 	**Reference :** KIT-V3.5	**Version :** 2.17	**Page :** 1/63
	Recipients : Content Providers	**Date :** 02/09/2009	
	Author : Fabien MOREAU	**Revised**	**Visa**
	Subject : KIT v3.5		

Document Type : General technical description
Document Title : Internet+ Technical description KITv3.5 subscription w-HA

Version	Date	Author	Action	Reasons
2.09	15/05/2009	AD	Update	English update of the French version
2.10	29/05/2009	AD	Delete	Delete Test responder and pull access control
2.11	03/07/2009	AD	Correction	MSCA Access + account creation procedure
2.12	11/08/2009	AD	Add	Server to server refund / unsbscription
2.13	18/08/2009	AD	Homogenization	homogenization doc signle one-off payment & subscription
2.14	28/08/2009	AD	Add	url set in the kit (§ ANNEXE III)
2.15	02/09/2009	AD	Correction	Call of urlRedirect (§ 3.2.4.5.)
2.16	19/10/2009	RC	Correction	Update of w-HA node + refund / unsubscription
2.17	22/03/2010	AD	Modification	termination notification schema
2.18	08/07/2010	AT	Add	Effective notification details (§ 3.2.4.1)

Table of contents

Introduction

In order to meet market demand, the Internet+ solution (www.internetplus.fr) has been developed in order to manage not only **Event Based** payment, but also **recurrent subscriptions**.

Some modifications to the w-HA platform accompany with this development: implementing a new application (v3.5), making a new Web interface available to the Content Providers allowing them to create offers and products (**MSCA** – Merchant Self Care Application).

The Content Provider's **product base**, i.e. all of the information (identifier, price, description, purchasing terms, ...) related to these offers and products is therefore present on the w-HA platform, and no longer on the w-HA application (Kit v3.5) installed on his (their) server(s).

There is still a **product catalogue**, limited to one identifier and one delivery URL (ffUrl), on Kit v3.5, to enable linking with the product base.

The new w-Ha application (Kit 3.5) installed on the Content Provider server(s) has a certain number of advantages such as:

- **multi-type payments: Event Based sales and subscriptions**
- **multi-merchants/shops: one single Kit for several merchants or shops**

Document aim

The aim of this document is to describe how the Kit v3.5 works in relation to the w-HA platform and in the Internet+ offer to manage Event Based sale purchases and subscriptions.

It is mainly aimed at the w-HA installer on the service Content Provider platform.

Another document called "Reference Manual for Subscription Packages (MSCA)" is aimed at shop manager(s).

1 – Subscription's general concepts

1.1 - Vocabulary

The aim of this paragraph is to define "protagonists" involved in a transaction with w-HA, and to explain different flows between protagonists.

w-HA :
the word "w-HA" can point, depending on the context:
- w-HA company
- w-HA platform

Content provider:
The « Merchant » (or content provider) is the seller of service accessible on the net.

The ISP (Internet Servcie Provider) of the client:
The ISP has commercial and financial connection with the internet user and has a partnership with w-HA.

Internet user:
The internet user is the final client who subscribes, on the net, to a subscription using w-HA payment solution.

Offer (oid):
This is the **subscription model**, defining the subscription **period and a rate** for an access to a product.

Subscription (uoid):
This is an aspect of the offer, connected to a user.

Transaction (trxid):
A subscription creates a transaction. Another transaction is created for each renewal. Transactions are billed to the internet user.

To sum-up:
An internet user **subscribes to an offer** (model) so, he has a valid subscription.
Then, he can access the product or service during the subscription validity period.

1.2 – Principle scheme (commercial and marketing aspect)

1.2.1 – Subscription process with the client point of view

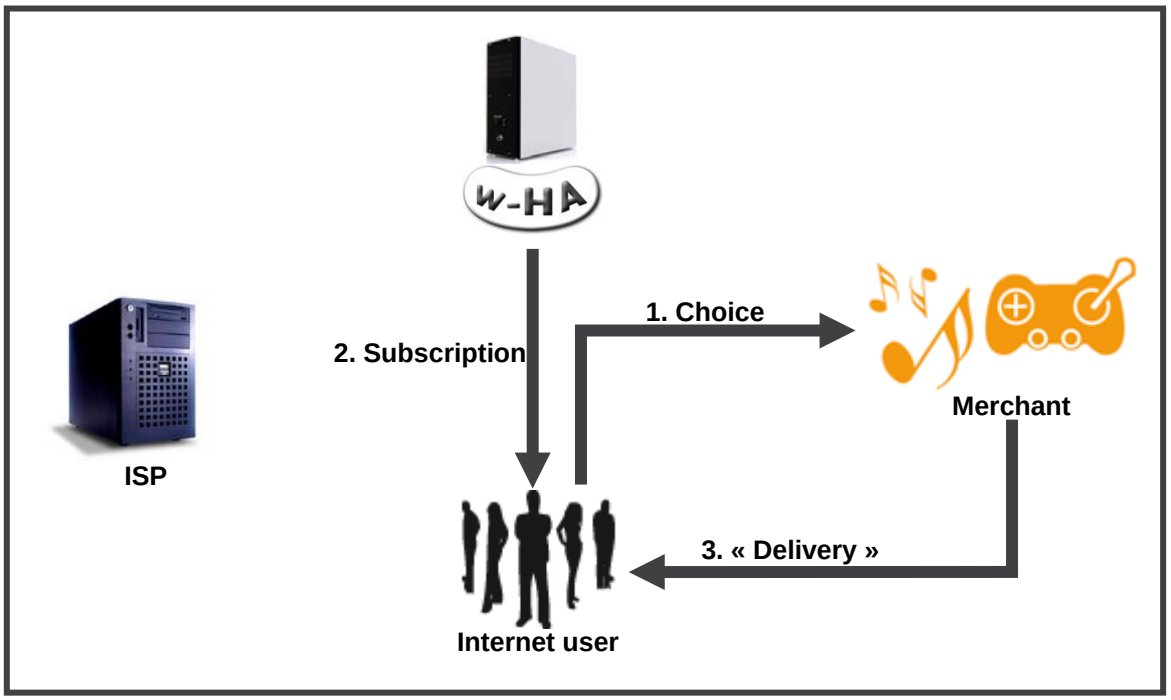


fig 1 : subscription process (internet user point of view)

The internet user will see pages bellow:

1. « merchand » screen: The internet user chooses a subscription on the merchant web site.

