ou si le moteur de Servlet du marchand n'est pas Tomcat, le fonctionnement du logiciel w-HA est tout à fait envisageable, à condition que le moteur de Servlets respecte bien les spécifications SUN énoncées ci-dessus. Dans ce cas, l'intégration de l'application au sein du moteur de Servlet est à la charge et sous l'entière responsabilité du marchand. w-HA assure le support technique sur le fonctionnement et le paramétrage de l'application w-HA.

3.2.3 Considérations Réseau

Pour que les applications « KIT V4 » puissent fonctionner, certains paramétrages des équipements réseaux sont à considérer (ouverture de ports, firewalls, proxy).

Il faut que la plate-forme d'hébergement du logiciel marchand w-HA :
- soit accessible depuis l'extérieur par le port du Serveur Web existant (par défaut : port 80)
- puisse initier une communication SSL (sur le port 443) avec la plate-forme w-HA, dont l'URL est :

https://orange.w-ha.com (pour l'Opérateur Orange France)

Les différents équipements réseaux (Firewalls, Proxies, ….) doivent donc être correctement configurés, pour permettre les accès précisés ci-dessus.

3.2.4 Autres pré-requis pour l'intégration

3.2.4.1 Redémarrage ("reboot") du Serveur Web

L'installation du logiciel marchand nécessite une modification de la configuration du (des) serveur(s) (serveur Http et Moteur de Servlets).
Ces modifications nécessitent un redémarrage du (des) serveur(s), voire de la machine dans certains cas.

3.2.4.2 Présence de l'administrateur système

Lors de l'installation du logiciel marchand et le redémarrage du serveur Web, la présence de l'administrateur système (droits "root") de la plate-forme d'hébergement est nécessaire (1/2 à 1 journée environ).

3.3 Description technique et paramétrage de la servlet w-HA

3.3.1 Servlets du KIT V4

Le KIT V4 est composé de deux servlets distinctes :

- pos-bundle
- admin

La servlet pos-bundle permet de gérer les actions déclenchées par le client final telles que les achats de produit ou la souscription d'un abonnement.

La servlet admin sera utilisé par l'éditeur pour gérer les transactions et les abonnements de sa boutique. Elle permet de confirmer ou rembourser des achats et résilier des abonnements...

3.3.2 purchaseCase

Ce paramètre est défini dans le catalogue produit (productsCurrent_XXX.xml).
Il permet de définir la cinématique d'un produit ainsi que son type.

Valeurs possibles du purchaseCase :

- 1 : Achat d'un produit à l'acte
- 8 : Achat d'un produit de type abonnement

Ce purchaseCase, lors de requêtes plateforme w-HA → kit marchand, est envoyé sous forme binaire.
Les requêtes https transmises par la plateforme contiendront ces codes en format binaire.

La cohérence du purchaseCase est vérifiée lors des requêtes kit marchand → w-HA.

En cas d'incohérence de valeur (par exemple un purchaseCase = 1 envoyé par le kit, sur un produit P1 de type abonnement), l'achat sera refusé et la cinématique stoppée au niveau de la plateforme w-HA.

3.3.3 Format

Le Kit V4 est compatible avec différents formats de pages : WML, OML, XHTML. Cette configuration peut se faire globalement au niveau du fichier de configuration web.xml.

Elle peut également se faire au niveau de chaque identifiant produit (à chaque appel de servlet en surchargeant le paramètre format). Cette fonctionnalité permet ainsi à l'éditeur d'avoir un kit compatible multi-formats, sans avoir à modifier d'autres paramètres supplémentaires.

Pour activer le formulaire Full Web, le format passé en paramètre ou choisi par défaut doit être le format XHTML (format=xhtml).

3.3.4 Mises en place des fichiers de configuration

Il existe trois fichiers de configuration à paramétrer :

1. web.xml
2. productsCurrent_XXXX.xml (ou productsPlanned_XXXX.xml) : fichiers générés par le MSCA cf. §3.1.1 et §3.2.1.)
3. marchands.xml

3.3.4.1 web.xml

Il contient uniquement le nom des servlets et leur package ainsi que les paramètres de « contexte ».

Description des balises obligatoires du fichier

- **servlet-name** : Nom de la servlet Mappée sur « pos-bundle » ou « admin »

- **servlet-class** : Nom de la classe Java de la servlet,
  com.wha.core.merchant.simulator.MerchantWhaServlet pour « pos-bundle »
  com.wha.core.merchant.simulator.AdminServlet pour « admin »

- **merchantId** : identifiant marchand par défaut

Remarque: Le Kit 3.5 peut fonctionner en mode multi boutique ou mono boutique

- 1er cas multi marchands : Le marchand précise lors de l'appel les identifiants marchands auxquels il souhaite rattacher ses différents produits ou services.
- 2e cas mono marchand : Le marchand ne précise pas de MerchantID lors de l'appel, la servlet prendra alors par défaut la valeur du paramètre « merchantId » du fichier web.xml.

- **merchantCallbackUrl :** URL du kit marchand utilisé par le nœud pour répondre au kit par un redirect http (OML et XHTML) ou on timer (WML). (uniquement pour la servlet « pos-bundle »)

- **merchantCurrency** : Devise du marchand par défaut

- **nodePaymentPanelUrl** : URL de la plateforme w-HA (uniquement pour la servlet « pos-bundle »)

- **nodeResponderUrl** : URL du responder marchand de la plateforme w-HA utilisé dans le cadre de confirmations, d'annulation, de remboursement par le marchand

- **xmlAllProductDatabase** : Liste des noms de fichiers XML où sont contenues les informations des produits. (uniquement pour la servlet « pos-bundle »)

Remarque :

Les fichiers XML des produits doivent respecter la nomenclature suivante :

<nom du fichier de la base des produits et des offres>_<identifiant du marchand>.xml

Exemple : productsCurrent_507.xml l …

Lorsque plusieurs fichiers produits doivent être pris en compte pour différents marchands il faut délimiter chaque nom de fichier XML par un point virgule. La liste doit se terminer également par un point-virgule.

Exemple :
<param-name>xmlAllProductDatabase</param-name>
<param-value>/WEB-INF/productlist**_12.**xml**;**/WEB-INF/base**_507.**xml**;**</param-value>

- **xmlMarchandsDatabase** : Nom du fichier XML où sont contenues les clés, valeurs des clés et URL d'annulation des différents marchands.

- **format** : format par défaut du kit. Valeurs possibles : wml, oml ou xhtml

- **subRequestFileName** : chemin à configurer pour les logs de requêtes générées avant l'appel à w-HA. (uniquement pour la servlet « pos-bundle »)

- **subResponseFileName** : chemin à configurer pour les logs de réponses générées en retour du nœud w-HA. (uniquement pour la servlet « pos-bundle »)

- **consumeFileName** : chemin à configurer pour les logs spécifiques à une consommation dans un bundle. (uniquement pour la servlet « pos-bundle »)

- **messageUrl** : Cette URL est utilisée pour la réponse des actions background et les erreurs. Elle doit être paramétrée en relatif. Ex : /jsp/message.jsp

- **messageWmlUrl :** Cette URL est utilisée pour la réponse des actions background et les erreurs dans le format WML. Elle doit être paramétrée en relatif. Ex : /jsp/message.jsp

3.3.4.2 productsCurrent_XXXXX.xml

Les fichiers XML des produits doivent être déclarés dans la balise **xmlAllProductDatabase** de la servlet pos-bundle du fichier web.xml

Ils doivent respecter la nomenclature suivante :

<nom du fichier de la base des produits et des offres>_<identifiant du marchand>.xml

Exemple : productsCurrent_507.xml

Ce fichier rassemble tous les paramètres de produits, pour une boutique donnée.
Il est découpé en deux parties au maximum : une première partie listant tous les produits du marchand (<product-list>).

```
<valista-product-database>
<product-list>
        <product>
                …
                …
        </product>
</product-list>

</valista-product-database>
```

Si la balise <product-list> est présente, un produit au moins doit être renseigné.

Description des balises obligatoires pour un **produit**

- **<productId>**P1**</ productId >** : identifiant externe du produit

- **<fulfillmentUrl>**http://www.monsite.com/acte/P1.jsp**</fulfillmentUrl>** : Url de redirection vers l'URL de livraison du service choisi.

- **<purchaseCase>**1**</purchaseCase>** : type de l'achat (acte, abonnement, bundle sec). La valeur peut être laissée vide, mais la balise est obligatoire.

Liste des valeurs possibles :
- 1 : Achat d'un produit à l'acte
- 8 : Achat d'un produit de type abonnement

- **<autoConfirm>**true**</autoConfirm>** : confirmation automatique (true) ou non (false) en background du marchand

Remarque importante : Si le paramètre est en mode false, l'utilisateur obtiendra son produit sans avoir confirmé sa transaction. Le paramètre « AutoConfirm » doit être à « true » pour tous les produits ; une erreur de paramétrage en ce domaine pourra entraîner la non comptabilisation des achats et des reversements.

Exemple de configuration d'un produit

<product>

<productId>P2</productId>

<fulfillmentUrl>http://www.monsite.com/P2.jsp</fulfillmentUrl>

<purchaseCase>1</purchaseCase>

<autoConfirm>true</autoConfirm>

</product>

3.3.4.3 marchands.xml

Ce fichier contient les caractéristiques de chaque boutique. Ce fichier permet de rendre le kit multi boutiques, puisque une ou plusieurs boutiques peuvent être définies aussi.

Description des balises obligatoires du fichier

- **mctId** : identifiant de la boutique
- **mctKeyId** : identifiant de la clé du marchand (en général égal au mctId)
- **mctKey** : valeur de la clé du marchand
- **mcttrxCancelFromPaymentPanelUrl :** URL de redirection d'un client dans le cas d'une annulation
- requestFilename : chemin à configurer pour les logs de requêtes générées avant l'appel à w-HA.
- responseFilename : chemin à configurer pour les logs de réponses générées en retour du nœud w-HA.
- consumeFilename : chemin à configurer pour les logs spécifiques à une consommation dans un bundle. (non utilisé)

3.4 Installation

3.4.1 Composants de l'application « Kit_V4 »

L'application « **Kit_V4** » est composée des éléments suivants :

Un fichier de configuration principal	web.xml
Un ou plusieurs Catalogue(s) produits	productsCurrent_XXX.xml
Une base marchand	marchands.xml
Une Servlet Java pour les différentes actions d'achat	pos-bundle
Une Servlet Java pour les différentes actions d'administration	admin
Une boutique de test de l'application	/Kit_V4/xhrml/
Librairies applicatives	/Kit_V4/WEB-INF/lib
Fichier de version	version.txt

3.4.2 Installation du kit

Le fichier de configuration web.xml doit être stocké dans le répertoire /WEB-INF d'installation (ex : C:\Tomcat90\webapps\Kit_V4).

Les fichiers de configuration des marchands et les catalogues produits peuvent être stockés dans le répertoire /WEB-INF ou dans un sous répertoire de /WEB-INF.

Deux fichiers compressés contenant les classes Java doivent être présents dans le répertoire /WEB-INF/lib : valista_vXX.jar et wha_vYY.jar (XX et YY représentant les numéros de versions de ces fichiers). **Pour le Kit V4**, un seul jar est nécessaire (wha_vZZ.jar).

Les pages de l'éditeur de services peuvent être hébergées en dehors du répertoire d'installation dans un serveur web (Apache par exemple) ou directement dans le répertoire d'installation du kit.

Exemple de page stockée sur le serveur marchand:
C:\Tomcat90\webapps\Kit_V4\xhtml\index.xhtml

3.5 Exemples de fichiers de configuration

3.5.1 web.xml

```
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN"
"http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">
<web-app>

    <servlet>
        <servlet-name>MerchantServlet</servlet-name>
        <servlet-class>com.wha.core.merchant.simulator.MerchantWhaServlet</servlet-
class>
        <init-param>
            <param-name>merchantId</param-name>
            <param-value>500</param-value>
        </init-param>
        <init-param>
            <param-name>messageUrl</param-name>
            <param-value>/jsp/message.jsp</param-value>
        </init-param>
        <init-param>
            <param-name>messageWmlUrl</param-name>
```

```
                <param-value>/jsp/wmlMessage.jsp</param-value>
            </init-param>
            <init-param>
                <param-name>merchantErrorUrl</param-name>
                <param-value>/Kit_V4/wml/error_message.wml</param-value>
                <description>Provide the error page URL</description>
            </init-param>
            <init-param>
                <param-name>merchantCallbackUrl</param-name>
                <param-value>http://www.monsite.com/Kit_V4/pos-bundle</param-value>
            </init-param>
            <init-param>
                <param-name>nodePaymentPanelUrl</param-name>
                <param-value>https://whamobile.orange.fr/app-bundlepurchase/node</param-
value>
            </init-param>
            <init-param>
                <param-name>nodeResponderUrl</param-name>
                <param-value>https://orange.w-ha.com/app-node-mct/responder</param-
value>
            </init-param>
            <init-param>
                <param-name>nodeInconsistencyUrl</param-name>
                <param-value>https://orange.w-ha.com/app-node-pcc/inconsistency</param-
value>
            </init-param>
            <init-param>
                <param-name>merchantCurrency</param-name>
                <param-value>EUR</param-value>
            </init-param>
            <init-param>
                <param-name>xmlAllProductDatabase</param-name>
                <param-value>/WEB-INF/products_MERCHANTID.xml;/WEB-
INF/products_MERCHANTID.xml;</param-value>
            </init-param>
            <init-param>
                <param-name>xmlMarchandsDatabase</param-name>
                <param-value>/WEB-INF/marchands.xml</param-value>
            </init-param>
            <init-param>
                <param-name>format</param-name>
                <param-value>xhtml</param-value>
            </init-param>
            <init-param>
```

```
                <param-name>timestampDelay</param-name>
                <param-value>300000</param-value>
            </init-param>
            <init-param>
                <param-name>masqueBitInteger</param-name>
                <param-value>11111111</param-value>
            </init-param>
    </servlet>

    <servlet-mapping>
            <servlet-name>MerchantServlet</servlet-name>
            <url-pattern>/pos-bundle</url-pattern>
    </servlet-mapping>

    <servlet>
            <servlet-name>AdminServlet</servlet-name>
            <servlet-class>com.wha.core.merchant.simulator.AdminServlet</servlet-class>
            <init-param>
                <param-name>merchantId</param-name>
                <param-value>500</param-value>
            </init-param>
            <init-param>
                <param-name>messageUrl</param-name>
                <param-value>/jsp/message.jsp</param-value>
            </init-param>
            <init-param>
                <param-name>messageWmlUrl</param-name>
                <param-value>/jsp/wmlMessage.jsp</param-value>
            </init-param>
            <init-param>
                <param-name>nodeResponderUrl</param-name>
                <param-value>https://orange.w-ha.com/app-node-mct/responder</param-
value>
            </init-param>
            <init-param>
                <param-name>merchantCurrency</param-name>
                <param-value>EUR</param-value>
            </init-param>
            <init-param>
                <param-name>xmlMarchandsDatabase</param-name>
                <param-value>/WEB-INF/marchands.xml</param-value>
            </init-param>
            <init-param>
                <param-name>format</param-name>
```

```
                <param-value>xhtml</param-value>
            </init-param>
    </servlet>

    <servlet-mapping>
            <servlet-name>AdminServlet</servlet-name>
            <url-pattern>/admin</url-pattern>
    </servlet-mapping>

    <welcome-file-list>
            <welcome-file>index.html</welcome-file>
            <welcome-file>index.wml</welcome-file>
    </welcome-file-list>
</web-app>
```

3.5.2 productsCurrent_500.xml

```
<?xml version='1.0' encoding='ISO-8859-1'?>

<valista-product-database>
    <product-list>
            <product>
                    <productId>P1</productId>
    <fulfillmentUrl>http://www.monsite.com/produit1.jsp</fulfillmentUrl>
                    <purchaseCase>1</purchaseCase>
                    <autoConfirm>true</autoConfirm>
            </product>

            <product>
                    <productId>A2</productId>
<fulfillmentUrl> http://www.monsite.com/abonnement2.jsp</fulfillmentUrl>
                    <purchaseCase>8</purchaseCase>
                    <autoConfirm>true</autoConfirm>
            </product>
    </product-list>


</valista-product-database>
```

3.5.3 marchands.xml

```
<?xml version='1.0'?>
<valista-key-database>
    <info>WEB Merchant key Database</info>
    <mct-list>
            <mct>
                    <mctId>500</mctId>
                    <mctKeyId>500</mctKeyId>
                    <mctKey>123456789123456789</mctKey>
                    <mcttrxCancelFromPaymentPanelUrl>http://www.monsite.com
/index.jsp</mcttrxCancelFromPaymentPanelUrl>
                    <requestFilename>request_500.txt</requestFilename>
                    <responseFilename>response_500.txt</responseFilename>
<consumeFilename>response_500.txt</consumeFilename >
            </mct>
            <mct>
                    <mctId>501</mctId>
                    <mctKeyId>501</mctKeyId>
                    <mctKey>123456789123456789</mctKey>
                    <mcttrxCancelFromPaymentPanelUrl>http://www.monsite.com
/index.jsp</mcttrxCancelFromPaymentPanelUrl>
                    <requestFilename>request_501.txt</requestFilename>
                    <responseFilename>response_501.txt</responseFilename>
<consumeFilename>response_501.txt</consumeFilename>
            </mct>
            <mct>
                    <mctId>502</mctId>
                    <mctKeyId>502</mctKeyId>
                    <mctKey>123456789123456789</mctKey>
                    <mcttrxCancelFromPaymentPanelUrl>http://www.monsite.com
/index.jsp</mcttrxCancelFromPaymentPanelUrl>
                    <requestFilename>request_502.txt</requestFilename>
                    <responseFilename>response_502.txt</responseFilename>
<consumeFilename>response_502.txt</consumeFilename>
            </mct>
    </mct-list>
</valista-key-database>
```

4 Mise en place rapide du paiement

Ce paragraphe donne une vision rapide de la mise en place du paiement via le KIT V4.