2. « w-HA » screen: w-HA displays the billing panel on which the internet user will confirm his subscription.

3. « merchand » screen: After w-HA authorization the internet user is redirected to the merchant web page to access to the service he subscribed.

Warning :
Only web page seen by the internet user are presented here. In reality, there are different infromation flows between the protagonists. Detail of thoses flows are given in §1.3.

1.2.2 – Financial flows (deferred)

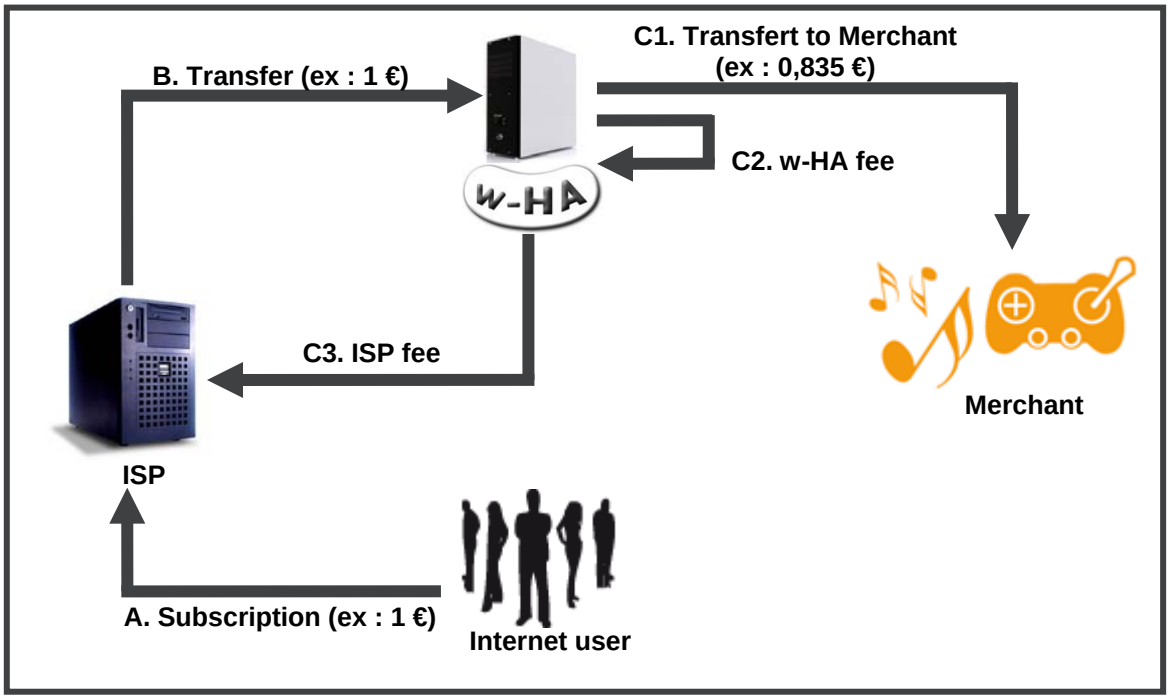


fig 2 : financials flows between all protagonists is the transaction

Financials flows between all protagonists are the transactions (deferred) are:

Example for a 1€ subscription:

A	The internet user is billed by its ISP
B	The ISP transfer 100% off the payment to w-HA
C1	w-HA transfers a part of the payment to the Merchant (ex : 0,835€)
C2	w-HA conserves a part of the payment
C3	w-HA transfers a part of the payment to the ISP

1.3 – Subscription kinematic with w-HA payment system (technical aspects)