Nous allons décrire ici la mise en place d'un paiement standard qui sera utilisé dans la majorité des cas.

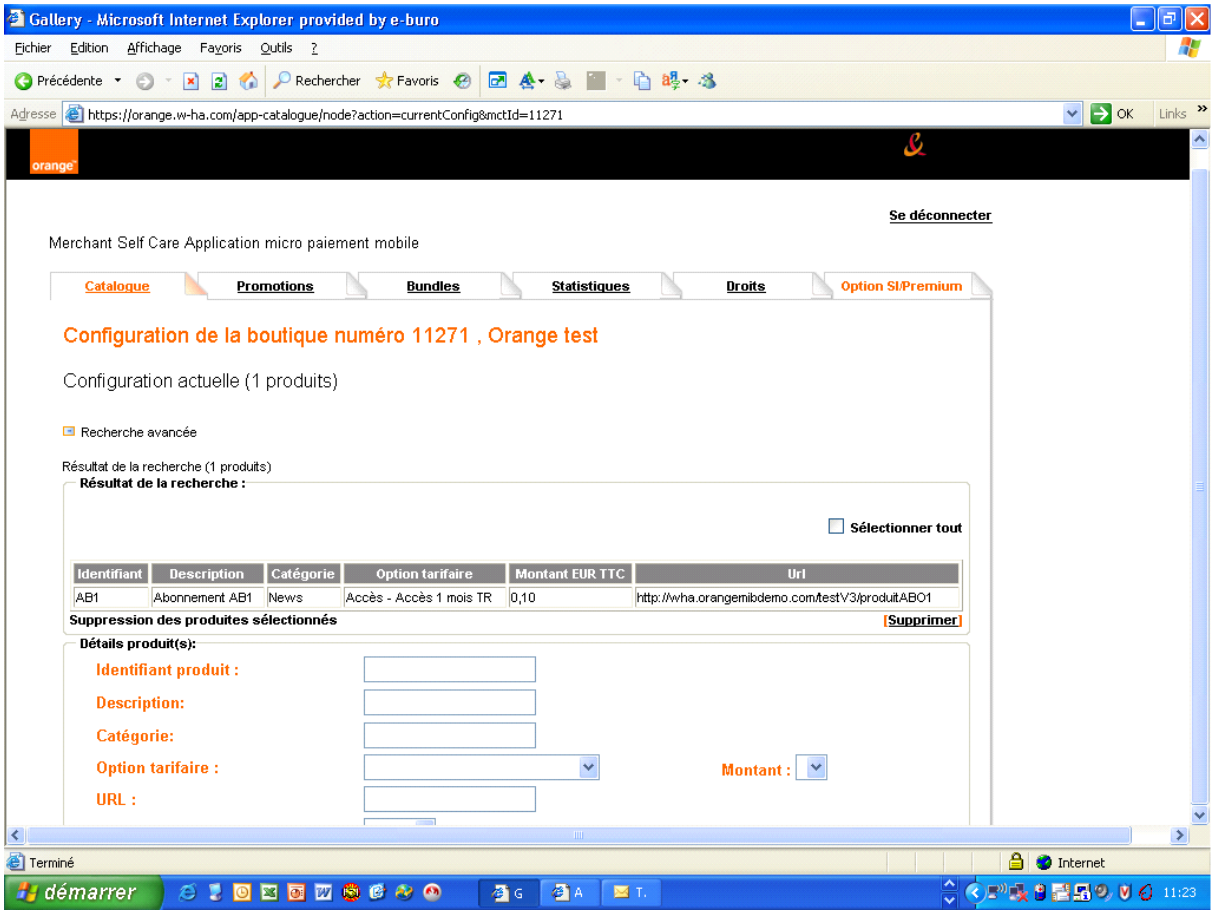
Tous les fichiers utilisés ici sont détaillés dans le chapitre §2.
Toutes les cinématiques utilisées sont détaillées dans le chapitre §4.

4.1 Mise en place du paiement à l'acte

4.1.1 Définir ses produits via le Catalogue centralisé dans le Merchant Self Care Application (MSCA)

Rappel : Le process d'initialisation du Catalogue Centralisé est décrit en §5.3.
Il faut dans un premier temps accéder à l'onglet « Catalogue » du Merchant Self Care Application (MSCA) qui se trouve à l'url suivante :

https://orange.w-ha.com/app-application-manager/node



Déclarer un ou plusieurs produits à l'acte.

Les informations à renseigner sont :

- L'identifiant Produit (par ex. P1)
- Description : le nom du produit qui sera affiché sur le panneau de paiement
- Catégorie : champ libre pouvant être utilisé par le marchand, à des champs statistiques notamment.
- Option tarifaire : acte, accès
- Montant : choix du prix TTC parmi la liste
- URL : URL de livraison vers laquelle sera redirigé le client après le paiement : fulfillmentUrl
- AutoConfirm : True par défaut (ne pas modifier)

Le Merchant Self Care Application fait l'objet d'une documentation spécifique, la consulter pour plus de détail.

N.B. : pour un produit de type abonnement, le changement de tarif ne s'opérera que pour les nouvelles souscriptions, les abonnements en cours continueront d'être reconduit à l'ancien tarif.

4.1.2 Exporter le catalogue

Vous devez exporter le catalogue centralisé dans un fichier XML qui sera utilisé par le kit. L'interface du MSCA vous permet de télécharger directement le fichier appelé productsCurrent_XXXX.xml.

Si votre boutique porte le numéro 500, le fichier doit porter le nom productsCurrent_500.xml.

4.1.3 Intégrer le fichier dans le KIT V4

Copiez ce fichier dans le répertoire WEB-INF du KIT V4.
Vous devez maintenant avoir au moins trois fichiers de configuration :

- Fichier de paramétrage général du KIT : web.xml
- Fichier de boutiques : marchands.xml
- Et au moins un fichier de produits : productsCurrent_XXXX.xml

Vérifiez que le fichier ou les fichiers produits sont déclarés dans le fichier web.xml.

Ils doivent être déclarés dans le paramètre **xmlAllProductDatabase** comme ceci :

<param-name>xmlAllProductDatabase</param-name>
<param-value>/WEB-INF/productsCurrent**_500.**xml**;**/WEB-INF/productsCurrent**_501.**xml**;**</param-value>

Redémarrer le moteur de Servlet pour prendre en compte les nouveaux fichiers.

4.1.4 Intégrer le paiement au site

4.1.4.1 Appel du KIT

L'éditeur doit intégrer un bouton de formulaire ou un lien pour faire appel au KIT et déclencher le processus de paiement pour le client.

L'URL appelée doit être de la forme suivante :

http://www.monsite.com/pos-bundle?action=purchaseListOffer&pid=P1

Le KIT peut être appelé via un lien pointant vers l'URL ou via un formulaire qui pourra être de cette forme :

```
<anchor title='Ok'>Buy Product #1
    <go method='post' href="/pos-bundle">
        <postfield name='action' value='purchaseListOffer'/>
        <postfield name='pid' value='P1'/>
    </go>
</anchor>
<br/>
<anchor title='Ok'>Buy Product #2
    <go method='post' href="/pos-bundle">
        <postfield name='action' value='purchaseListOffer'/>     paramètre obligatoire
        <postfield name='pid' value='P2'/>                       paramètre obligatoire
        <postfield name='format' value='wml'/>                   paramètre surchargé
        <postfield name='logo' value='1234'/>                    paramètre marchand
        <postfield name='mobile' value='06xxxxxxxx'/>            paramètre marchand
    </go>
</anchor><br/>
```

Le client sera alors redirigé vers le panneau de paiement correspondant au produit appelé et si le paiement est validé, vers la page de livraison déclarée dans le fichier de produits.

4.1.4.2 Protection de l'URL de livraison

Il est fortement conseillé de protéger la page de livraison, afin d'éviter que le client puisse revenir sur celle-ci sans passer par le paiement.

Pour cela il faut re-calculer la valeur du HMAC de l'URL de livraison et comparer celle-ci avec le HMAC renvoyé par la plate-forme w-HA.

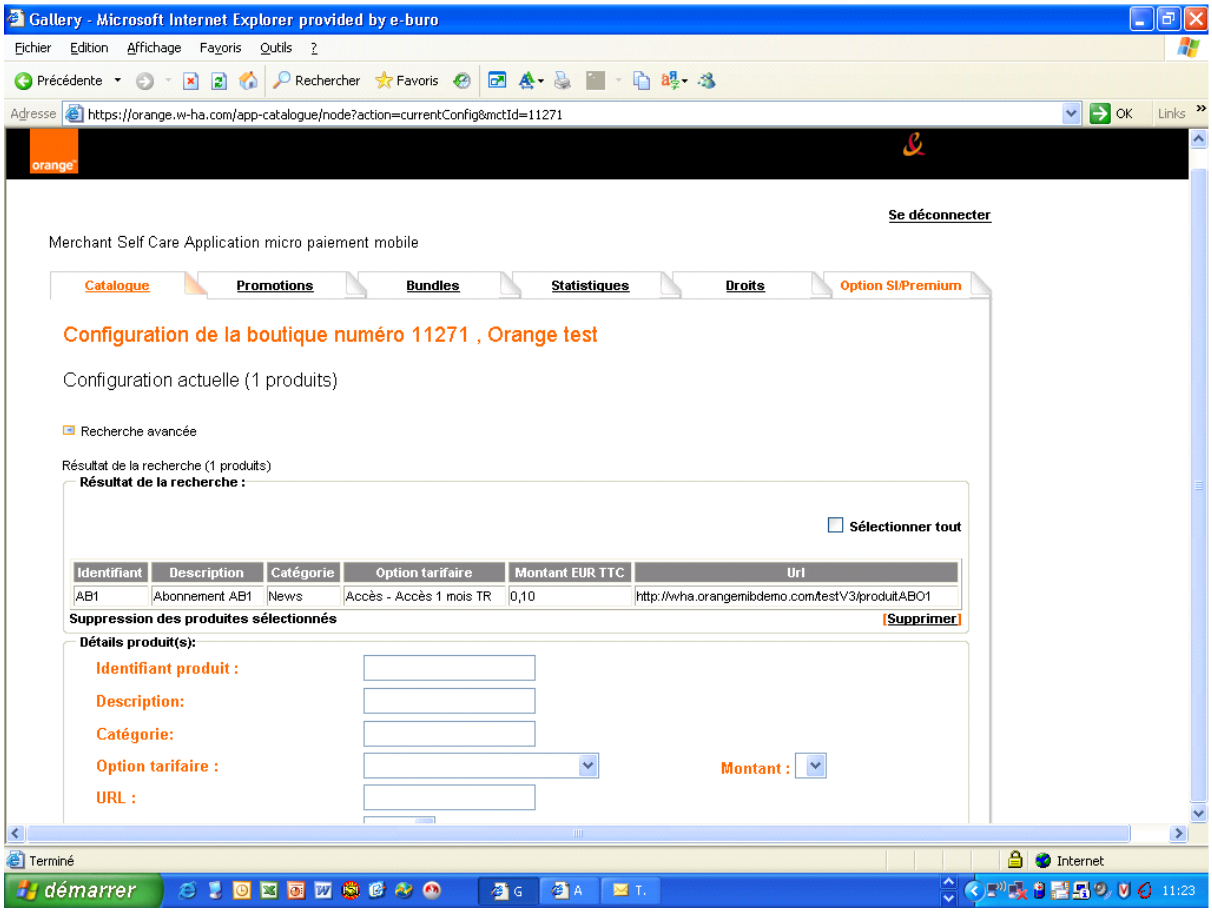
Ainsi, on s'assure que le client est bien passé par w-HA.

Le détail du calcul est décrit au paragraphe §4.1.4 Redirection finale et calcul du HMAC.

4.2 Mise en place du paiement à l'abonnement

4.2.1 Définir ses produits via le Catalogue centralisé dans le Merchant Self Care Application (MSCA)

Rappel : Le process d'initialisation du Catalogue Centralisé est décrit en §5.3.
Il faut dans un premier temps accéder à l'onglet « Catalogue » du Merchant Self Care Application (MSCA) qui se trouve à l'url suivante :
https://orange.w-ha.com/app-application-manager/node



Déclarer un ou plusieurs produits à l'abonnement.

Les informations à renseigner sont :

- L'identifiant Produit (par ex. A1)
- Description : le nom du produit qui sera affiché sur le panneau de paiement
- Catégorie : champ libre pouvant être utilisé par le marchand, à des champs statistiques notamment.
- Option tarifaire : le type d'abonnement
    - Abonnement hebdomardaire tacitement reconductible
    - Accès 1 mois
    - Abonnement mensuel tacitement reconductible
    - Accès 24 heures
- Montant : choix du prix TTC parmi la liste
- URL : URL de livraison vers laquelle sera redirigé le client après le paiement : fulfillmentUrl
- ForbidNewSubscription : si le paramètre est égal à True, aucune souscription à ce produit ne sera possible. Cette option est utilisée pour empêcher les nouvelles souscriptions mais pour l'accès aux abonnés en cours. S'il est à false, la souscription est ouverte.

Le Merchant Self Care Application fait l'objet d'une documentation spécifique, la consulter pour plus de détail.

N.B. : pour un produit de type abonnement, le changement de tarif ne s'opérera que pour les nouvelles souscriptions, les abonnements en cours continueront d'être reconduit à l'ancien tarif.

4.2.2 Exporter le catalogue

Vous devez exporter le catalogue centralisé dans un fichier XML qui sera utilisé par le kit. L'interface du MSCA vous permet de télécharger directement le fichier appelé productsCurrent_XXXX.xml.

Si votre boutique porte le numéro 500, le fichier doit porter le nom productsCurrent_500.xml.

4.2.3 Intégrer le fichier dans le KIT V4

Copiez ce fichier dans le répertoire WEB-INF du KIT V4.
Vous devez maintenant avoir au moins trois fichiers de configuration :

- Fichier de paramétrage général du KIT : web.xml
- Fichier de marchands : marchands.xml
- Et au moins un fichier de produits : productsCurrent_XXXX.xml

Vérifiez que le fichier ou les fichiers produits sont déclarés dans le fichier web.xml.

Ils doivent être déclarés dans le paramètre **xmlAllProductDatabase** comme ceci :

<param-name>xmlAllProductDatabase</param-name>
<param-value>/WEB-INF/productsCurrent**_500.**xml**;**/WEB-INF/productsCurrent**_501.**xml**;**</param-value>

Redémarrer le moteur de Servlet pour prendre en compte les nouveaux fichiers.

4.2.4 Intégrer le paiement au site

4.2.4.1 Appel du KIT

L'éditeur doit intégrer un bouton de formulaire ou un lien pour faire appel au KIT et déclencher le processus de paiement pour le client.