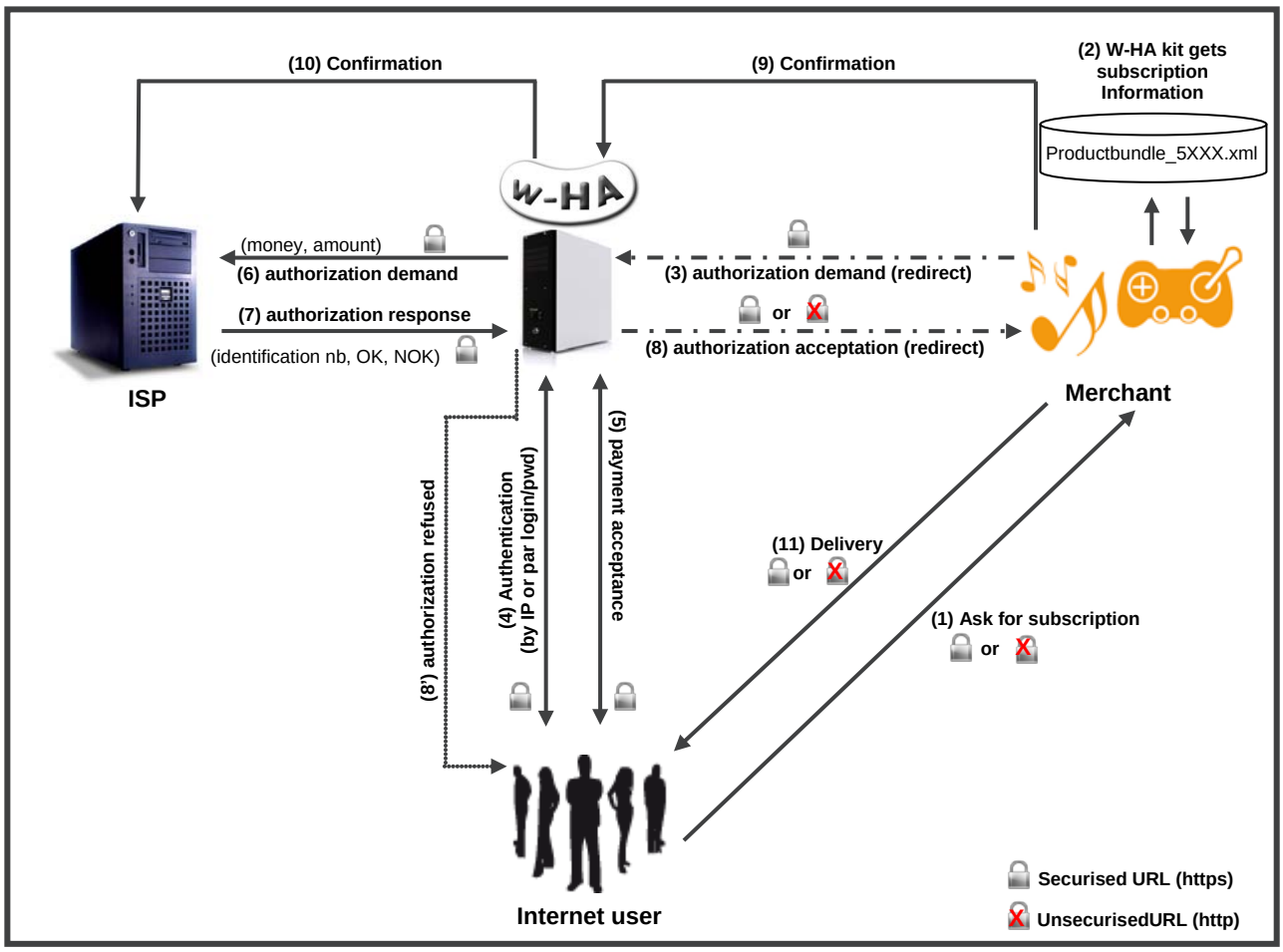


fig 3 : details of information flows all protagonists is the transaction

1.4 – Internet user Registration, Authentication and Payment

1.4.1 - Inscription

w-HA ISP partners

In general, the internet user **registration** of the w-HA payment system at its ISP is **automatic** (optin) when he creates his internet access. The internet user can disable this option later if he wants to. He can also accept or not “nomad payment” (see § bellow).

Others ISP

If the internet user subscribed to internet to an **ISP, which is not w-HA partner**, or if he wishes to pay with an other payment system, he can open an **Orbeo** account with his **bankcard** on http://www.w-ha.com.

1.4.2 - Authentication

Internet connexion with ISP partner
If the internet user uses an **ISP partner** connexion when he subscribes, he is **automatically identified by his IP address** and he is immediately redirected to the « billing panel » (designed by his ISP).

Internet connexion with an other ISP
If the internet user use an **ISP, which is not internet+ partner** (or a company internet access) when he subscribes, he is redirected to the « panel of ISP choice ».

If he accepted « **nomad payment** », he can **choose his ISP and authenticate him self (login/pwd)** then, he will be redirected to the « **billing panel** ».

He can also decide to subscribe with his **Orbeo account**, if the merchant allow this type of payment. The internet user has to **authenticate him self (login/pwd)** when he opens his Orbeo account.

1.4.3 - Payment

When the internet user is authenticated (automatically by his IP address or by login/pwd), he is redirected to the billing panel.

This billing panel is designed by his ISP and display information bellow:
- merchant name
- merchant logo
- subscription description
- subscription period
- subscription amount

The internet user has to **clic just one more time to subscribe** (confirmation button). Thanks to its system, w-HA perform **one of the best transformation rates of the market!**

1.5 – The 3 subscription's stages

1.5.1 – The client subscribes (Beginning of the Subscription)

The event that initiates a subscription is: An internet user **subscribes to an offer.**
The client, in exchange of the payment of his subscription, gets an access to the offer on a defined period.

1.2.2 - Access controls' (Subscription's life)

Once the client has his subscription, the content provider must check for each access if the subscription is valid or not.

The **access control** must to be **done by the content provider**. (w-HA does not do it)

1.2.3 – Termination (End of the subscription)

The subscription terminates when one of the events below happens:
- End of the period's subscription (if the subscription is not tacitly renewable)
- The client asks for the termination of his subscription
- The client asks to be refund

2 – Content provider's access control

The content provider needs a database to store subscriptions data (subscriptionID, subscription date, status…).

w-HA has its own database. When a client asks for the termination of his subscription w-HA send « termination notification » to the content provider. Thanks to this process, databases are synchronized.

In case of database desynchronization, the content provider has to synchronization it database to w-HA database.

2.1 - The client subscribes (Beginning of the Subscription) / Content provider's access control

For the Internet+ commercial offer, user authentication will be carried out from the service Content Provider website.

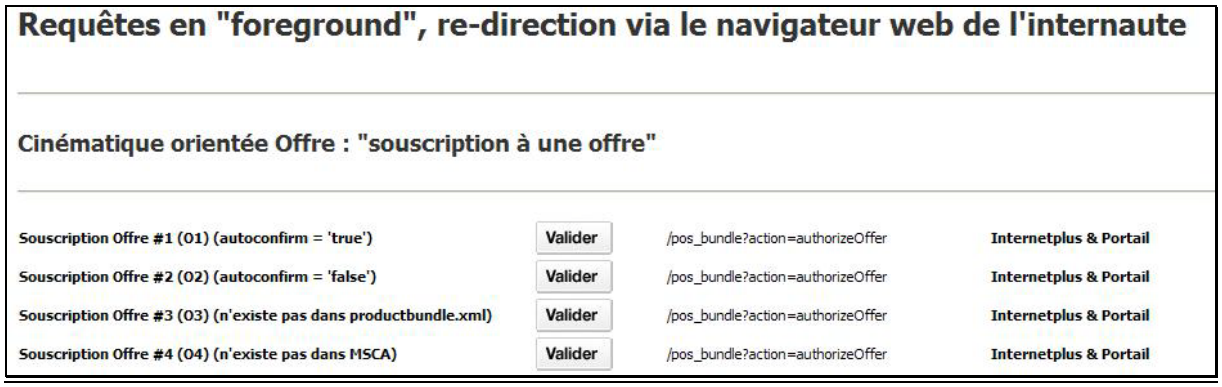
The user chooses a login and a password and the Content Provider application connects its own username, for example UserId:

The content provider must associate its own userID to the w-HA subscriptionID (uoid).

Example:
Login: fabien.moreau@orange-ftgroup.com
Password: ********
} userID = fmoreau1234

There are 4 steps to get a subscription:

2.1.1 - Step #1 : Authorization request for a subscription using the servlet : /pos_bundle?action=authorizeOffer



Warning!

A client can subscribe to the same offer several times.
The content provider must check if this client has already subscribe to this offer and not let him subscribe again if it is so.

The client is re-directed toward a subscription pannel on the w-HA platform.

Example:

Warning!

If the client clicks on the « Confirm your purchase » button and if there is no error (ex : exceeding the payment ceiling), the state of the User Offer (uoid) is **authorized** in w-HA database. At this step, this client **will not be charged** (**waiting for the "confirm" status** at step #2)

2.1.2 - Step #2 : 1st CP database update

Once the w-HA platform has made the subscription authorization request, the CP can **update its database** with the **subscription's identification**, providing for instance:

*** its status :**

- Authorized (button « Confirm your purchase »)
- Cancelled (link « cancel »)

*** its end date**

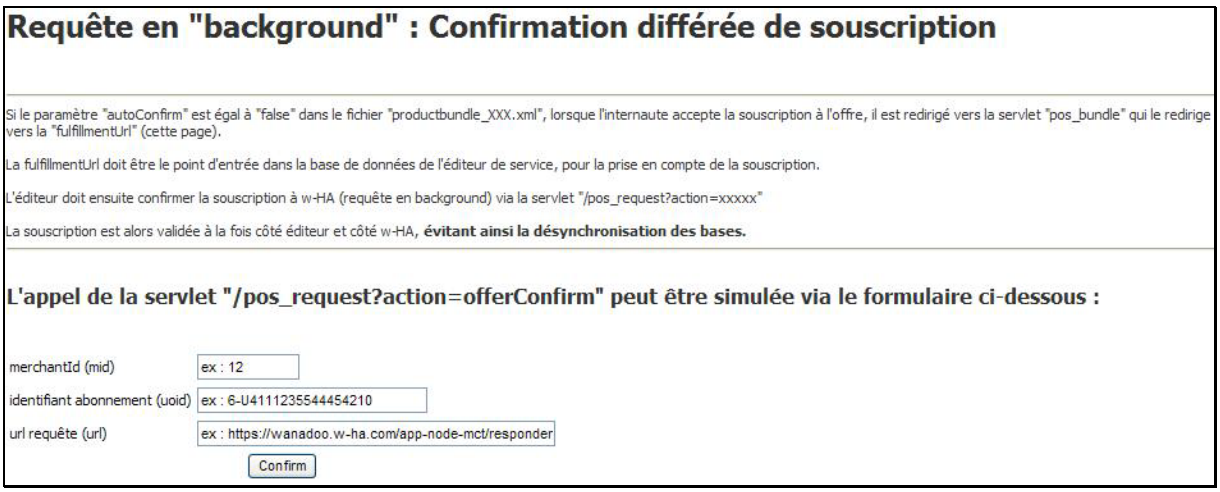
Example:
O1 and O2 are 1-month tacit renewal offers

UserId	UserOfferId (uoid)	oid	pid	Status	Date Souscription	Date Reconduction	Date Fin
fmoreau12	6-U7141248844587211	O1	P1	**Authorized**	21/8/7 12:16	21/9/7 12:16	null
msantos012	6-U8241248844587211	O2	P2	**Authorized**	22/8/7 15:12	22/9/7 15:12	null
fgalle45	none	O3	P1	Cancelled	null	null	null
gcronimus	6-U8223448846757123	O5	P2	**Authorized**	23/8/7 23:34	23/11/7 23:34	null
fgalle45	none	O1	P1	Cancelled	null	null	null
fmoreau12	6-U0441248844587211	O3	P1	**Authorized**	24/8/7 07:12	24/9/7 07:12	null

2.1.3 - Step #3 : Subscription confirmation, using the servlet : /pos_request?action=offerConfirm

The content provider sends a confirmation request (server to server) toward the w-HA platform.

Example:



Warning!

In order to differentiate the authorization and the confirmation, the **current offer**'s **« autoconfirm »** parameter's value must be « **false** », in the file « **productbundle_XXX.xml** »

Warning!

The **subscription's status** becomes « **confirmed** » in the w-HA database when the **confirmation request** is received. The client will then be **charged.**

Warning!

When a subscription is authorized, the content provider must **« confirm » this subscription within 24 hours**.
After time, the subscription is cancelled and can not be confirmed anymore.

2.1.4 - Step #4 : 2nd Content provider's database update

After the « confirm »'s acknowledgment reception coming from w-HA platform, the offer's subscription is considered as effective.

The content provider can **update his database**, in updating the **subscription's status** for instance.

The offer switches from the **authorized** to the **confirmed** status.