L'URL appelée doit être de la forme suivante :

http://www.monsite.com/pos-bundle?action=purchaseListOffer&pid=P1

Le KIT peut être appelé via un lien pointant vers l'URL ou via un formulaire qui pourra être de cette forme :

```
<anchor title='Ok'>Buy Product #1
    <go method='post' href="/pos-bundle">
            <postfield name='action' value='purchaseListOffer'/>
            <postfield name='pid' value='A1'/>
    </go>
</anchor>
<br/>
<anchor title='Ok'>Buy Product #2
    <go method='post' href="/pos-bundle">
            <postfield name='action' value='purchaseListOffer'/>     paramètre obligatoire
            <postfield name='pid' value='A2'/>                       paramètre obligatoire
            <postfield name='format' value='wml'/>                   paramètre surchargé
            <postfield name='profil' value='toto123'/>               paramètre marchand
            <postfield name='mobile' value='06xxxxxxxx'/>            paramètre marchand
    </go>
</anchor><br/>
```

Le client sera alors redirigé vers le panneau de paiement correspondant au produit appelé et si le paiement est validé, vers la page de livraison déclarée dans le fichier de produits.

4.2.4.2 Protection de l'URL de livraison

Il est fortement conseillé de protéger la page de livraison, afin d'éviter que le client puisse revenir sur celle-ci sans passer par le paiement.

Pour cela il faut re-calculer la valeur du HMAC de l'URL de livraison et comparer celle-ci avec le HMAC renvoyé par la plate-forme w-HA.

Ainsi, on s'assure que le client est bien passé par w-HA.

Le détail du calcul est décrit au paragraphe §4.1.4 Redirection finale et calcul du HMAC.

4.3 Particularités Full Web

4.3.1 Panneau Full Web « small »

4.3.1.1 Taille du panneau Full Web « Small »

La taille du panneau de paiement Full Web de type Small est :
- Largeur = 490 px
- Hauteur = 305 px

4.3.1.2 Paramètre d'appel

Pour activer ce panneau de paiement, il suffit d'ajouter le paramètre suivant lors de l'appel au Kit V4 :
- altDisplay=1

Par exemple :
http://monsite.com/Kit_V4/pos-bundle?action=purchaseListOffer&pid=P1&**altDisplay=1**

Toutes autres valeurs du paramètre « altDisplay » entraînera l'affichage du panneau Full Web classique.

4.3.1.3 Intégration iFrame

Pour un affichage idéal, la page doit être présentée dans une iframe avec les caractéristiques suivantes :
- Taille de l'iframe = 490*305 px
- Pas de scrollbar

Par exemple :

<iframe id="Iframe1" frameborder="0" vspace="0" hspace="0" marginheight="30px" marginwidth="30px" **width="490px" height="305px" scrolling="no" src="index_iplusmobile.html"**></iframe>

La page « index_iplusmobile.html » sera une page éditeur contenant un lien (ou un bouton) appelant le Kit V4 avec le paramètre d'appel « altDisplay=1 ».

4.3.2 Pré-saisie du MSISDN

Pour saisir automatiquement le MSISDN du client, il suffit d'ajouter le paramètre suivant lors de l'appel au Kit V4 :

- webId=06AABBCCDD

Par exemple :
http://monsite.com/Kit_V4/pos-bundle?action=purchaseListOffer&pid=P1&**webId=0682469418**

Un contrôle sur le format du MSISDN est réalisé par w-HA. Si le format n'est pas correct, le MSISDN ne s'affichera pas dans la panneau de paiement.

4.4 Particularités OTP accéléré (OTP 1 étape)

Pour utiliser la fonctionnalité de l'OTP accéléré, **l'éditeur doit transmettre le msisdn de l'utilisateur directement au serveur W-HA (via le Kit) dans le paramètre webID**. Ceci permettra de passer sur l'écran de paiement directement avec réception d'un code de sécurité (One time Password), que l'utilisateur devrait renseigner.

Si le marchand n'a pas envoyé de paramètre webId avec format correct, l'utilisateur est alors redirigé vers le parcours OTP en 2 étapes correspondant.

5 Description des appels au Kit V4

5.1 Accès client (servlet pos-bundle)

L'utilisateur interagit avec le marchand pour déclencher une transaction. Il faut déterminer le type de panneau de paiement à afficher. Celui-ci est défini selon les paramètres envoyés à la servlet pos-bundle.

5.1.1 Achat d'un produit à l'acte

L'achat se passe en 4 temps :
1. Le client sur le site clique sur un lien qui appelle le KIT V4
2. Le KIT V4 génère l'appel à w-HA grâce aux informations du KIT
3. w-HA traite la demande et envoie une réponse au KIT
4. Le KIT interprète la réponse et redirige le client vers l'URL de livraison

5.1.1.1 Appel du KIT V4

Servlet : pos-bundle
Action : purchaseListOffer : Achat d'un produit.
Paramètres d'entrée : pid, mid, url, purchaseCase, format, cur...

Paramètres obligatoires :

- pid : identifiant externe du produit (présent dans le fichier products_XXX.xml)

Paramètres facultatifs :

- mid : merchantId associé
- purchaseCase : surchage le paramètre de la base produit products_XXX.xml
- productDesc : libellé du produit (sera affiché sur le panneau de paiement. Texte tronqué à 70 caractères, côté serveur, si nécessaire). En cas de paramètre absent, c'est le libellé enregistré dans le catalogue centralisé qui est pris en compte.
- url : url du nœud à contacter par le kit
- format : les formats gérés dépendent de l'application utilisée. Valeur autorisée = xhtml.
- webId : msisdn de l'utilisateur
- operatorId : code opérateur
- altDisplay : affichage alternatif dans le cadre des Achats Web (altDisplay = 1).

Le paramètre **webId** permet à l'éditeur de pré-authentifier l'utilisateur (envoi de son MSISDN). Il est pré-saisi par l'application dans le cas d'un accès à une cinématique avec authentification OTP (Mobile ou Full Web)
Le paramètre **operatorId** permet à l'éditeur d'envoyer le code de l'opérateur associé, ceci afin de permettre à l'application Kiosque Data de pré-charger la bonne charte graphique associée (si cette charte existe déjà, dans le cas contraire c'est la charte Orange par défaut qui reste proposée). Ce paramètre est applicable uniquement pour les cinématiques Full Web. Ci-dessous la liste des codes possibles (en bleu les chartes déjà existantes) :

- ora (Orange)
- sos (Sosh)
- m6m (M6 Mobile)
- vim (Virgin Mobile)
- mym (MyNRJ Mobile)
- tv2 (Tele2)
- bmo (Breizh Mobile)
- mot (Monaco Telecom)
- sym (Symacom Mobile)
- leo (Leo)

Le paramètre **altDisplay** permet à l'éditeur de proposer un affichage alternatif (plus petit) des écrans de paiement Achats Web. Ce paramètre est applicable uniquement pour les cinématiques Full Web

Les paramètres mid, purchaseCase, url et format, non surchargés, auront les valeurs par défaut situées dans le fichier web.xml

Exemple de requête d'appel de la servlet sans paramètre de surcharge :

https://merchant_server/Kit_V4/pos_bundle?action=purchaseListOffer&pid=P1

Exemple de requête d'appel de la servlet avec paramètres de surcharge :

https://merchant_server/Kit_V4/pos_bundle?action=purchaseListOffer&pid=P1&cur=EUR&mid=502&purchasecase=1&url=https://orange.w-ha.com/app-bundlepurchase/node&format=xhtml

5.1.1.2 Appel généré par le KIT V4 vers la plateforme w-HA :

Il s'agit d'une redirection de la servlet pos-bundle du kit marchand. Le message en provenance du kit marchand doit respecter le protocole de message.

Exemple de requête envoyée à la plate-forme w-HA générée par le kit:

https://orange.w-ha.com/app-bundlepurchase/node?m=h=0a3404ba0a6dd0b987d7ffb38d17f7c2;p=502;k=502;v=4:{c=PurchaseTypeReq;v={purchasecase=1;mp={_ap_lg=fr;format=xhtml;};merchantCallbackURL=https://merchant_server/Kit_V4/pos_bundle;pi=P2;t=468da447bd1c4821bbc5def0498fd441;}}

Description du protocole de requête envoyé au nœud :

m = message avec contenu encodé en UTF-8 :

- h=hmac généré avec les paramètres de la requête et la clé marchand
- p=identifiant de l'éditeur de service
- k=identifiant de la clef
- v=la version du protocole (3 ou 4)
- c=PurchaseTypeReq, requête d'achat
- purchaseCase : type de l'achat, 1 pour un achat à l'acte
- merchantCallbackURL=URL de retour vers le kit de l'éditeur de service
- mp=paramètres de l'éditeur de service fournis à la servlet du kit.
- pi=identifiant du produit
- t=token de la requête (généré avec le Kit V4 uniquement)

Calcul du hmac :

Le paramètre h est généré en fonction des paramètres de la requête et de la clé marchand (HMAC-MD5 pour le kit V3, HMAC-SHA256 pour le kit V4).
Dans l'exemple ci-dessus, les paramètres pris en compte pour le chiffrement sont :

c=PurchaseTypeReq;v={purchasecase=1;mp={_ap_lg=fr;format=xhtml;};merchantCallbackURL=https://merchant_server/Kit_V4/pos_bundle;pi=P2;t=468da447bd1c4821bbc5def0498fd441;}

5.1.1.3 Réponse de la plateforme w-HA :

S'il y a acceptation de l'achat le client est redirigé sur la servlet pos-bundle du kit marchand avec le message `PurchaseTypeSuccess`.

Exemple de message retour:

m=h=e982dfaafdd237191229afca6c9ef6cb;p=502;k=502;v=4:{c=PurchaseTypeSuccess;v={pid=P2;purchasecase=1;responderURL=https://orange.w-ha.com/app-node-mct/responder;mp={_ap_lg=fr;format=xhtml;};puid=105-5189182275232667;amt=1;}}

Description du protocole du message reçu par le kit :
m : message

- h=le hmac généré avec la clef (keyvalue), il protège des éventuelles modifications des valeurs des paramètres et permet d'authentifier l'éditeur de service.
- p=identifiant de la boutique
- k=identifiant de la clef
- v=version du protocole (3 ou 4)
- c=message retour, PurchaseTypeSuccess dans le cas d'un achat réussi.
- pid=identifiant externe du produit
- mp=paramètres de l'éditeur de service fournis à la servlet du kit.
- puid=identifiant de transaction
- purchaseCase : type de l'achat 1 pour un achat à l'acte
- responderURL : URL du responder pour la confirmation d'un achat acte
- amt=montant du produit

Remarque : l'identifiant du produit permet de récupérer les paramètres liés au produit dans le fichier productsCurrent_XXX.xml

5.1.1.4 Redirection finale du client : fulfillmentUrl

Il s'agit d'une redirection ou on timer (format=wml) effectuée par la servlet pos-bundle du kit marchand vers une page définie par le marchand dans la base produit du kit. Cette page doit être capable de gérer dynamiquement (script php ou autres) le langage supporté par le terminal, grâce à la récupération du paramètre « format » renvoyé par la plate-forme de paiement dans la fulfillmentUrl.

Les paramètres suivants sont envoyés :

- hmac=calculé avec la fulfillmentUrl + ses paramètres
- trxId=Identifiant externe de l'achat
- les paramètres du marchand classés par ordre alphabétique

Lors de la redirection du message vers l'url « fulfillmentURL », le calcul du HMAC contient le paramètre trxId.

Exemple de fullfilmentUrl :

https://merchant_server/Kit_V4/jsp/product.jsp?hmac=
d694596371784c17ea3704ac6c77ec4ff38354aab1ec6622a5e8ecd981fc987a&lg=fr&trxId=105-
5189182275232667&format=xhtml

Exemple de fullfilmentUrl (achat en mode WIFI) :

Dans le cas d'une utilisation du Kit en mode de connexion WIFI (sur mobile ou sur PC), l'alias sera rajouté automatiquement aux propriétés marchand (mp), et donc dans les paramètres de la fullfilmentUrl.

https://merchant_server/Kit_V4/jsp/product.jsp?hmac=
d694596371784c17ea3704ac6c77ec4ff38354aab1ec6622a5e8ecd981fc987a&lg=fr&trxId=105-
5189182275232667&format=xhtml&alias=328585576733

Cet alias ne sera ajouté que pour des connexions en mode WIFI, et pour des marchands dont la clé d'alias a été fournie à w-HA. Cet alias sera également rajouté pour les autres actions listées ci-dessous (achat d'un produit à l'abonnement, refus d'un achat).

5.1.2 Achat d'un produit en abonnement

L'achat se passe en 4 temps :
1. Le client sur le site clique sur un lien qui appelle le KIT V4
2. Le KIT V4 génère l'appel à w-HA grâce aux informations du KIT
3. w-HA traite la demande et envoie une réponse au KIT
4. Le KIT interprète la réponse et redirige le client vers l'URL de livraison

5.1.2.1 Appel du KIT V4

Servlet : pos-bundle
Action : purchaseListOffer.
Paramètres d'entrée : pid, mid, url, purchaseCase, format, cur…

Paramètres obligatoires :

- pid : identifiant externe du produit (présent dans le fichier products_XXX.xml)

Paramètres facultatifs :

- mid : merchantId associé
- purchaseCase : surchage le paramètre de la base produit products_XXX.xml
- url : url du nœud à contacter par le kit
- format : les formats gérés dépendent de l'application utilisée, les valuers sont wml, oml et xhtml.
- webId : msisdn de l'utilisateur
- operatorId : code opérateur
- altDisplay : affichage alternatif dans le cadre des Achats Web (altDisplay = 1).

Les paramètres mid, purchaseCase, url et format, non surchargés, auront les valeurs par défaut situées dans le fichier web.xml

Exemple de requête d'appel de la servlet sans paramètre de surcharge :

https://merchant_server/Kit_V4/pos_bundle?action=purchaseListOffer&pid=A3

Exemple de requête d'appel de la servlet avec paramètres de surcharge :

https://merchant_server/Kit_V4/pos_bundle?action=purchaseListOffer&pid=A3&cur=EUR&mid=502&purchasecase=8&url=https://orange.w-ha.com/app-bundlepurchase/node&format=xhtml

5.1.2.2 Appel généré par le KIT V4 vers la plateforme w-HA :

Il s'agit d'une redirection de la servlet pos-bundle du kit marchand.

Appel vers la plateforme w-HA :

Il s'agit d'une redirection de la servlet pos_bundle du kit marchand. Le message en provenance du kit marchand doit respecter le protocole de message.

Exemple de requête envoyée à la plateforme W-HA :

https://orange.w-ha.com/app-bundlepurchase/node?m=h=1633d348d1a0474e0221eeb1761a9130;p=502;k=502;v=4:{c=PurchaseTypeReq;v={purchasecase=8;mp={_ap_lg=fr;format=xhtml;};merchantCallbackURL=https://merchant_server/Kit_V4/pos-bundle;pi=A3;t=468da447bd1c4821bbc5def0498fd441;}}

Description du protocole de requête envoyé à W-HA :

- m : message donc le contenu est encodé en UTF-8
    - h=hmac généré avec les paramètres de requête et clé marchand
    - p=identifiant de l'éditeur de service
    - k=identifiant de la clef
    - v=version du kit (3 ou 4)
    - c=PurchaseTypeReq : Requête d'achat
    - purchaseCase : type de l'achat, 8 pour l'achat d'un abonnement
    - merchantCallbackURL=Url de retour vers le kit de l'éditeur de service
    - mp=paramètres de l'éditeur de service fournie à la servlet du kit.
    - pi=identifiant du produit
    - t=token de la requête (Kit V4 uniquement)

5.1.2.3 Réponse de la plateforme W-HA :

S'il y a acceptation de l'achat le client est redirigé sur la servlet pos-bundle du kit marchand avec le message `PurchaseTypeSuccess`.

Exemple de message retour:

h=3e3df038bdbd9d90eefd0082dbc723c1;p=502;k=502;v=4:{c=PurchaseTypeSuccess;v={pid=A3;purchasecase=1000;responderURL=https://orange.w-ha.com/app-node-mct/responder;mp={schId=5;_ap_lg=fr;format=xhtml};puid=3040;amt=1;}}

Description du protocole du message reçu par le kit :

m : message

- h= le hmac généré avec la clef, il protège des éventuelles modifications des valeurs des paramètres et permet d'authentifier l'éditeur de service.
- p=identifiant du marchand
- k=identifiant de la clef
- v=version du protocole (3 ou 4)
- c=message retour, PurchaseTypeSuccess dans le cas d'un achat réussi.
- pid=identifiant externe du produit
- mp=paramètres de l'éditeur de service fournis à la servlet du kit.

- puid=identifiant de l'abonnement
- purchaseCase : type de l'achat, 1000 (8 en binaire) pour l'achat d'un abonnement
- responderURL : URL du responder pour la confirmation (pas pour l'abonnement)
- amt=montant du produit

Remarque : l'identifiant du produit permet de récupérer les paramètres liés au produit dans le fichier productsCurrent_XXX.xml.