Example:

O1 and O2 are 1-month tacit renewal offers

UserId	UserOfferId (uoid)	oid	pid	Status	Date Souscription	Date Reconduction	Date Fin
fmoreau12	6-U7141248844587211	O1	P1	**Confirmed**	21/8/7 12:16	21/9/7 12:16	null
msantos012	6-U8241248844587211	O2	P2	**Confimed**	22/8/7 15:12	22/9/7 15:12	null
fgalle45	none	O3	P1	Cancelled	null	null	null
gcronimus	6-U8223448846757123	O5	P2	**Confirmed**	23/8/7 23:34	23/11/7 23:34	null
fgalle45	none	O1	P1	Cancelled	null	null	null
fmoreau12	6-U0441248844587211	O3	P1	**Confirmed**	24/8/7 07:12	24/9/7 07:12	null

2.2 - Access control / consumption (subscription life)

The cinematic is product oriented.

A cinematic product oriented is a cinematic for which the internet user is looking for the product/service consumption.

Two cases are possible:

- A – The internet user never suscribed to an offer containing this product/service.
- B - The internet user already suscribed to an offer (at least) containing this product/service.

The content provider must control in his database if the internet user already suscribed to the offer or not.

Warning!
If the content provider does not do the access control, the internet user can suscribe many times to a single offer!

2.2.1 – The internet user NEVER suscribed to an offer containing this product

If the internet user never suscribed to an offer containing this product, **the content provider must display the subscription offer (billing panel)**: calling the servlet /pos_bundle?action=authorizeOffer (§ 3.1.1 - Subscription demand / « pos_bundle?action=authorizeOffer »)

2.2.2 - The internet user ALREADY suscribed to an offer containing this product.

If the internet user already suscribed to an offer containing this product and the subscription is stil valid, **the content provider must redirect the internet user to the product, without any redirection to the w-HA platform.**

Warning!
The content provider is in charge of checking if the internet user subscription is valid or not.

Of course, the content provider is also in charge of updating his database if the subscription authorizes the internet user to:

- An unlimited access to the product
- A single access to the product
- N access to the product
- …..

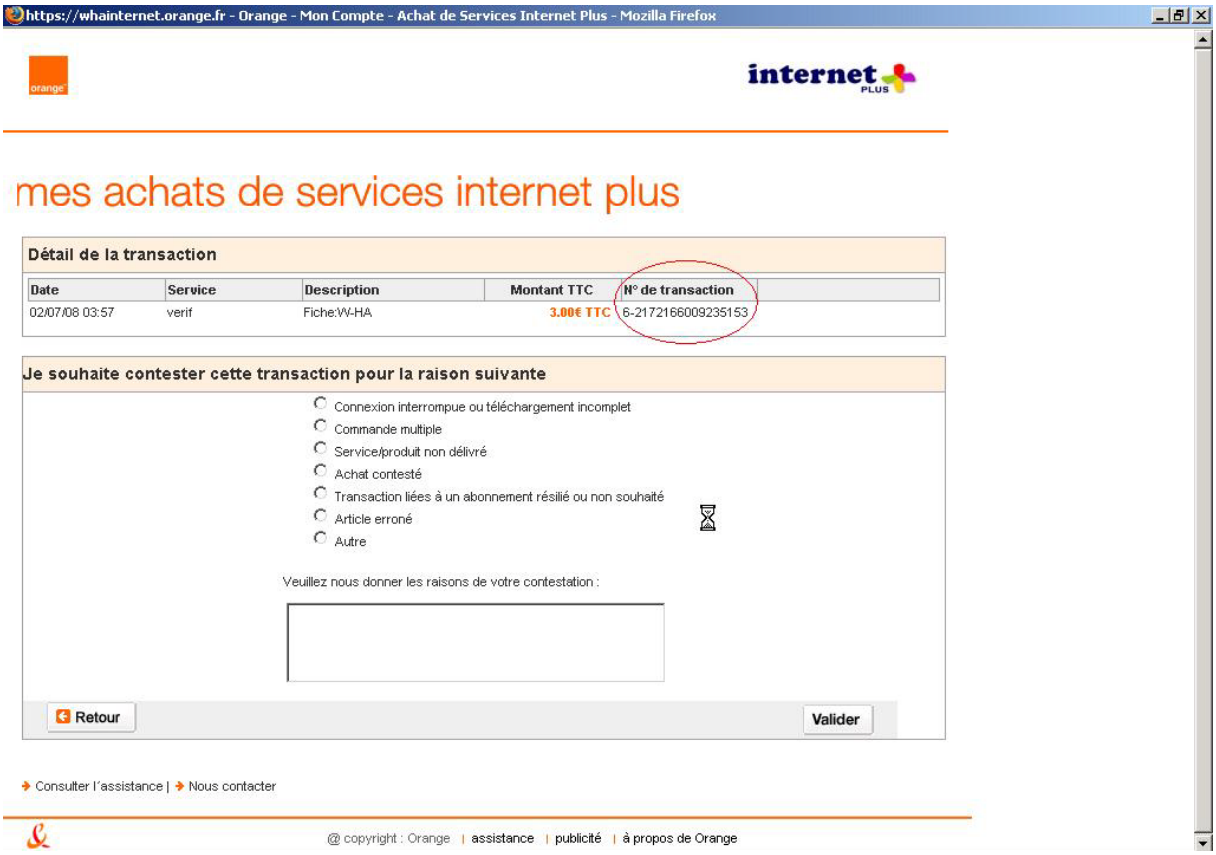
2.3 – Refund / Termination / Expiration (subscription end) / Content provider access control

This paragraph is about the subscription "end life". Its object is to explain how the content provider should update his database when this event happens.

2.3.1 – Refund of a transaction linked to a subscription

An internet user Orange (for instance) may on the web, the refund of a transaction:
www.orange.fr > espace client > conso internet (voir le détail) > Achats de service internet plus (voir le détail) > Consulter votre relevé détaillé
(direct access: https://wanadoo.w-ha.com/app-am/node)

In **tab « Mes achats »***(=my purchases),* click on « voir »*(=see)*:



Note: a **transaction ID** (for an Orange client) is like: **6-**XXXXXXXXXXXXXXX

The demand is transmitted to the internet access provider who decides (or not) the refund.