5.1.2.4 Redirection finale du client : fulfillmentURL

Il s'agit d'une redirection ou on timer (format=wml) effectuée par la servlet pos-bundle du kit marchand vers une page définie par le marchand dans la base produit du kit. Cette page doit être capable de gérer dynamiquement (script php ou autres) le langage supporté par le terminal, grâce à la récupération du paramètre « format » renvoyé par la plate-forme de paiement dans la fulfillmentUrl.

Les paramètres suivants sont envoyés :

- hmac=calculé avec la fulfillmentURL + ses paramètres
- schId=identifiant du type d'abonnement :
  - 2=1 semaine tacitement reconductible
  - 3=1 mois
  - 4=1 mois tacitement reconductible
  - 5=24 heures
- subId=Identifiant de l'abonnement
- les paramètres du marchand classés par ordre alphabétique

Lors de la redirection du message vers l'url « fulfillmentURL », le calcul du HMAC contient les paramètres subId et SchId.

Exemple de fullfilmentUrl :

https://merchant_server/Kit_V4/jsp/product.jsp?hmac=
d694596371784c17ea3704ac6c77ec4ff38354aab1ec6622a5e8ecd981fc987a&lg=fr&schId=5&trxId=3040&format=xhtml

En cas de paiement en WIFI, si l'éditeur a fourni une clé d'aliasing, l'alias est transmis en paramètre dans la fulfillmentUrl.

Celui-ci doit être pris en compte dans le calcul du HMAC.

Exemple de fulfillmentUrl avec alias :

Exemple de fullfilmentUrl (achat en mode WIFI) :

Dans le cas d'une utilisation du Kit en mode de connexion WIFI (sur mobile ou sur PC), l'alias sera rajouté automatiquement aux propriétés marchand (mp), et donc dans les paramètres de la fullfilmentUrl.

https://merchant_server/Kit_V4/jsp/product.jsp?hmac=
d694596371784c17ea3704ac6c77ec4ff38354aab1ec6622a5e8ecd981fc987a&lg=fr&trxId=105-
5189182275232667&format=xhtml&alias=328585576733

Cet alias ne sera ajouté que pour des connexions en mode WIFI, et pour des marchands dont la clé d'alias a été fournie à w-HA. Cet alias sera également rajouté pour les autres actions listées ci-dessous (achat d'un produit à l'abonnement, refus d'un achat).

En cas de paiement en 3/4G, l'alias est enrichi par PEPSS (plate forme Orange) dans toute réponse en tant que header spécifique.

5.1.3 Refus d'un achat ou d'une consommation

Lorsque l'utilisateur refuse un achat ou une consommation, il est redirigé sur le kit avec le message PurchaseTypeCancel. Le kit interprète le message et redirige le kit sur l'URL mcttrxCancelFromPayementPanelUrl paramétrée dans le fichier marchands.xml avec l'identifiant de la boutique (merchantId).

Exemple de message retour:

m=h=1d702dc1e3a8d1749333aeedc50a1415;p=502;k=502;v=4:{c=PurchaseTypeCancel;v={_ap_
lg=fr;format=xhtml;}}

Description du protocole du message reçu par le kit :
m : message

- h= le hmac généré avec la clef, il protège des éventuelles modifications des valeurs des paramètres et permet d'authentifier l'éditeur de service.
- p=identifiant de la boutique
- k=identifiant de la clef
- v=version du protocole (3 ou 4)
- c=message retour, PurchaseTypeCancel l'utilisateur a refusé.

- v=les paramètres marchand

Redirection finale du client : mcttrxCancelFromPaymentPanelUrl

Il s'agit d'une redirection effectuée par la servlet pos-bundle du kit marchand vers une page définie par le marchand dans le fichier de configuration des boutiques (marchands.xml). Cette page doit être capable de gérer dynamiquement (script php ou autres) le langage supporté par le terminal, grâce à la récupération du paramètre « format » renvoyé par la plate-forme de paiement dans la mcttrxCancelFromPayementPanelUrl.

Les paramètres suivants sont envoyés :

- hmac : calculé avec la fulfillmentURL + les paramètres marchand classées par ordre alphabétique
- les paramètres du marchand (dont le paramètre « format »)

Exemple de mcttrxCancelFromPayementPanelUrl:

http://www.monsite.com/index.php?hmac=d694596371784c17ea3704ac6c77ec4ff38354aab1ec6622a5e8ecd981fc987a &lg=fr&format=wml

En cas de paiement en WIFI, si l'éditeur a fourni une clé d'aliasing, l'alias est transmis en paramètre dans l'URL d'annulation.

Celui-ci doit être pris en compte dans le calcul du HMAC.

Exemple de mcttrxCancelFromPayementPanelUrl:

https://merchant_server/Kit_V4/xhtml/index.xhtml?hmac=d694596371784c17ea3704ac6c77ec4ff38354aab1ec6622a5e8ecd981fc987a &lg=fr&format=xhtml

5.1.4 Redirection finale et calcul du hmac

Chaque cinématique de paiement réussie (achat à l'acte, à l'abonnement) se termine par la redirection finale du kit vers la fulfillmentURL, spécifique à chaque produit.

L'url de redirection finale est de la forme :

http://www.monsite.com/product.jsp?hmac=d694596371784c17ea3704ac6c77ec4ff38354aab1ec6622a5e8ecd981fc987a &cid=105-C1190013481100114&lg=fr&oid=B1&pid=P1&uoid=105-U1110838811207154&format=wml

En plus de la fulfillmentURL configurée par l'éditeur, des paramètres sont rajoutés à l'url (merchant properties + identifiant de transaction), ainsi qu'un paramètre hmac encrypté.

Le calcul du hmac est réalisé de la façon suivante :

- Toutes les merchant properties sont récupérées.
- Le ou les identifiants sont ajoutés selon les cas (voir les chapitres sur les appels du kit)
- L'alias est ajouté en cas de paiement en WIFI (et si la boutique a une clé d'aliasing)
- Toutes ces valeurs sont triées par ordre alphabétique, puis concaténées selon la forme suivante :
  `AvalueName1=value1&BvalueName2=value2&CvalueName3=value3`
- Le hmac est finalement calculé en prenant en paramètre cette chaîne un utilisant comme clé de calcul la clé de la boutique (keyvalue)

L'affichage du hmac dans la fulfillmentURL permet ainsi à l'éditeur, s'il le souhaite, de vérifier l'intégrité et la validité des paramètres envoyés à la fulfillmentUrl.

C'est à la charge de l'éditeur de services de développer, sur son serveur d'application, cette fonctionnalité.

5.2 Requêtes serveur à serveur entre l'éditeur et w-HA (servlet admin)

Il s'agit d'une communication entre le kit marchand et la plateforme w-HA (serveur à serveur) sur protocole https sur le responder de la plate-forme w-HA. Ces requêtes utilisent la servlet « admin » du KIT V4.

5.2.1 Confirmation d'un achat

Un achat à l'acte non confirmé (autoConfirm=false dans la base produit) peut être confirmé ultérieurement en utilisant l'action confirm.
Le montant de confirmation peut être inférieur au montant de la transaction, ce qui entraînera un remboursement de la différence.
Il existe un délai maximum pour confirmer une transaction. Après ce délai, la transaction sera automatiquement annulée.

`confirm` : Action pour confirmation d'une transaction.

Paramètres :

- trx : identifiant de transaction
- amt : montant à confirmer TTC (prix non réduit, hors promotion)
- cur : devise associée
- url : url du nœud w-HA à contacter
- mid : identifiant marchand lié à la transaction (optionnel, à transmettre en mode multi marchands)

Exemple :

```
https://merchant_server/Kit_V4/admin?action=confirm&trx=105-
5189182275232667&amt=1&cur=EUR&url=https://orange.w-ha.com/app-node-
mct/responder
```

Cette action déclenche l'envoi d'une requête à la plate-forme w-HA.

Exemple de requête envoyée à W-HA :

```
https://orange.w-ha.com/app-node-
mct/responder?m=h=ac17dd9f2d2de6a1ee85103020f61ada;p=502;k=502;v=4:{c=m_confirm;
v={s_rate=0;n_amt=1;g_amt=1;v_amt=0;trxId=105-
5189182275232667;v_rate=0;s_amt=0;cur=EUR;t=468da447bd1c4821bbc5def0498fd441;}}
```

Description du message envoyé par le Kit :

m = message :

- h= hmac généré avec la clé et paramètres
- p=identifiant du marchand
- k=identifiant de la clef
- v=version du kit (3 ou 4)
- c=message m_confirm pour une demande confirmation de transaction
- s_rate (non utilisé)
- n_amt= montant net = g_amt

- g_amt=montant
- v_amt (non utilisé)
- trxId=identifiant de la transaction
- v_rate (non utilisé)
- s_amt (non utilisé)
- cur=devise
- t=token (Kit V4 uniquement)

Message retour :

1) En cas de succès, le nœud répond un acquittement positif :

```
c=ack;
```

Un forward (pas de redirection) est fait sur le la messageUrl avec les paramètres fournies précédemment :

- trx : identifiant de transaction
- amt : montant à confirmer TTC
- cur : devise associée
- url : url du nœud w-HA à contacter

2) En cas d'échec, le noeud répond un acquittement négatif :

```
c=ex;v={m=[message erreur];t=[exception];c=[identifiant du code  d'erreur];}
```

Description :

- c = ex, exception
- m = message d'erreur de l'exception
- t = exception retourné par le nœud
- c = identifiant du code d'erreur

Un forward (pas de redirection) est fait sur le la messageUrl avec les paramètres fournies précédemment :

- trx : identifiant de transaction
- amt : montant à confirmer TTC
- cur : devise associée
- url : url du nœud w-HA à contacter

L'exception est alors transmise par l'attribut « message » récupérable par la commande java :

```
request.getAttribute("message");
```

L'attribut message contiendra le message :

```
Confirm for [trxId] failed<br/>[exception]
```

Exemple d'erreur :

```
c=ex;v={m=TRX_NOT_FOUND;t=com.valista.core.api.node.pos.MerchantTransactionManag
er$ConfirmationException;c=0;}
```

Exemple de valeur de l'attribut message :

```
Confirm for « identifiant transaction » failed <br/>
[com.valista.core.api.node.pos.MerchantTransactionManager$ConfirmationException]
TRX_NOT_FOUND
```

Pour savoir si la demande de confirmation a réussi ou échoué, il est donc nécessaire de contrôler la présence de l'attribut message.

5.2.2 Annulation d'un achat

Un achat à l'acte non confirmé (autoConfirm=false dans la base produit) peut être annulé ultérieurement en utilisant l'action cancel.

`cancel` : Action pour annulation de la transaction.

Paramètres :

- trx : identifiant de transaction
- url : url du nœud w-HA à contacter
- mid : identifiant marchand lié à la transaction (optionnel, à transmettre en mode multi-marchands)

```
https://merchant_server/Kit_V4/admin?action=cancel&trx=105-
8174539536141774&url=https://orange.w-ha.com/app-node-mct/responder
```

Cette action déclenche l'envoi d'une requête au nœud.

Exemple de requête envoyée au serveur Kiosque Data :

```
https://orange.w-ha.com/app-node-
mct/responder?m=h=3821c99dc59aa859fae0bef0c2318338;p=502;k=502;v=4:{c=m_cancel;v
={trxId=105-8174539536141774;t=468da447bd1c4821bbc5def0498fd441;}}
```

Description du message envoyé par le kit :

m = message :

- h=le hmac généré avec la clef, il protége des éventuels modifications des valeurs des paramètres et permet d'authentifier l'éditeur de service.
- p=identifiant du marchand
- k=identifiant de la clef

- v=version du kit (3 ou 4)
- c=message m_cancel pour une demande d'annulation de transaction
- trxId=identifiant de la transaction
- t=token (Kit V4 uniquement)

Message retour :

1) En cas de succès, le nœud répond par un acquittement positif :

```
c=ack;
```

Un forward (pas de redirection) est fait sur le la messageUrl avec les paramètres fournis précédemment :

- trx : identifiant de transaction
- url : url du nœud w-HA à contacter

2) En cas d'échec, le noeud répond par un acquittement négatif :

```
c=ex;v={m=[message erreur];t=[exception];c=[identifiant du code d'erreur];}
```

Description :

- c = exception
- m = message d'erreur de l'exception
- t = exception retourné par le nœud
- c = identifiant du code d'erreur

Un forward (pas de redirection) est fait sur le la messageUrl avec les paramètres fournis précédemment :

- trx : identifiant de transaction
- amt : montant à confirmer TTC
- cur : devise associée
- url : url du nœud w-HA à contacter

L'exception est alors transmise dans l'attribut « message » récupérable par la commande java :

```
request.getAttribute("message");
```

L'attribut message contiendra la valeur suivante :

```
Cancel for [trxId] failed<br/>[exception]
```

Exemple d'erreur :

```
c=ex;v={m=TRX_NOT_FOUND;t=com.valista.core.api.node.pos.MerchantTransactionManag
er$CancellationException;c=0;}
```

Exemple de valeur de l'attribut message :

```
Cancel for [trxId] failed <br/>
[com.valista.core.api.node.pos.MerchantTransactionManager$CancellationException]
TRX_NOT_FOUND
```

Pour savoir si la demande d'annulation a réussi ou échoué, il est donc nécessaire de contrôler la valeur de l'attribut message.

Remboursement d'un achat

Une transaction peut être remboursée intégralement avec l'action fullRefund. En cas de remboursement(s) partiel(s) déjà effectué(s), c'est le montant restant qui sera remboursé.

fullRefund : Action de remboursement d'une transaction.

Paramètres :

- trx : identifiant de transaction
- url : url du nœud w-HA à contacter
- mid : identifiant marchand lié à la transaction (optionnel, à transmettre en mode multi marchands)

https://merchant_server/Kit_V4/admin?action=fullRefund&trx=105-5189182275232667&url=https://orange.w-ha.com/app-node-mct/responder

Cette action déclenche l'envoi d'une requête à la plate-forme w-HA.

https://orange.w-ha.com/app-node-mct/responder?m=h=67ff7d7ed125eb7fb7052117c84770fa;p=502;k=502;v=4:{c=m_fullRefund;v={trxId=105-5189182275232667;rid=rq74963;d=0;t=468da447bd1c4821bbc5def0498fd441;}}

Description du message envoyé par le kit :

m = message :

- h = le hmac généré avec la clef, il protége des éventuels modifications des valeurs des paramètres et permet d'authentifier l'éditeur de service.
- p = identifiant du marchand
- k = identifiant de la clef
- v = version du protocole (3 ou 4)
- c = message m_ fullRefund pour une demande de remboursement de transaction
- trxId = identifiant de la transaction
- rid = identifiant de la requête (au choix de l'éditeur)

- d = toujours 0 (non utilisé)
- t = token (Kit V4 uniquement)

Message retour :

1) En cas de succès, le nœud répond par un acquittement positif :
c=ack;

Un forward est fait sur le messageUrl avec les paramètres fournis précédemment :

- trx : identifiant de transaction
- url : url du nœud w-HA à contacter

2) En cas d'échec, le noeud répond par un acquittement négatif :
c=ex;v={m=[message erreur];t=[exception];c=[identifiant du code d'erreur];}

Description :

- c = exception
- m = message d'erreur de l'exception
- t = exception retourné par le nœud
- c = identifiant du code d'erreur

Un forward est fait sur le la messageUrl avec les paramètres fournis précédemment :

- trx : identifiant de transaction
- amt : montant à confirmer TTC
- cur : devise associée
- url : url du nœud w-HA à contacter

L'exception est alors transmise dans l'attribut « message » récupérable par la commande java :
request.getAttribute("message");

L'attribut message contiendra le message :
Full refund for [trxId] failed
 [exception]

Exemple d'erreur :

c=ex;v={m=TRX_NOT_FOUND;t=com.ipin.core.api.node.pos.MerchantTransactionManager$RefundException;c=0;}

Attribut message :

Full refund for « identifiant transaction » failed
[com.valista.core.api.node.pos.MerchantTransactionManager$RefundException]
TRX_NOT_FOUND

Pour savoir si la demande de remboursement a réussi ou échoué, il est donc nécessaire de contrôler la valeur de l'attribut message.

5.2.3 Remboursement partiel d'un achat

Une transaction peut être remboursée partiellement avec l'action partialRefund.

`partialRefund` : Action de remboursement partiel d'une transaction.

Paramètres :

- trx : identifiant de transaction
- amt : montant à rembourser en EUR (2 décimales prises en compte)
- url : url du nœud w-HA à contacter
- mid : identifiant marchand lié à la transaction (optionnel, à transmettre en mode multi marchands)

Le **montant** envoyé peut être **inférieur ou égal** au montant total du produit (dans le cas où il est égal, l'action aura donc les mêmes conséquences qu'une action *fullRefund*).

```
https://merchant_server/Kit_V4/admin?action=partialRefund&trx=105-
5189182275232667&amt=0.99&url=https://orange.w-ha.com/app-node-mct/responder
```

Cette action déclenche l'envoi d'une requête à la plate-forme w-HA.

```
https://orange.w-ha.com/app-node-
mct/responder?m=h=3abf067ad3683781715c63cb4ed73e24;p=502;k=502;v=4:{c=m_partialR
efund;v={trxId=105-
5189182275232667;amt=0.99;t=468da447bd1c4821bbc5def0498fd441;}}
```

Description du message envoyé par le kit :

m = message :

- h = hmac généré avec paramètres de requête et clé marchand.
- p = identifiant du marchand
- k = identifiant de la clé marchand
- v = version du kit (3 ou 4)
- c = message m_ partialRefund pour une demande de remboursement partiel de transaction
- trxId = identifiant de la transaction à rembourser

- amt = montant à rembourser
- t = token (Kit V4 uniquement)

Message retour :

1) En cas de succès, le nœud répond par un acquittement positif :
```
c=ack;
```

Un forward est fait sur le la messageUrl.

2) En cas d'échec, le noeud répond par un acquittement négatif :
```
c=ex;v={m=[message erreur];t=[exception];c=[identifiant du code  d'erreur];}
```

Description :
- c = exception
- m = message d'erreur de l'exception
- t = exception retourné par le nœud
- c = identifiant du code d'erreur

Exemple d'erreur :
```
c=ex;v={m=TRX_NOT_FOUND;t=com.ipin.core.api.node.pos.MerchantTransactionManager$
RefundException;c=0;}
```

Un forward est fait sur le la messageUrl avec les paramètres fournis précédemment.

L'exception est alors transmise dans l'attribut « message » récupérable par la commande java :
```
request.getAttribute("message");
```

L'attribut message contient le message-type :
```
Partial refund for [trxId] failed<br/> [exception]
```

Exemple 1 : délai de remboursement dépassé
```
Partial refund for « identifiant transaction » failed <br/>
[com.valista.core.api.node.pos.MerchantTransactionManager$RefundException]
REFUND_REQUEST_TIMEOUT
```

Exemple 2 : le remboursement demandé dépasse le montant restant à rembourser
```
Partial refund for « identifiant transaction » failed <br/>
[com.valista.core.api.node.pos.MerchantTransactionManager$RefundException]
REFUND_OVERFLOW
```

Pour savoir si la demande de remboursement a réussi ou échoué, il est donc nécessaire de contrôler la valeur de l'attribut message.

5.2.4 Résiliation d'un abonnement

Le marchand peut stopper les reconductions automatiques d'un abonnement utilisateur.
L'abonnement sera ainsi résilié à la prochaine date anniversaire de l'abonnement
L'abonnement doit s'appliquer sur un produit appartenant au marchand exécutant cette action.

cancelSubscription : Action pour arrêter les reconductions d'un abonnement utilisateur.

Paramètres :

- subId : identifiant unique de l'abonnement
- mid : identifiant marchand lié à l'abonnement (optionnel, pour les kits multi-marchands)

https://merchant_server/Kit_V4/admin?action=cancelSubscription&subId=6559179

Cette action déclenche l'envoi d'une requête de résiliation à la plate-forme W-HA.

Exemple de requête envoyée à w-HA :

https://orange.w-ha.com/app-node-mct/responder?m=h=
d694596371784c17ea3704ac6c77ec4ff38354aab1ec6622a5e8ecd981fc987a;p=502;k=502;v=4:{c
=m_closeContract;v={cid=6559179;t=468da447bd1c4821bbc5def0498fd441;}}

Description du message envoyé par le kit :

- m = message :
- h = le hmac généré avec la clef, il protége des éventuelles modifications des valeurs des paramètres et permet d'authentifier de façon sécurisée l'éditeur de service.
- p = identifiant du marchand
- k = identifiant de la clef
- v = version du protocole
- c = message m_ closeContract pour une demande de résiliation d'abonnement
- cid = identifiant de l'abonnement
- t = token

Message retour :

1) En cas de succès, le nœud w-HA répond par un acquittement positif :
c=ack;

Le kit affiche alors la messageUrl (ou messageWmlUrl) avec le paramètre fourni précédemment :

- subId : identifiant de l'abonnement

La réussite de l'action est retournée dans l'attribut message, récupérable par la commande java :

request.getAttribute("message");

L'attribut message aura la valeur suivante : Subscription resiliation for subId [subId] successful.

2) En cas d'échec, le noeud répond par un acquittement négatif :

c=ex;v={m=message erreur;t=exception;c=identifiant du code d'erreur;}

Description :

- c = ex
- m = message d'erreur de l'exception
- t = exception retournée par le nœud
- c = identifiant du code d'erreur

Le kit affiche alors la messageUrl avec le paramètre fourni précédemment :

- subId : identifiant de l'abonnement

L'exception est transmise dans l'attribut `message`, récupérable par la commande java : request.getAttribute("message");

L'attribut message aura la valeur suivante :
Subscription resiliation for subId [subId] failed
[message exception]
11
Exemple de message d'erreur :
c=ex ;v={m=CONTRACT_NOT_FOUND;t=com.valista.core.api.node.pos.MerchantTransactionManager$ContractException;c=0;}

Les différentes erreurs pouvant être retournées sont :
SUBSCRIPTION NOT FOUND : l'identifiant d'abonnement n'existe pas, ou l'abonnement n'est pas de type tacitement reconductible.
MERCHANT NOT OWNER OF SUBSCRIPTION : le marchand associé à l'identifiant d'abonnement ne correspond pas à celui configuré dans le kit.
INCORRECT STATUS FOR SUBSCRIPTION : le statut de l'abonnement n'est pas actif, l'abonnement ne peut donc pas être résilié.
java.lang.NumberFormatException : l'identifiant d'abonnement n'est pas un nombre correct.

Exemple de valeur de l'attribut message :

Subscription resiliation for subId 1603934115 failed
SUBSCRIPTION NOT FOUND

Pour savoir si la demande de résiliation d'abonnement a réussi ou échoué, il est donc nécessaire de contrôler la valeur de l'attribut `message`.

5.2.5 Consultation des transactions d'un abonnement (SubTrxReq)

L'action SubTrxReq permet de récupérer les transactions d'un abonnement. Cette action n'est accesible que dans le kit v4 >= 4.04. La réponse est au format JSON

SubTrxReq : Consultation d'un abonnement et de ses transactions (**Détails dans la section 8.4)**

Paramètres :

- subId : identifiant de l'abonnement
- history : nombre de transactions maximum retournées (limité à 12 mois max d'ancienneté)
- mid : identifiant marchand lié à l'abonnement (optionnel, à transmettre en mode multi marchands)

```
https://merchant_server/Kit_V4/admin?action=SubTrxReq&subId=8766026&history=12&url=https://orange.w-ha.com/app-node-mct/responder
```

Cette action déclenche l'envoi d'une requête à la plate-forme w-HA.

https://orange.w-ha.com/app-node-mct/responder?m=h=88d8bca03aa409a767fd60a85a2dc8b6ede75d4cd11e6bff361f02a7f422bf49;p=502;k=502;v=4:{c=SubTrxReq;v={sId=8766026;t=5a8e9fb7fd2f4045a7652d7c2d77296f;history=12;}}

Description du message envoyé par le kit :

m = message :

- h = le hmac généré avec la clef, il protége des éventuels modifications des valeurs des paramètres et permet d'authentifier l'éditeur de service.
- p = identifiant du marchand
- k = identifiant de la clef
- v = version du protocole (4)
- c = message SubTrxReq pour une consultation d'abonnement avec transactions associées
- sId = identifiant de l'abonnement
- history = nombre de transactions maximum
- t = token

Message retour :

1) En cas de succès, l'application fournit une réponse sous forme d'acquittement, exemple ci-dessous :

```
c=ack;v={transactions={105-
3142522640388718={is_refunded=0;amount=0.0;trx_id=105-
3142522640388718;trx_date=2022-01-14 18:57:52;};105-
4181606443123524={is_refunded=0;amount=0.0;trx_id=105-
4181606443123524;trx_date=2022-01-14
19:00:12;};};subscription={status=active;next_renewal_date=2022-01-28
18:57:52;subscription_date=2022-01-14 18:57:52;};}
```

2) La réponse finale est convertie au format JSON par le kit :

```
[{"status":"active", "next_renewal_date":"2022-01-28 18:57:52",
"subscription_date":"2022-01-14 18:57:52"}, {"is_refunded":"0", "amount":"0.0",
"trx_id":"105-3142522640388718", "trx_date":"2022-01-14 18:57:52"},
{"is_refunded":"0", "amount":"0.0", "trx_id":"105-4181606443123524",
"trx_date":"2022-01-14 19:00:12"}]
```

3) En cas d'échec (par exemple : abonnement non trouvé), l'application transmet une réponse d'erreur :

```
c=ex;v={m=[message erreur];t=[exception];c=[identifiant du code d'erreur];}
```

Description :

- c = exception
- m = message d'erreur de l'exception
- t = exception retournée par le nœud
- c = identifiant du code d'erreur

Exemple d'erreur :

```
c=ex;v={m=SUBSCRIPTION_NOT_FOUND;t=com.wha.core.merchant.data.SubscriptionExcept
ion;c=9;}
[SubscriptionException] MERCHANT_NOT_TRUSTED (marchand incorrect associé à
l'abonnement)
[SubscriptionException] SUBSCRIPTION_NOT_FOUND (identifiant abonnement inconnu)
[MissingParameterException] Missing parameter subId (SUBSCRIPTION IDENTIFIER)
[LocalMessagingException] INVALID_COMMAND_DATA-history (la valeur du paramètre
history n'est pas un nombre entier positif)
[LocalMessagingException] INVALID_COMMAND_DATA-subId (la valeur du paramètre sId
n'est pas un nombre entier positif)
```

5.3 Génération automatique des logs

Le kit marchand permet d'avoir un suivi global de chaque action. Il existe 2 fichiers de logs dont chaque chemin peut être configuré par l'éditeur :

- un fichier de logs d'actions du kit avant l'appel à la plateforme w-HA (fichier de requêtes)
- un fichier de logs d'actions suite à une réponse de la plateforme w-HA (fichier de réponses)

Le format identique pour ces 2 fichiers est le suivant :

date|url|mctId|itemId|purchaseCase|trxId|uoid|action|mp|t|

date : Date au format : yyy-mm-dd hh:mm:ss
url : Url d'appel du serveur W-HA. (renseigné ou non selon le type d'action)
mctId : Identifiant marchand concerné par l'action.
itemId : Identifiant produit
purchaseCase : purchaseCase correspondant à l'action (si action d'achat)
trxId : Identifiant de transaction (fichier de réponses uniquement sur action d'achat acte / abo)
action : Nom l'action (confirm, cancel, error, refund, partialRefund)
mp : Liste des merchant properties (ex : mp1=xxxx,mp2=yyyy,mp3=zzzz)
t : token envoyé par le kit (fichier de requêtes uniquement)

Selon les actions réalisées, certains valeurs ne sont pas toujours remplies (par exemple, l'url est remplie uniquement dans le fichier de requêtes…).

Les erreurs suite à un échec de confirmation d'achat, à un échec d'annulation de transaction sont donc archivées dans ces fichiers (action=error).

5.4 Liste des codes d'erreur

En cas d'acquittement négatif (suite donc à un retour de la plateforme w-HA), une exception est transmise dans l'attribut « message ». Un code d'erreur y est associé. Voici la liste complète de ces différents codes d'erreur :

5.4.1 Codes Erreurs pour les Achats

5.4.1.1 Erreur d'Annulation

com.valista.core.api.node.pos.MerchantTransactionManager$CancellationException :
0 : TRX_NOT_FOUND : Transaction non trouvée
1 : INVALID_TRX_STATUS : Mauvais statut de la transaction (une transaction déjà confirmée ne peut être annulée)
2 : INVALID_MERCHANT_INFO : Mauvais identifiant de la boutique
3 : INVALID_TRX_TYPE : Type de transactions invalide (remboursement / confirmation)

5.4.1.2 Erreur de Résiliation

com.valista.core.api.node.pos.MerchantTransactionManager.ContractException
0 : CONTRACT_NOT_FOUND
1 : MERCHANT_NOT_MEMBER_OF_CONTRACT
2 : INTERNAL

5.4.1.3 Erreur de Remboursement

com.valista.core.api.node.pos.MerchantTransactionManager.RefundException
0 : TRX_NOT_FOUND
1 : INVALID_TRX_STATUS
2 : INVALID_MERCHANT_INFO
4 : INVALID_TRX_TYPE
5 : REFUND_REQUEST_TIMEOUT
6 : ACCOUNT_STATUS_INVALID
7 : REFUND_OVERFLOW
8 : REFUND_BELOW_MINIMUM

6 Retour Serveur vers le Kit

Chaque cinématique de paiement réussie (achat à l'acte, à l'abonnement, accès à un abonnement) se termine par une redirection vers l'url Marchand paramétrée « fullfilmentUrl », et qui peut être spécifique à chaque produit.

L'URL de redirection finale est de la forme :

https://merchant_server/Kit_V4/jsp/product.jsp?hmac=d694596371784c17ea3704ac6c77ec4ff38354aab1ec6622a5e8ecd981fc987a &cid=105-C1190013481100114&lg=fr&oid=B1&pid=P1&uoid=105-U1110838811207154&format=xhtml&t=468da447bd1c4821bbc5def0498fd441

Les paramètres nécessaires au marchand sont rajoutés dans l'url (merchant properties, identifiant de transaction,token), ainsi qu'un paramètre hmac.

Le calcul du hmac est réalisé de la façon suivante :

- Toutes les merchant properties sont récupérées (y compris le paramètre « alias » si l'achat a été fait en mode de connexion WIFI ou sur le WEB, et que le marchand est configuré pour l'aliasing).

- Le préfixe « _ap_ », s'il existe, est effacé des noms de merchant properties.
- Un identifiant unique (nom + valeur), est ajouté selon les cas (trxId dans le cas d'un achat de produit à l'acte, subId dans le cas d'un accès à un abonnement).
- Toutes ces valeurs sont triées par ordre alphabétique, puis concaténées selon la forme suivante : `AvalueName1=value1&BvalueName2=value2&CvalueName3=value3`

L'affichage du hmac dans la fullfilmentUrl permet ainsi à l'éditeur, s'il le souhaite, de vérifier l'intégrité et la validité des paramètres envoyés à la ffUrl. C'est à la charge de l'éditeur de services de développer, sur son serveur d'application, cette fonctionnalité.

7 Sécurisation des URL générées par le Kit

A partir de la version 4.01 du Kit Marchand, un mécanisme de sécurisation est mis en place (gestion de tokens à usage limité dans le temps, où à usage unique dans le cas d'un achat) afin d'empêcher le rejeu d'une même requête :

- après un achat déjà validé (applicable uniquement sur les actions d'achat « purchaseListOffer »)
- après un parcours d'achat déjà avancé (exemple avec le parcours OTP où l'utilisateur est allé jusqu'à l'étape 2 « Saisie du code de sécurité »)
- après une durée définie dans le paramétrage serveur Kiosque Data (durée en minutes, initialement fixée à 1 minute, et applicable sur toutes les actions du kit)

L'envoi d'une URL avec token non conforme aboutit à une page d'erreur (cas d'un parcours d'achat), ou un code d'erreur (cas d'une action Editeur)

Ci-dessous les différentes étapes du mécanisme, du client (le Kit V4) vers le serveur Kiosque Data.

1) Client / Kit Marchand

A chaque action appelée sur le Kit V4, un UUID token est généré automatiquement (32 caractères hexadécimaux).

Ce token est rajouté dans les paramètres de toute requête envoyée vers le serveur W-HA (paramètre « t », comme indiqué dans les exemples d'URLs précédents). Le hmac, calculé et ajouté sur toute requête, tient compte de ce paramètre supplémentaire.