Warning!

If the internet access provider accepts the refund, **it can also terminate (immediately) the subscription** which initiated the transaction.

2.3.2 – Subscription termination

There are two ways to terminate a subscription:

2.3.2.1 – Subscription termination by the internet user (online)

The content provider can redirect any subscriber, who whishes to terminate his subscription, to the management interface of his internet access provider.

The content provider does not need to know the internet provider of the internet user. (Orange, Neuf-Cegetel, …).

The content provider just need to do a link to the URL https://route.w-ha.com/app-am/node on his web site, with the parameter subscription ID (uoid=X-UXXXXXXXXXXXXXXX).

Ex : https://route.w-ha.com/app-am/node?uoid=8-U123451234512345

When he is logged in, the internet user is redirected to the subscription details.

Possible cases:

- **no « uoid » transmited in the URL**: The client is redirected to the home page of the application “Mon Compte”(=my account) to the purchase list of the last month
- the **« uoid » transmited does not match with any valid subscription** for that merchant:The client is redirected to the home page of the application “Mon Compte”(=my account) to the purchase list of the last month
- the **« uoid » transmited matches to a valid subscription**: The client is redirected to the terminated subscription details

An internet user (Orange for instance) can demand online the subscription termination:
www.orange.fr > espace client > conso internet (voir le détail) > Achats de service internet plus (voir le détail) > Consulter votre relevé détaillé
(direct access: https://wanadoo.w-ha.com/app-am/node)

Click on **tab « Mes abonnements »***(=my subscriptions)*:

The termination demande is automatically accepted by the w-HA platform.

The subscription stays active until the end of its validity period (anniversary subscription date) and subscriptions products have to be consumed before that date.

The termination demands via that interface will intiate **2 termination notifications** to the content provider (servlet **/bundle-responder/responder**) (ex for Orange):
(§ 3.2.4 – Termination notification / « responder ?c=NMPOC_NEW »)

- a notification (**code r=200**) is sent **immediately** to the content provider for him to be informed that the subscription termination is **demanded** (and automatically accepted).
- a notification (**code r=201**) is sent **at the end of the current period** to the content provider for him to be informed that the subscription termination is **effective**. (The content provider must stop the access for the user and update his database)

2.3.2.2 – Subscription termination by the Clients Service Orange Internet

When a client does a termination demand to Clients Service Orange Internet, the internet access provider can:

- A – terminate the subscritpion **immediately** (because of a refund for instance)
- B - terminate the subscritpion **at the end of the current period** (anniversary date)

In both cases, only one termination notification is sent to the content provider: when termination is effective (immediately in case A or at the anniversary date case B)

2.3.3 – "push" mode: Termination notifications

When there is a subscription termination, a termination notification is send to the content provider on the /bundle-responder/responder servlet.
(§ 3.2.4 – Termination notification / « responder ?c=NMPOC_NEW »)
The content provider can update his database modifying the subscription status.

For example, the subscription status can be switched from "**Confirmed"** to:

- Unsubscribed (for an immediate termination) + end date (= termination date)
- To be unsubscribed (for a termination at expiry date) + end date (= anniversary date)

For example:

O1 and O2 are two offers tacitely renewable (TR) with a 1 month period.
The subscription 6-U7141248844587211 is terminated immediately. Demand on 28/09/2007 at 16:15
The subscription 6-U8241248844587211 is terminated at expiry date. Demand on 28/09/2007 at 16:15

UserId	UserOfferId (uoid)	oid	pid	Status	Subscription date	Renewal date (*)	End date (**)
fmoreau12	6-U7141248844587211	O1	P1	**Unsubscribed**	21/8/7 12:16	**null**	**28/9/7 16:15**
msantos012	6-U8241248844587211	O2	P2	**To be Unsubscribed**	22/8/7 15:12	22/9/7 15:12	**22/10/7 15:12**
fgalle45	none	O3	P1	Cancelled	null	null	null
gcronimus	6-U8223448846757123	O5	P2	**Confirmed**	23/8/7 23:34	23/11/7 23:34	null
fgalle45	none	O1	P1	Cancelled	null	null	null
fmoreau12	6-U0441248844587211	O3	P1	**Confirmed**	24/8/7 07:12	24/10/7 07:12	null

(*) The “Renewal date” column is used to notify:
- either the date of the **last renewal** (ex : line 2)
- either the date of the **next renewal** (ex : line 4 and 6)

(**)The “Renewal date” column is used to notify:
- either the **termination demand date** (ex : line 2)
- either the **effective termination date** (ex : line 1)

3 – Functioning details of each functionality / servlet

3.1 - Servlets with redirection of the internet user to the w-HA platform

The functionality “subscription” (cf 1.2.1 – The client subscribes) lead the internet user (via his browser) to the redirection to the w-HA platform.

Caution !

The « **pos_bundle** » sevlet **must be accessible from the public internet (port 80).**

3.1.1 - Subscription demand / « pos_bundle?action=authorizeOffer »

3.1.1.1 – Calling the servlet

When the content provider wishes to start a client subscription, he calls the “pos_bundle” servlet with the parameter **« action » at « authorizeOffer »**.

A button on the offer presentation page of the content provider calls this servlet.