2) Serveur W-HA / Kiosque Data

Pour toute requête reçue par le serveur (et dont la validité a été contrôlée au moyen du hmac comme habituellement), et provenant d'un Kit V4, la validité du token est contrôlée. Deux cas sont possibles :

- Token non utilisé

Le token, non trouvé en en base W-HA, est enregistré, avec une date d'expiration associée (date de création + 1 minute). Le contrôle serveur est passant, et selon le cas :

  - les contrôles avant affichage du panneau de paiement peuvent se poursuivre
  - l'action Editeur est déclenchée (annulation / confirmation / remboursement / résiliation)

- Token déjà utilisé

Le token a déjà été utilisé (existant en base W-HA), un contrôle sur sa validité est effectué. Le token est révoqué s'il entre dans au moins l'un de ces cas :

  - la date d'expiration du token est dépassée
  - un achat a déjà été effectué avec ce même token
  - un parcours d'achat a déjà été avancé avec ce token

Exemple de requête d'achat en Kit V3 :

https://whamobile.orange.fr/app-bundlepurchase/node?m=h=c2a46c301a279df17c8c2037c9e41dec;p=502;k=502;v=3:{c=PurchaseTypeReq;v={purchasecase=1;mp={_ap_lg=fr;format=xhtml;};merchantCallbackURL=https://merchant_server/Kit_V3/pos-bundle;pi=P2;}}

Exemple de la même requête d'achat en Kit V4 :

https://whamobile.orange.fr/app-bundlepurchase/node?m=h=**d694596371784c17ea3704ac6c77ec4ff38354aab1ec6622a5e8ecd981fc987a**;p=502;k=502;**v=4**:{c=PurchaseTypeReq;v={purchasecase=1;mp={_ap_lg=fr;format=xhtml;};merchantCallbackURL=https://merchant_server/**Kit_V4**/pos-bundle;pi=P2;**t=468da447bd1c4821bbc5def0498fd441**;}}

8 Limite d’achats par transaction

8.1 Limite d’achat « utilisateur » par transaction

Cette limite d’achat par transaction est définie au niveau de l’utilisateur, quel que soient les moyens de paiement associés
L’ajout / activation de cette limite, sur un utilisateur spécifique, peut être réalisé par changement de la valeur dans le CSR, pour un utilisateur donné
Pour tous les autres utilisateurs n’ayant ni option particulière, ni eu d’action spécifique dans le CSR, c’est dans ce cas une valeur limite par défaut qui est chargée (100€).

Le contrôle de la limite utilisateur est effectué systématiquement pour tout achat : aucun achat n’est possible au-delà de la limite d’achat « utilisateur » par transaction

Cette limite d’achat est également vérifiée lors d’une reconduction d’abonnement (avec donc suspension éventuelle de l’abonnement)

8.2 Limites d’achat « format tarifaire » par transaction

Ces limites permettent de définir des maximums d’achats par format tarifaire et par type de marchand, et / ou par marchand spécifique.

Pour les achats de type abonnement, le contrôle sur la limite par format tarifaire s’effectue uniquement lors de l’achat initial ; il n’y aura donc pas de contrôles lors des éventuelles reconductions.

8.2.1 Limite d’achat par format tarifaire et par type de marchand

Cette limite permet de définir des maximums d’achats par format tarifaire et par type de marchand.
Les différents types de marchands sont : Orange World, Internet+ Mobile, Direct Billing.

Limite d'achat par transaction par défaut : 100 EUR

Type Marchand	Acte simple	Abonnement 1 semaine	Abonnement 1 mois
Orange World	30	30	30
I+ Mobile	20	3	10
Direct Billing	50	50	50

Tableau des limites selon le format tarifaire et le type de marchand

8.2.2 Limite d'achat par format tarifaire et par marchand

Une limite d'achat peut, de manière facultative, être configurée par format tarifaire et par marchand. Elle permet un contrôle plus précis des limites d'achat selon le type de format et selon un marchand en particulier.

Cette configuration est entièrement facultative pour les marchands :

- si la configuration existe, le contrôle sur cette limite par format tarifaire sera effectué, en plus du contrôle de la limite utilisateur.
- si la configuration n'existe pas, c'est le contrôle de la limite « format tarifaire » par type marchand qui s'appliquera.
  → Par conséquent, pour un marchand spécifique, le paramétrage peut s'effectuer sur 1 à N formats tarifaires.

9 Annexe

9.1 Synchronisation des abonnements

9.1.1 Description fonctionnelle

Cette nouvelle fonctionnalité permet à un éditeur de **récupérer l'ensemble des abonnements en cours** associés à une liste d'identifiants marchands, ainsi qu'un **historique des abonnements clôturés**. Les caractéristiques principales de ces abonnements sont retournées. Ces informations sont écrites dans un fichier plat compressé, dont le téléchargement doit être effectué par l'éditeur.

Le téléchargement du fichier de synchronisation doit se faire en 2 étapes :

1. Une première étape de **demande de synchronisation** avec le paramétrage souhaité par l'éditeur. Cette étape initialise la procédure et lance le démarrage de la synchronisation par w-HA.

2. Une deuxième étape pour le **téléchargement du fichier**, étape accessible uniquement lorsque la génération du fichier s'est terminée.

Les éditeurs ont la possibilité d'utiliser cette fonctionnalité selon 2 méthodes d'accès différentes : via l'interface du MSCA, ou par appel direct en HTTPS, sans interface utilisateur.

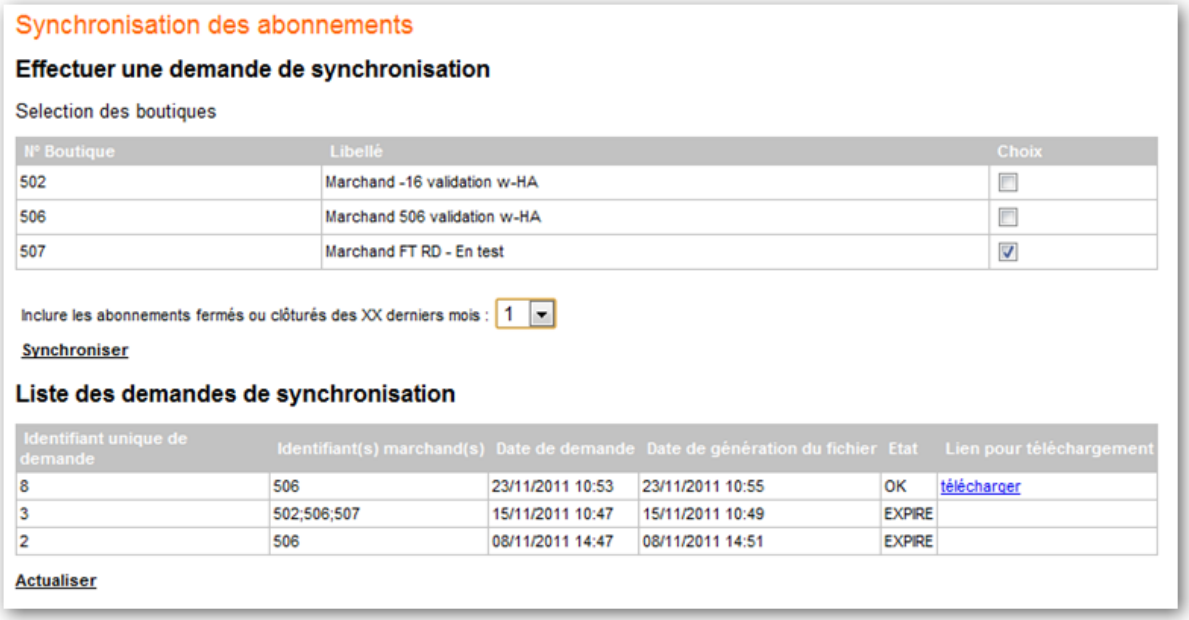
9.1.2 Synchronisation via l'interface MSCA

La fonctionnalité est accessible via un onglet spécifique du MSCA intitulé "Synchronisation des Abonnements

Pour rappel, le MSCA est accessible ici : https://orange.w-ha.com/app-application-manager/node

L'affichage présente 2 parties principales, représentant les deux étapes de la synchronisation :

- Demande de synchronisation
- Téléchargement du fichier de synchronisation



Demande de Synchronisation

L'utilisateur a la possibilité de sélectionner un ou plusieurs identifiants marchands. Le clic sur un lien en dessous du tableau va initialiser une demande de synchronisation d'abonnements pour tous les identifiants marchands sélectionnés :

- tous les abonnements à l'état ouvert ou suspendu, et liés aux identifiants marchands choisis, seront retournés dans le fichier final à télécharger.

- par défaut, tous les abonnements fermés ou clôturés du mois précédent, et liés à ces marchands, seront également synchronisés. L'utilisateur a cependant la possibilité (via une boite de sélection en dessous du tableau), de demander un historique plus ou moins important de ces abonnements fermés, et compris entre 0 et 12 mois.

ATTENTION :
Il existe une limite maximum sur le nombre de demandes de synchronisation par compte et par jour :
10 demandes par compte et par jour.
Si cette limite est atteinte, un message d'erreur en informe l'utilisateur.

Téléchargement du fichier

Le fichier est disponible durant 48H. Au delà, la demande passe en statut expiré et le fichier est supprimé.

Les précédentes demandes de synchronisation (limitées aux 15 dernières demandes du compte MSCA concerné) sont affichées sous forme de tableau.

Liste des demandes de synchronisation

Identifiant unique de demande	Identifiant(s) marchand(s)	Date de demande	Date de génération du fichier	Etat	Lien pour téléchargement
8	506	23/11/2011 10:53	23/11/2011 10:55	OK	télécharger
3	502;506;507	15/11/2011 10:47	15/11/2011 10:49	EXPIRE	
2	506	08/11/2011 14:47	08/11/2011 14:51	EXPIRE	

Actualiser

Ce récapitulatif des dernières demandes permet donc à l'utilisateur de télécharger le fichier de synchronisation souhaité, lorsque cela est possible, grâce au lien affiché dans la dernière colonne.

Les différents cas d'impossibilité de téléchargement sont les suivants :

- Le fichier est toujours en cours de génération (Etat = EN COURS)
- Le fichier date de plus de 48 heures (Etat = EXPIRE)
- Le téléchargement d'un fichier de synchronisation a démarré il y a moins de 60 minutes pour ce même compte

L'identifiant unique de synchronisation est utile pour l'éditeur en cas de demande de téléchargement par la méthode « appel HTTPS ».

9.1.3 Synchronisation par appel direct en HTTPS

La demande de synchronisation et le téléchargement peuvent également s'effectuer directement par appel direct HTTPS sur la plate-forme w-HA, sans passer par l'interface du MSCA.

Afin de sécuriser l'envoi en paramètres du login et du mot de passe des comptes MSCA, toute requête doit être lancée en **HTTPS uniquement**.

La procédure s'effectue également en 2 temps : une première requête de demande de synchronisation avec les paramètres souhaités, et une deuxième requête pour le téléchargement.

Requête de demande de Synchronisation

L'URL d'appel est de type :
https://orange.w-ha.com/app-sync-sub/node?action=syncSubscriptionsRequest&login=xxx&pass=yyy&mctId=502&history=x

Les paramètres sont les suivants :

- **action** :syncSubscriptionsRequest
- **login** : login du compte MSCA effectuant la demande
- **pass** : mot du passe du compte MSCA
- **mctId** : liste des identifiants marchands séparés par « ; ». Les identifiants marchands doivent être associés au compte MSCA passé en paramètre.
- **history (facultatif)**: nombre de mois dont on souhaite l'historique des abonnements fermés / clôturés. La valeur doit être un nombre entier compris entre 0 et 12. Ce paramètre est facultatif : s'il n'est pas envoyé en paramètre, w-HA retourne **par défaut un historique d'un mois**.

Exemple :

Requête de génération d'un fichier de synchronisation portant sur 2 identifiants marchands, et dont on souhaite récupérer l'historique des abonnements clôturés sur les 3 derniers mois :

https://orange.w-ha.com/app-sync-sub/node?action=syncSubscriptionsRequest&login=xxx&pass=yyy&mctId=502;507&history=3

➔ Si cette requête est envoyée le 1er septembre 2011 à 12H00, tous les abonnements ouverts ou suspendus associés aux identifiants marchands 502 et 507 seront retournés, ainsi que tous les abonnements fermés / clôturés et souscrits entre le 1er juin 2011 12H00 et le 1er septembre 2011 12H00.

w-HA retourne en réponse un **identifiant unique de demande de synchronisation** (identifiant numérique). La réponse est quasi-immédiate et la génération du fichier va démarrer.

Format de la réponse :

- synchronizationRequestId = 125358

Cas d'erreurs

Pour toute erreur détectée par le système, un message d'erreur est retourné. La génération du fichier ne s'effectuera pas. Les codes d'erreurs sont les suivants :

- error = missing parameter (xxx)

paramètre manquant dans la requête

- error = invalid value for mctId

identifiant marchand inexistant ou mal formaté

- error = invalid value for history

la valeur doit être un entier entre 0 et 12 inclus

- error = incorrect login or password

login ou mot de passe incorrect

- error = login unauthorized

Accès MSCA n'ayant pas la fonctionnalité « Synchronisation des abonnements » activée

- error = mctId unauthorized with this login

un ou plusieurs des identifiants marchands ne sont pas associés au login MSCA

- error = pending synchronization request (123456)

une génération de fichier est déjà en cours pour ce login

- error = maximum synchronizations reached with this login

Le login a atteint le maximum autorisé de demandes de synchronisations journalières

- error = internal

erreur interne du système

Requête de téléchargement du fichier de synchronisation

L'URL d'appel est de type :

https://orange.w-ha.com/app-sync-sub/node?action=syncSubscriptionsDownload&login=xxx&pass=yyy&synchronizationRequestId=123538

Les paramètres obligatoires sont les suivants :

- **action** : syncSubscriptionsDownload
- **login** : login du compte MSCA effectuant la demande
- **pass** : mot du passe du compte MSCA
- **synchronizationRequestId** : identifiant unique de demande de synchronisation

L'identifiant unique de synchronisation est connu de l'utilisateur grâce à la première requête de demande de synchronisation.

Si la génération du fichier est terminée, le **téléchargement du fichier** s'enclenche automatiquement. Le fichier est au format compressé, voir le chapitre suivant pour les détails sur le contenu du fichier.

Cas d'erreurs

Pour toute erreur détectée par le système, un message d'erreur est retourné. Le téléchargement du fichier ne s'effectuera donc pas. Les codes d'erreurs sont les suivants :

- error = missing parameter (xxx)

paramètre manquant dans la requête

- error = invalid synchronizationRequestId

l'identifiant est inexistant ou mal formaté

- error = incorrect login or password

login ou mot de passe incorrect

- error = synchronizationRequestId unauthorized with this login

la demande de synchronisation n'a pas été effectuée par ce login

- error = login unauthorized

login n'ayant pas la fonctionnalité « Synchronisation des abonnements » activée

- error = synchronizationRequestId expired

la demande de synchronisation pour cet identifiant a expiré

- error = file generation in progress

le fichier est en cours de génération et n'est pas encore prêt pour le téléchargement

- error = download temporary disabled

un téléchargement pour ce même identifiant a été initialisé il y a moins de 60 minutes

- error = internal

erreur interne du système

9.1.4 Génération du fichier de données

Les informations récupérées pour chaque abonnement, et stockées dans le fichier de synchronisation d'abonnements, sont les suivantes :

mctId	Identifiant marchand.
subId	Identifiant unique d'abonnement
status	Statut de l'abonnement. Les valeurs possibles sont : - **active** : l'abonnement est ouvert et accessible pour l'utilisateur. - **suspended** : l'abonnement est provisoirement suspendu (pour cause de solde insuffisant par exemple). L'abonnement n'est plus accessible le temps de la suspension. - **terminated** : la reconduction a été arrêtée par action sur le CSR / CCE / Kit. L'abonnement reste accessible jusqu'à la fin de la période en cours. - **terminated_by_user** : la reconduction a été arrêtée par action de l'utilisateur dans son espace client. L'abonnement reste accessible jusqu'à la fin de la période en cours. - **closed** : l'abonnement est clôturé (plus d'accès possible pour l'utilisateur). (les différents cas possibles sont : résiliation immédiate via le CSR / CCE, fin de période atteinte après une action d'arrêt des reconductions, fin de période atteinte pour un produit à l'accès, clôture d'un dossier utilisateur, restriction de services appliquée sur un utilisateur, clôture d'un abonnement après X tentatives de reconduction en échec).

subscription_date	Date de souscription de l'abonnement. Cette information est toujours retournée.
last_renewal_date	Date de dernière reconduction réussie. Si cet abonnement n'a pas (encore) été reconduit, cette information ne sera pas retournée.
next_renewal_date	Date de la prochaine tentative de reconduction (prochaine date anniversaire ou éventuellement date de prochaine tentative de reconduction, si des premiers essais ont échoué, par exemple pour des utilisateurs prépayés ayant un solde insuffisant). Information retournée uniquement si l'abonnement est en statut active ou suspended.
closing_date	Date de fermeture effective de l'abonnement (date où l'utilisateur n'a ou n'aura plus accès à l'abonnement). Information présente uniquement si l'abonnement est en statut terminated ou terminated_by_user ou closed.
alias	Alias éventuel de l'utilisateur. Cette information est retournée uniquement si le marchand associé à cet abonnement est configuré pour de l'aliasing. Valeur "0" retournée dans le cas contraire.
productId	Identifiant de l'offre souscrite par le client

Toutes ces informations sont écrites dans le fichier avec le séparateur ";".
Les dates sont au format : "yyyy/MM/dd HH:mm:ss"

Exemple :

mctId	subId	status	subscription_date	last_renewal_date	next_renewal_date	closing_date	alias	productId
506	16381455	closed	10/04/2009 14:41	10/11/2011 14:52		10/12/2011 14:41	341 021 502 696	A4
506	22576184	closed	25/11/2011 17:37	16/12/2011 17:47		19/12/2011 10:50	391 213 397 398	ABO1
506	23068194	closed	13/01/2012 09:13			13/01/2012 09:21	391 213 397 398	ABO1

Dès la fin de la génération, le fichier est stocké sur les plates-formes w-HA. Il est dès lors disponible au téléchargement, uniquement pour le compte qui en a fait la demande, et pendant une durée limitée.
Le nom du fichier est un fichier .csv et compressé au format .zip, de la forme :

WHA_dataSubscriptions_[identifiant unique de demande].zip

9.2 APIs supplémentaires pour la gestion de l'abonnement tacitement reconductible « isSubscriptionOpen » et « closeSubscription »

9.2.1 Appel via le KIT

L'application « KIT V4 » contient les fonctionnalités nécessaires pour :

- Vérifier si l'abonnement d'un utilisateur (sId) est ouvert ou fermé
  [méthode java : subscriptionProxy.isSubscriptionOpen(sId)]

- Arrêter la reconduction d'un abonnement (pId)
  [méthode java : subscriptionProxy.closeSubscription(pId)]

Cela est possible en utilisant des classes et méthodes incluses dans les librairies du KIT V4.