3.1.1.2 – Calling example of the servlet

http://192.168.1.8/bundle/pos_bundle?
action=authorizeOffer
&oid=O1
&mid=13
&url=https://route.w-ha.com/app-bundlepurchase/node
&sessionId=1234
&userId=abcd

In red: Mandatory parameters
In bleue: Surcharge parameters
In green: Optional parameters (merchant properties)

3.1.1.3 – Calling parameters of the servlet

To activate a subscription, the “pos_bundle” servlet is called with the following settings:

Mandatory parametrers:

action=authorizeOffer: to indicate to the servlet that this is a subscription request

oid=*offer identifier*: offer identifier to which the user should subscribe

Surcharge parametrers:

mId=*shop identifier*: shop identifier affected by the “confirmation” request; the keyId values and the matching keyValue are collected by the servlet in the “merchants.xml” file.

url=*Url of the w-HA node (overrun)*: this setting is used:

- When the Content Provider uses the shop defined by default in the « web.xml » file but does not want to use the node by default. For example, by using his Internetplus shop on an operator’s portal and only offering payment to those operator subscribers (node operator.w-ha.com instead of route.w-ha.com)
- In a compulsory manner if the Content Provider has overrun the mid setting (see below). The node url to use by default is not specified in the “merchants.xml” file, so it must be specified here.

Warning !
For **Internet+**, the URL to be parametered is: **https://route.w-ha.com/app-bundlepurchase/node**

Optional parametrers:
Additional parameters of the content provider (or “merchant properties” (mp)) :

Those parameters are not theoretically obligatory. In general, the content provider uses at least one parameters type: “user ID”. This parameters allows him to associate the userID in his database to the w-HA subscription ID (uoid).

Merchant properties are in the subscription notification Url (fulfillmentUrl) which must be an executable (servlet, cgi, asp, php, ...)

Caution !
The additional parameter “wha_desc2=current” is obligatory if the billing apnel is displayed in current page.

3.1.1.4 – Servlet functioning

When it is called using the setting action=authorizeOffer, the “pos_bundle” servlet performs the following actions:

- Gets information connected to the shop (merchantId, URLs…) in the “web.xml”.
- Gets information connected to the offer to be subscribed by the user (OfferId…) in the “productbundle_XXX.xml” file.
- Gets shop settings (merchantId, keyId, keyValue, …) in the “merchants.xml” file if the mid setting has been overrun.
- Creates a secure and signed URL (https) calling the w-HA platform and containing this information as well as the settings (merchant properties) added by the merchant
- Redirects the internet user to this URL.
- Updates the “subscription-request.txt” log file

3.1.1.5 – URL example created by servlet

The internet user is rederected to an URL like the one bellow (encoded URL):

```
https://route.w-ha.com/app-
bundlepurchase/node?m=h%3D17c5d513e865a2995141ef60ef503ddd%3Bp%3D515%3Bk%3D515%
3Bv%3D3%3A%7Bc%3DOfferAuthorizeReq%3Bv%3D%7BmUrl%3Dhttp%3A%2F%192.168.1.8%2F
bundle%2Fpos_bundle%3Bpromo%3Dpromo%3Boid%3DO1%3Bmp%3D%7B_ap_userId%3Dabcd%
3B_ap_sessionId%3D1234%3Bts%3D2008-04-
14+14%3A38%3A15.25%3Bcur%3DEUR%3B%7D%3B%7D%7D
```

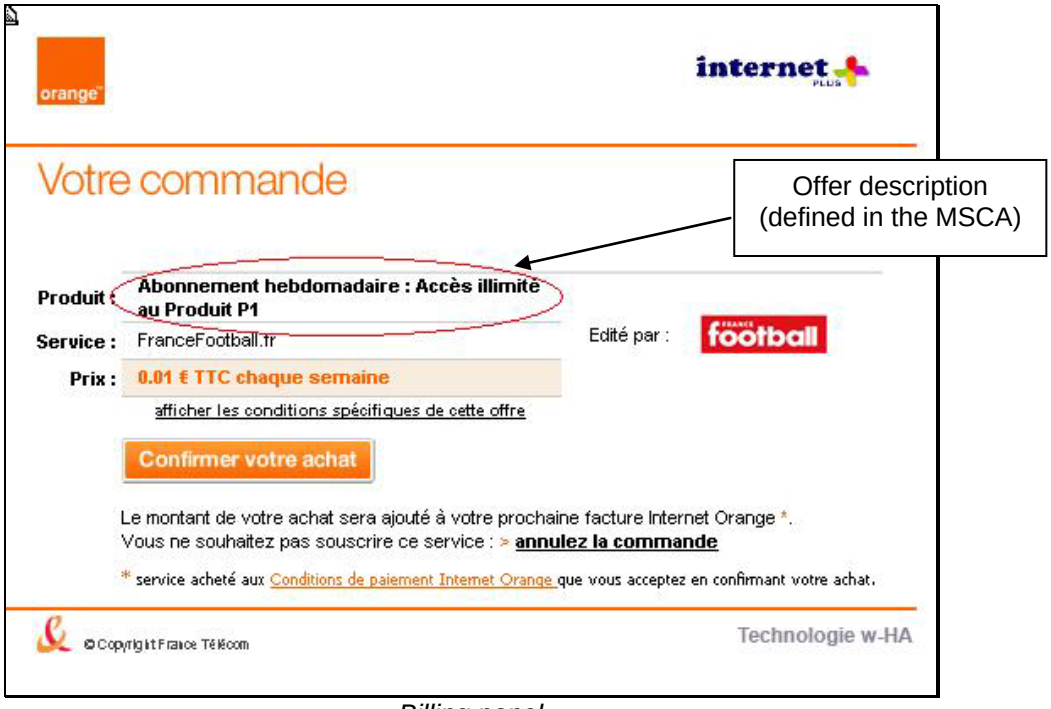
Decoded URL:

```
https://route.w-ha.com/app-bundlepurchase/node?
m=h=17c5d513e865a2995141ef60ef503ddd;
p=515;
k=515;
v=3:{
c=OfferAuthorizeReq;
v={
mUrl=http://192.168.1.8/bundle/pos_bundle;
promo=promo;
oid=O1;
mp={
_ap_userId=abcd;
_ap_sessionId=1234;
ts=2008-04-14 14:38:15.25;
cur=EUR;};}}
```