Exemple d'utilisation des APIs pour la gestion de l'abonnement tacitement reconductible

Un exemple de code source implémentant ces deux méthodes est donné ci-dessous :

Exemple de code source :

Fichier « **testproxy.java** » :

```
import com.ipin.message.KeyTable;
import com.ipin.message.Partner;
import com.ipin.subscription.api.server.client.SubscriptionException;
import com.ipin.subscription.api.server.client.impl.SubscriptionProxy;

import java.rmi.RemoteException;

public class TestProxy {


   public static void main(String[] args) {

//Instantiate a keytable for handling keys
```

```
        KeyTable keyTable = new KeyTable();
//Add a specific key in the key table
        keyTable.addKey(506, 506, "0123456789abcdef0123456789abcdef");

//Instantiate a partner with the wHA merchant identifier
        Partner partner = new Partner(506, 506, "https://orange.w-ha.com/app-node-sub/responder");

//Instantiate a SubscriptionProxy for sending request to the NODE
        SubscriptionProxy subscriptionProxy;
        subscriptionProxy = new SubscriptionProxy(partner, keyTable);
        try {

//Sending a request to the node in order to check if a given subscription is open or not
/ based on the subscriptionId (10 in this case) returned when the sub is open
            subscriptionProxy.isSubscriptionOpen(10);

//to be used when a merchant wants to discontinue a
//subscription, all subscription with this productId (A1 in this case) will be marked as
//non-renewable
            subscriptionProxy.closeSubscription("A1");


        } catch (SubscriptionException e) {
            e.printStackTrace();  //To change body of catch statement use Options | File Templates.
        } catch (RemoteException e) {
            e.printStackTrace();  //To change body of catch statement use Options | File Templates.
        } catch (NumberFormatException e) {
            e.printStackTrace();  //To change body of catch statement use Options | File Templates.
        }

    }

}
```

Exemple de fichier de compilation du fichier testproxy.java

compile.bat

```
set CLASSPATH=D:\....\webapps\Kit_V4\WEB-INF\lib\valista_v1.0.jar
set CLASSPATH=%CLASSPATH%;D:\....\webapps\ Kit_V4\WEB-INF\lib\wha_v1.02.jar
set SOURCE=D:\....\webapps\Kit_V4\testproxy.java
```

D:\jdk1.5.0_01\bin\javac -classpath %CLASSPATH% -g -verbose %SOURCE%

Exemple de fichier d'exécution du programme TestProxy

execute.bat

```
set CLASSPATH=D:\....\webapps\ Kit_V4\WEB-INF\lib\ valista_v1.0.jar
set CLASSPATH=%CLASSPATH%;D:\....\webapps\ Kit_V4\WEB-INF\lib\ wha_v1.02.jar
set CLASSPATH=%CLASSPATH%;D:\....\webapps\ Kit_V4\

D:\jdk1.5.0_01\bin\java -cp %CLASSPATH% TestProxy
```

9.2.2 Appel sans le KIT

Ces méthodes peuvent être appelées sans l'aide du KIT. Il suffit pour cela de construire la requête à envoyer à la plate-forme w-HA

Exemple d'une URL à fabriquer :

isSubscriptionOpen
https://orange.w-ha.com/app-node-sub/responder?m=h=d694596371784c17ea3704ac6c77ec4ff38354aab1ec6622a5e8ecd981fc987a;p=12268;k=12268;v=2:{c=SubStatusReq;v={sId=12558425; }}

closeSubscription
https://orange.w-ha.com /app-node-sub/responder?m=h=d694596371784c17ea3704ac6c77ec4ff38354aab1ec6622a5e8ecd981fc987a;p=12268;k=12268;v=2:{c=CloseSubReq;v={pId=product_1;}}

Détail des paramètres :

- h=hmac calculé à partir de la derrière partie de l'url (ici : c=SubStatusReq;v={sId=12558425;}) et de la KeyValue
- p=mctid
- k=keyid=mctid
- v=2 (ne pas changer)
- c=**SubStatusReq** ou **CloseSubReq** (type de requête)
- sId=numéro de l'abonnement à contrôler (pour isSubscriptionOpen)
- pId = identifiant de l'offre à fermer (pour closeSubscription)

Le hmac est utilisé pour sécuriser les requêtes entre vos serveurs et ceux de w-HA.

Voici un exemple d'une fonction en php qui réalise des hmac à partir d'une clef et d'une chaîne de caractères.

```
function hmac ($string,$key)
{

   $contenu_signature = "";
   $contenu_signature = $string;


   //Encodage base64 de la chaine chiffrée avec l'algorithme HMAC-SHA-256
   $signature = base64_encode(hash_hmac('sha256',$contenu_signature, $key, true));
   return $signature;
}


 $hmac = hmac("c=SubStatusReq;v={sId=12558425;}","Votre Keyvalue");
```

Dans la variable $hmac, il y aura donc d694596371784c17ea3704ac6c77ec4ff38354aab1ec6622a5e8ecd981fc987a (après avoir mis la bonne KeyValue).

Ensuite, dans le retour que fait w-HA à cette requête, il y a 3 réponses possibles :
-L'abonnement est encore actif :
exemple : h=
d694596371784c17ea3704ac6c77ec4ff38354aab1ec6622a5e8ecd981fc987b;p=506;k=506;v=2:{c=ack;v={s=true;}}

-L'abonnement est inactif (résilié ou terminé) :
exemple: h=
d694596371784c17ea3704ac6c77ec4ff38354aab1ec6622a5e8ecd981fc987c;p=506;k=506;v=2:{c=ack;v={s=false;}}

-Erreur dans la requête (mauvais hmac par exemple) :
e=3

De la même manière, dans les deux premiers exemples, le h correspond à un hmac calculé à partir de votre KeyValue par la Plate-forme w-HA. Vous pouvez donc vérifier l'intégrité du message reçu en recalculant le hmac de la même manière que précédemment, avec comme chaînes de caractères :
c=ack;v={s=true;}
ou
c=ack;v={s=false;}

Le hmac calculé doit être identique à celui de la réponse, sinon cette dernière n'est pas intègre.

9.3 Récupération des informations d'un abonnement multimédia « consultSubscription »

9.3.1 Appel via le KIT

La méthode consultSubscription permet de récupérer toutes les caractéristiques d'un abonnement multimédia souscrit via le Kit V4.

Cette méthode permet de récupérer des informations détaillées sur un abonnement. L'abonnement doit s'appliquer sur un marchand configuré dans le kit (avec couple identifiant marchand / clé marchand présent dans le fichier xml de configuration marchands).

Cette API va utiliser les informations contenues dans le fichier marchands.xml et générer une requête qui sera envoyée à la plate-forme w-HA

Les informations retournées seront les suivantes :

status	Statut de l'abonnement. Cette information est toujours retournée. Les valeurs possibles sont les suivantes : - **active** : l'abonnement est ouvert et accessible pour l'utilisateur. - **suspended** : l'abonnement est provisoirement suspendu (pour cause de solde insuffisant par exemple). L'abonnement n'est plus accessible le temps de la suspension. - **terminated_by_user** : la reconduction de l'abonnement est arrêtée par action de l'utilisateur dans son espace client. L'abonnement reste accessible jusqu'à la date de clôture. - **terminated** : la reconduction de l'abonnement est arrêtée par action du Service client Orange ou Editeur via le Customer Care (dans le MSCA) ou via le Kit V4. L'abonnement reste accessible jusqu'à la date de clôture.

	- **closed** : l'abonnement a été clôturé de façon immédiate (par action du Service Client Orange, pour solde insuffisant suite aux tentatives de reconduction en échec, pour fermeture du compte client).
subscription_date	Date de souscription de l'abonnement. Cette information est toujours retournée.
last_renewal_date	Date de dernière reconduction réussie. Si cet abonnement n'a pas (encore) été reconduit, cette information ne sera pas retournée.
next_renewal_date	Date de la prochaine tentative de reconduction (prochaine date anniversaire ou éventuellement date de prochaine tentative de reconduction, si des premiers essais ont échoué, par exemple pour des utilisateurs prépayés ayant un solde insuffisant). Information retournée uniquement si l'abonnement est en statut **active** ou **suspended**.
closing_date	Date de fermeture effective de l'abonnement. Information présente uniquement si l'abonnement est en statut **terminated, terminated_by_user** ou **closed**
alias	Alias éventuel de l'utilisateur. Cette information est retournée uniquement si le marchand associé à cet abonnement est configuré pour de l'aliasing. Valeur "0" retournée dans le cas contraire.
productId	Identifiant de l'offre souscrite par le client

En cas d'erreur, les différentes exceptions retournées au kit sont :

- SUBSCRIPTION_NOT_FOUND (identifiant d'abonnement non trouvé par w-HA)
- MERCHANT_NOT_TRUSTED (l'identifiant marchand n'est pas associé à l'abonnement)
- INVALID_MESSAGE (erreur du respondeur w-HA, vérifier la clé marchand dans le kit)
- XML_NOT_FOUND (identifiant marchand introuvable dans le fichier marchands du kit)

Un exemple d'implémentation de cette API est disponible sur la page consultSubscription.jsp livrée avec le kit.

9.3.2 Appel sans le KIT

Cette méthode peut être appelée sans l'aide du KIT. Il suffit pour cela de construire la requête à envoyer à la plate-forme w-HA

Exemple d'une URL à fabriquer :

https://orange.w-ha.com/app-node-sub/responder?m=h=d694596371784c17ea3704ac6c77ec4ff38354aab1ec6622a5e8ecd981fc987a;p=506;k=506;v=2;{c=ConsultSubReq;v={sId=20798975;}}

Détail des paramètres :

- h=hmac calculé à partir de la dernière partie de l'url (ici : c=ConsultSubReq;v={sId=20798975;}) et de la KeyValue
- p=mctid
- k=keyid=mctid
- v=2 (ne pas changer)
- c=ConsultSubReq (type de requête : ne pas changer)
- sId=numéro de l'abonnement

Le hmac est utilisé pour sécuriser les requêtes entre vos serveurs et ceux de w-HA.
Voici un exemple d'une fonction en PHP qui réalise des hmac à partir d'une clef et d'une chaîne de caractères.

```
function hmac ($string,$key)
{

   $contenu_signature = "";
   $contenu_signature = $string;


   //Encodage base64 de la chaine chiffrée avec l'algorithme HMAC-SHA-256
   $signature = base64_encode(hash_hmac('sha256',$contenu_signature, $key, true));
   return $signature;
}

 $hmac = hmac("c=SubStatusReq;v={sId=20798975;}","Votre Keyvalue");
```

Dans la variable $hmac, il y aura
donc d694596371784c17ea3704ac6c77ec4ff38354aab1ec6622a5e8ecd981fc987a (après avoir mis la bonne KeyValue).

Ensuite, dans le retour que fait w-HA à cette requête, il y a plusieurs réponses possibles :
- Erreur dans la requête (mauvais hmac par exemple) :
e=3

- La requête est correcte

Exemple d'un abonnement résilié :

h=d694596371784c17ea3704ac6c77ec4ff38354aab1ec6622a5e8ecd981fc987a;p=506;k=506;v=2:{c=ack;v={status=closed;alias=391213397398;closing_date=2011-12-19 10:50:33;productId=ABO1;subscription_date=2011-11-25 17:37:49;last_renewal_date=2011-12-16 17:47:10;}}

Exemple d'un abonnement qui ne correspond pas à la boutique :

h=d694596371784c17ea3704ac6c77ec4ff38354aab1ec6622a5e8ecd981fc987a;p=506;k=506;v=2:{c=ex;v={m=MERCHANT_NOT_TRUSTED;t=com.ipin.subscription.api.server.client.SubscriptionException;c=0;}}

De la même manière, dans l'exemple, le h correspond à un hmac calculé à partir de votre KeyValue par la Plate-forme w-HA.
Vous pouvez donc vérifier l'intégrité du message reçu en recalculant le hmac de la même manière que précédemment, avec la chaîne de caractères suivante :

c=ack;v={status=closed;alias=391213397398;closing_date=2011-12-19 10:50:33;productId=ABO1;subscription_date=2011-11-25 17:37:49;last_renewal_date=2011-12-16 17:47:10;}

ou

c=ex;v={m=MERCHANT_NOT_TRUSTED;t=com.ipin.subscription.api.server.client.SubscriptionException;c=0;}
Le hmac calculé doit être identique à celui de la réponse, sinon cette dernière n'est pas intègre.

9.4 API subscriptionTransactions (récupérer les transactions d'un abonnement)

Cette API permet de récupérer les caractéristiques principales de l'abonnements, mais également le détail des transactions associées.
Par défaut, les transactions des 12 derniers mois, au maximum, seront retournées.

L'abonnement doit être lié à un marchand configuré dans le kit (avec couple identifiant marchand / clé marchand présent dans le fichier xml de configuration marchands).

Il est possibile de spécifier un paramètre facultatif pour limiter l'historique des transactions retournées.

Cette API est disponible dans le kit V4 uniquement (>= 4.04).

com.wha.api.SubscriptionManager	
List	**subcriptionTransactions**(long subscriptionId, int history, long merchantId)

Description des paramètres envoyés par le kit :

m = message :

- h = HMAC SHA-256 généré selon paramètres et clé marchand
- p = identifiant du marchand
- k = identifiant de la clé marchand
- v = version du kit (4 uniquement pour cette API)
- c = type de message / action : *SubTrxReq*
- sId = identifiant d'abonnement
- history = nombre de transactions maximum retournées (limité à 12 mois max d'ancienneté)
- t = token à usage unique, généré automatiquement par le kit V4

Le paramètre *history* peut avoir une valeur nulle (dans ce cas, récupération par défaut des 12 derniers mois)

Exemples de requêtes envoyées à W-HA (générées par l'API) :

https://orange.w-ha.com/app-node-sub/responder?m=h=25b0416fbdfcf23dc5e6e26003973b3daa2a4820daa90c6fa345018e6fd9bac8;p=502;k=502;v=4:{c=SubTrxReq;v={sId=8766026;t=57dd5e19bbc245609823c5338b1a7dad;history=12;}}

https://orange.w-ha.com/app-node-sub/responder?m=h=4d8c83ff86b54d9425508917dc5d41ce884df0e1f30761776c2f2c25f0335da1;p=502;k=502;v=4:{c=SubTrxReq;v={sId=8766026;t=6ce31b35cd3e4ee2be02b2fd65a5fbfe;}}

Réponse de l'API

La réponse est fournie sous forme de Liste Java, contenant les données suivantes, en 2 parties.

9.4.1 Détail de l'abonnement

Le premier élément de la liste retourne une Map, contenant les informations sur l'abonnement :

type	Type de donnée. Toujours égal à « **subscriptionDetail** »
status	Statut de l'abonnement. - **active** : l'abonnement est ouvert et accessible pour l'utilisateur. - **suspended** : l'abonnement est provisoirement suspendu (pour cause de solde insuffisant par exemple). L'abonnement n'est plus accessible le temps de la suspension. - **terminated_by_user** : la reconduction de l'abonnement est arrêtée par action de l'utilisateur dans son espace client. L'abonnement reste accessible jusqu'à la date de clôture. - **terminated** : la reconduction de l'abonnement est arrêtée par action sur le CSR / CCE / Kit marchand / Kit NetAuth. L'abonnement reste accessible jusqu'à la date de clôture. - **closed** : l'abonnement a été clôturé de façon immédiate (par action sur le CSR, pour solde insuffisant suite à X tentatives de reconduction en échec, pour fermeture du compte client).
subscription_date	Date de souscription de l'abonnement, au format YYYY-MM-DD HH24:MI:SS
next_renewal_date	Date de la prochaine tentative de reconduction (prochaine date anniversaire ou éventuellement date de prochaine tentative de reconduction, si des premiers essais ont échoué). Information retournée uniquement si l'abonnement est en statut *active* ou *suspended*. Au format YYYY-MM-DD HH24:MI:SS

9.4.2 Détail des transactions

Tous les autres éléments de la liste retournent une Map représentant chacune une transaction. Il y aura donc autant d'éléments dans la liste que de transactions associées à l'abonnement, sur les 12 derniers mois. (+ l'élément initial de type « subscriptionDetail »).

La liste des transactions est retournée dans l'ordre antéchronologique.

type	Type de donnée. Toujours égal à « **transaction** »
trx_id	Identifiant externe de la transaction
amount	Montant TTC de la transaction
trx_date	Date de la transaction correspondant à la date de l'achat par le client (lorsque le statut transaction est passé en statut 1). Au format YYYY-MM-DD HH24:MI:SS
is_refunded	Indicateur si la transaction a déjà été remboursée intégralement. 0 : transaction non remboursée 1 : transaction déjà remboursée 2 : transaction remboursée partiellement

La récupération des identifiants de transaction permet, par exemple, de lancer ensuite des demandes de remboursement (par l'action Editeurs Refund, qui demande notamment l'identifiant de transaction en paramètre).

En cas d'erreur, une exception Java sera retournée par l'API, parmi les suivantes :

- [SubscriptionException] SUBSCRIPTION_NOT_FOUND (identifiant d'abonnement inconnu)
- [SubscriptionException] MERCHANT_NOT_TRUSTED (l'identifiant marchand n'est pas associé à l'abonnement)
- [MessagingException] INVALID_MESSAGE 3 (erreur de connexion au responder W-HA)
- [MessagingException] INVALID_RESPONSE 10 (erreur ack non retourné par responder W-HA)
- [KeyDataException] XML_NOT_FOUND 2 (identifiant marchand introuvable dans le fichier marchands du kit)

Un exemple d'implémentation de cette API est disponible sur la page subscriptionTransactions.jsp, livrée avec le kit.

9.5 Vérification de la cohérence base centrale - base locale

9.5.1 Définition

L'API checkLocalAndRemote permet de vérifier la cohérence entre la base locale du kit marchand et la base centralisée (le catalogue produit).

La cohérence est vérifiée seulement lors du démarrage du kit et du chargement à chaud de la base produit et aussi si l'utilisateur invoque l'API de vérification.

Les problèmes de cohérence sont tracés dans les logs du moteur de servlet (par ex Tomcat) sous la forme : « Incohérence sur le marchand » + merchantId.

Ils sont également remontés au niveau du catalogue centralisé pour que l'utilisateur connecté puisse en prendre connaissance. Un message d'alerte via une pop-up apparaîtra à la prochaine connexion au catalogue centralisé.

9.5.2 Détection de l'incohérence au démarrage

En cas de détection d'incohérence un message d'erreur est affiché dans les logs mais le kit démarre et permet d'effectuer des achats, deux cas sont alors possibles :

Achat impossible

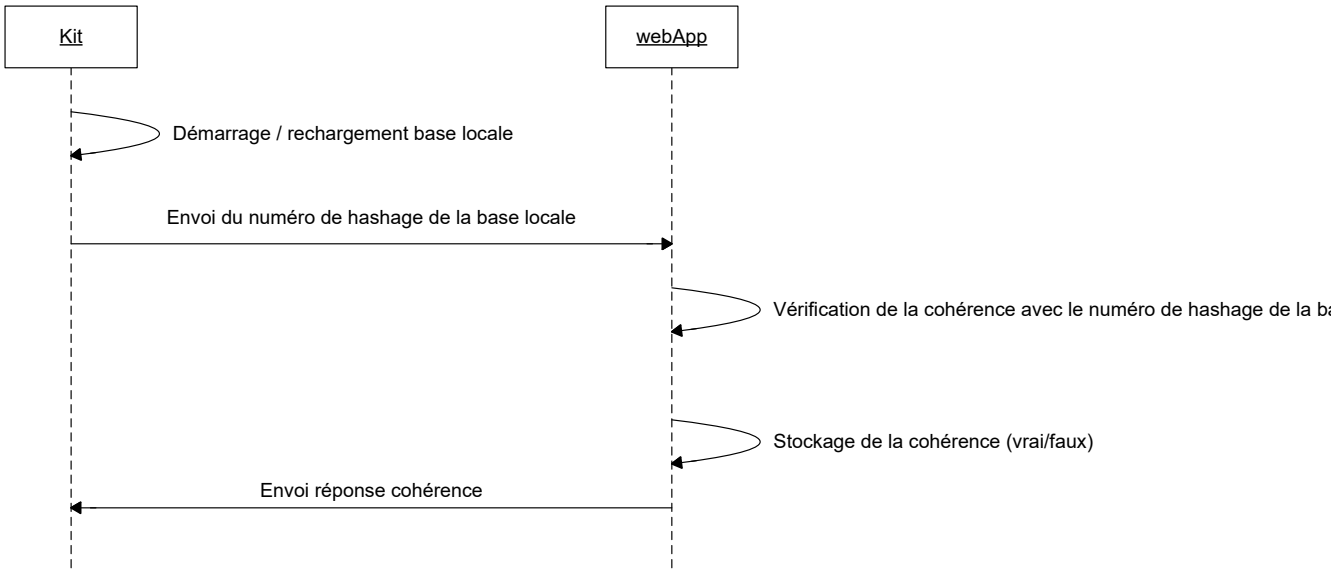
- produit absent de la base locale
- produit absent de la base centralisée

Achat possible

- produit avec incohérences (valeurs différentes sur le purchaseCase, la fulfillmentUrl, ou l'autoConfirm entre la base locale et la base centralisée) : l'achat est possible, mais ce sont les données de la base locale qui seront prises en compte.

9.5.3 Cinématique

Pour vérifier la cohérence de la base locale avec la base centralisée, on compare un numéro de hachage qui est généré en fonction de la configuration de la base du kit avec celui stocké au niveau du nœud.



9.5.4 L'API de vérification

Il est possible de vérifier les marchands indépendamment du lancement de la servlet. On utilise dans ce cas une méthode qui permet de faire le test. On lui passe la liste des marchands à tester, et elle renvoie en retour la même liste de marchands avec leurs résultats respectifs.

La liste des marchands est composée de deux éléments pour chaque marchand : la clé est l'identifiant du marchand et la valeur est le flux (java.io.InputStream) ouvert sur le fichier de configuration des produits (productsCurrent_XXX.xml).

En retour, la clé est toujours l'identifiant du marchand et la valeur est un objet Booléen pour indiquer le résultat de la vérification de cohérence.

9.5.5 Exemple d'utilisation

Voici un exemple de JSP, pour vérifier la cohérence des bases.

Fichier inconsistencyCheck.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"%>
<%@ page import="java.util.Map, java.util.HashMap, com.wha.pcc.InconsistencyManager,
java.util.Iterator"%>

<html>

<head>
    <title>Inconsistency Check</title>
</head>

<%

Map merchantList = new HashMap();

/* define parameters here */
String nodeInconsistencyUrl = "https://orange.w-ha.com/app-node-pcc/inconsistency";
merchantList.put(new String("502"), getServletContext().getResourceAsStream("/WEB-
INF/productsCurrent_502.xml"));
merchantList.put(new String("507"), getServletContext().getResourceAsStream("/WEB-
INF/productsCurrent_507.xml"));

InconsistencyManager im = new InconsistencyManager();
Map resultMap = im.checkLocalAndRemote(merchantList, nodeInconsistencyUrl);
Iterator iterator = resultMap.entrySet().iterator();
%>

<body>
    <p>
            Inconsistency Check results :<br/>
            <%
                while (iterator.hasNext()) {
                    Map.Entry entry = (Map.Entry) iterator.next();
                    out.print("" + entry.getKey() + " : " + entry.getValue() + "<br>");
                }
            %>
    </p>
</body>
</html>
```

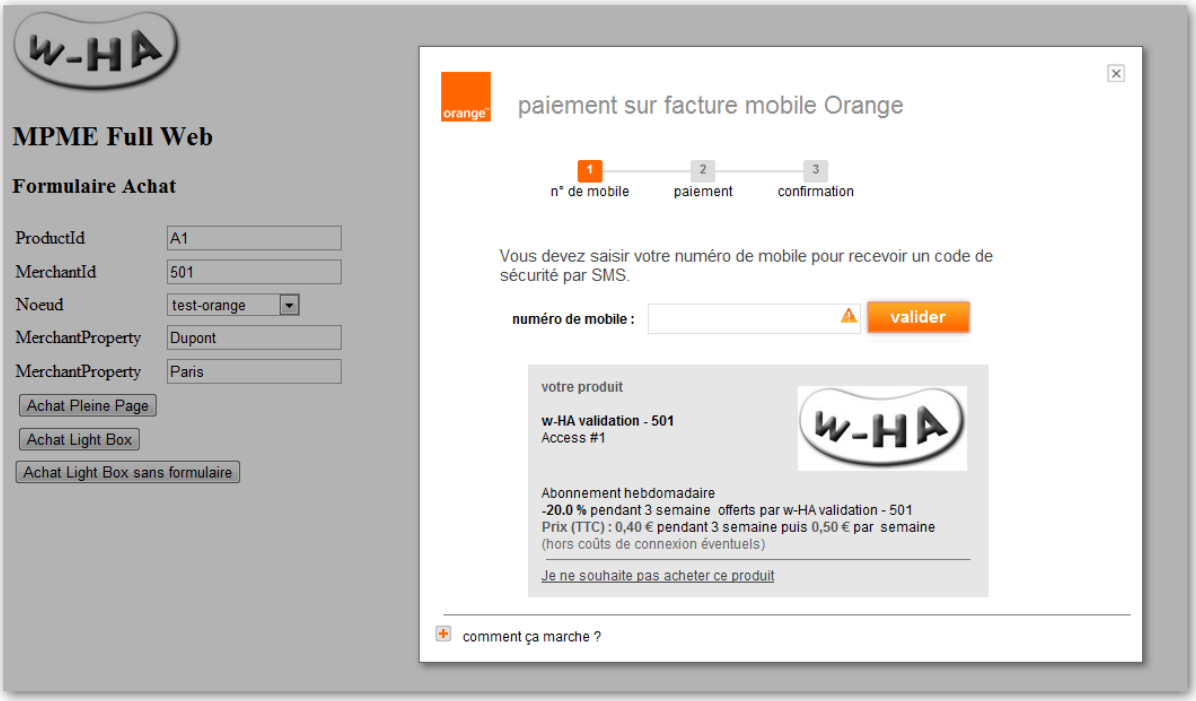
9.6 Intégration Light Box pour MPME Full Web

Par défaut l'appel du KIT v4 via un navigateur WEB renvoie le client vers le panneau MPME Full Web en pleine page sur la page courante.
On peut néanmoins intégrer le panneau Full Web en surimpression du site d'origine : **Light Box**.

Les scripts ainsi que la feuille de style nécessaire à l'utilisation d'une light box sont disponibles dans l'archive **achats_web_lightbox.rar**, disponible sur l'espace éditeur w-HA : http://www.w-ha.com/espace-editeurs/clients/index.php

Cette archive contient :

- Un répertoire « js », incluant les 2 scripts **oafmashup.js** et **lightBox.js,** la feuille de style **lightBox.css** et le fichier **proxy.html**
- Un répertoire « xhtml », incluant un exemple de fichier html **index_web.html**



Panneau Full Web en Light Box

9.6.1 Installation du script

Copier le dossier js contenant les scripts **oafmashup.js** et **lightBox.js**, la feuille de style **lightBox.css** et le fichier **proxy.html** dans le dossier de déploiement de votre serveur web (par exemple dans le dossier Kit_V4).

Référencer les scripts et feuilles de style CSS dans la page HTML ou JSP appelante.

Exemple de référencement des scripts et styles :

<script type="text/javascript" src="../js/**oafmashup.js**"></script>

```
<script type="text/javascript"      src="../js/lightBox.js"></script>
<link href="../js/lightBox.css" rel="stylesheet" type="text/css" />
```

9.6.2 Appel au kit

9.6.2.1 Via un formulaire

Paramétrer le formulaire d'appel au kit en précisant **son id et son action** - qui seront utilisés par le script pour ouvrir la light box - et le **paramètre « format »** à la valeur « **xhtml** ».

Exemple de formulaire pour acheter le produit P1 :

```
<form id="formP1" action="/Kit_V4/pos-bundle">
    <input type="hidden" name="pid" value="P1"/>
    <input type="hidden" name="format" value="xhtml"/>
    <input type="hidden" name="action" value="purchaseListOffer"/>
    <input type="hidden" name="mid" value="_MCTID_"/>
    <input type="submit" value="Product # P1 javascript"/>
</form>
```

9.6.2.2 Via son URL

Le kit peut également être directement en construisant l'URL. Un script Light Box est disponible si vous êtes dans cette configuration.
Dans tous les cas, en Full Web, il est nécessaire de préciser le format « xhtml »
Par exemple :

http://monsite.com/Kit_V4/pos-bundle?action=purchaseListOffer&**format=xhtml**&pid=P1

9.6.3 Utilisation de la light box

9.6.3.1 A partir d'un formulaire

Paramétrer le bouton d'appel à la light box en utilisant l'ID et l'action du formulaire précédent.
Le script Light Box prenant en charge les formulaires se nomme : « openLightBoxForm »
Il prend en paramètre le nom du formulaire et l'URL du kit (sans paramètres)

Exemple d'appel à une light box via un formulaire:

```
<div id="pilot">
<input type="button" value="Achat P1" onClick="openLightBoxForm('formP1','/Kit_V4/pos-bundle');"/>
</div>
```

9.6.3.2 A partir d'une URL

Paramétrer le bouton d'appel à la light box en utilisant l'URL d'appel du kit.
Le script Light Box prenant en charge l'URL se nomme : « openLightBox »
Il prend en paramètre l'URL d'appel du kit.

Exemple d'appel à une light box via l'URL :

```
<div id="pilot">
<input type="button" value="Achat P1" onClick=" openLightBox('/Kit_V4/pos-bundle?action=purchaseListOffer&format=xhtml&pid=P1');"/>
</div>
```

9.6.4 Code HTML

Insérer deux blocs au sein du <body> du code html, le premier contenant le corps de la page html , le second pour afficher le fond gris transparent de la light box lors de son utilisation.

Exemple de code html incluant une light box :

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.04 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html>
<head>
<script type="text/javascript"  src="../js/oafmashup.js"></script>
<script type="text/javascript"  src="../js/lightBox.js"></script>
<link href="../js/lightBox.css" rel="stylesheet" type="text/css" />
```

```
</head>

<body>

<div class="content">
<!-- Contenu de la page html -->
<form id="formP1" action="/Kit_V4/pos-bundle">
<!—Formulaire d’appel au kit pour le produit P1 -->
</form>

<div id="pilot">
<input type="button" value="Achat P1" onClick="openLightBoxForm('formP1','/Kit_V4/pos-
bundle');"/>
<!—Appel à la light box-->
</div>
</div>

<div id="TransparentLayerId" class="bgLightBox">
<!-- Ce bloc reste vide et doit être inséré juste au dessus de la fin de balise <body> -->
</div>

</body>
</html>
```