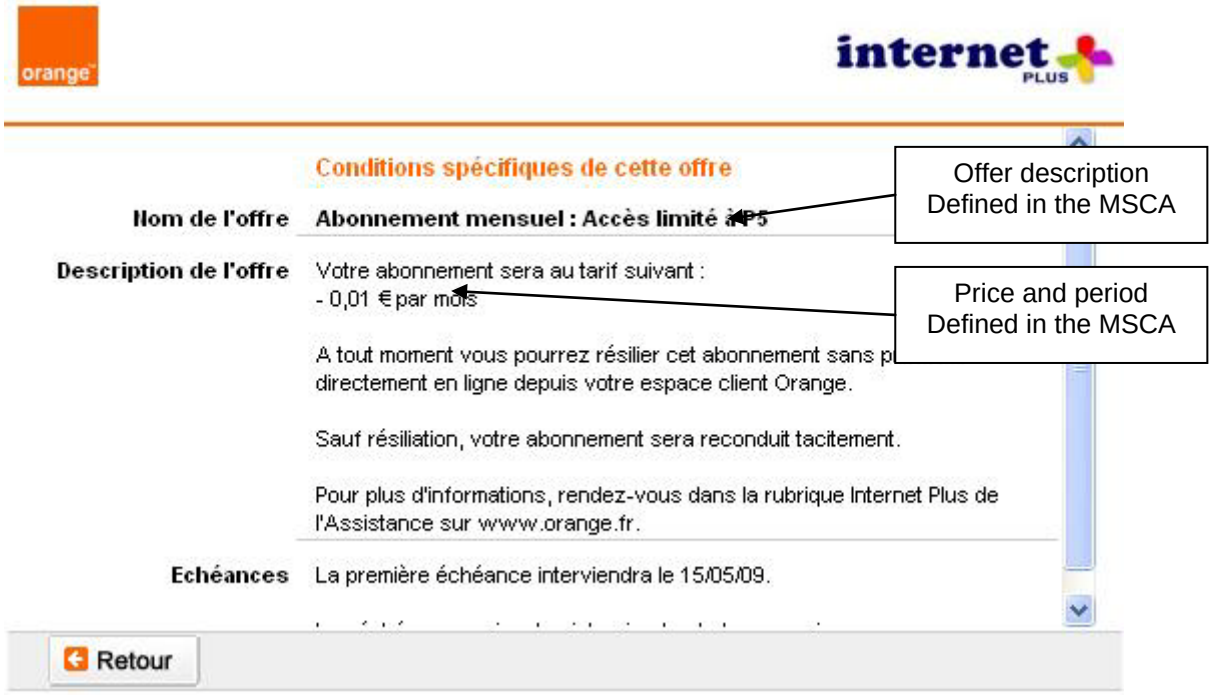
This url is securised (https) and signed).

3.1.1.6 – w-HA response to the servlet

When the user is redirected to the w-HA platform for a subscription request (see previous paragraph), a payment panel is displayed:



Billing panel



Termes and conditions

The user can either, accept the subscription or refuse.

1st case: user refuses the subscription

If the user clicks on the “cancel order” link, the subscription is not validated on the w-HA platform: no subscription identifier is generated.

The user is redirected to the Content Provider “pos_bundle” servlet (merchant.callback.url) with a “**OfferAuthorizeCancel**” message.

URL example (**pos_bundle**) to which the user is redirected in case of cancellation from the payment panel:

```
http://192.168.1.8/bundle/pos_bundle?m=h%3D032e20600035df64a059644b9c12593a%3Bp%3D515%3Bk%3D515%3Bv%3D3%3A%7Bc%3DOfferAuthorizationCancel%3Bv%3D%7B_ap_userId%3Dabcd%3B_ap_sessionId%3D1234%3Bts%3D2008-04-14+14%3A46%3A44.343%3Bcur%3DEUR%3B%7D%7D
```

Decoded URL:

```
http://192.168.1.8/bundle/pos_bundle?
m=h=032e20600035df64a059644b9c12593a;
p=515;
k=515;
v=3:{
c=OfferAuthorizationCancel;
v={
_ap_userId=abcd;
_ap_sessionId=1234;
ts=2008-04-14 14:46:44.343;
cur=EUR;}}
```

This URL is secure (https) or not (http) and signed.

Information included in this URL are:
- an hmac to ensure message integrity,
- "OfferAuthorizeCancel" message
- Additional Content Provider settings (mp) (with the prefix : _ap_)

The "pos_bundle" servlet updates the "log_response_XXX.txt" log file and redirects the user towards the **mcttrxCancelFromPaymentPanelUrl** set in the "merchants.xml" file.

URL example (**mcttrxCancelFromPaymentPanelUrl**) to which the user is redirected in case of cancellation from the payment panel:

http://192.168.1.4/demo/bundle/html/panel_cancel1.html?hmac=514f1bec55a3e95639a41a9e46af6b8b&cur=EUR&sessionId=1234&ts=2008-04-14+14%3A46%3A44.343&userId=abcd

Decoded URL:

http://192.168.1.4/demo/bundle/html/panel_cancel1.html?
hmac=514f1bec55a3e95639a41a9e46af6b8b
&cur=EUR
&sessionId=1234
&ts=2008-04-14 14:46:44.343
&userId=abcd

This URL is secure (https) or not (http) and signed.

Information included in this URL are:
- an hmac to ensure message integrity,
- merchant properties (cur + Content Provider's own settings)

2nd case: operator refuses the subscription (via the w-HA platform)

If the user clicks on the "Confirm your purchase" button, but the subscription cannot be accepted (monthly limit reached, operator invoice payment refused, …) by the w-HA platform, **no subscription identifier is created**.

An error message is displayed for the user, directly on the payment panel.

There is not return to the "pos_bundle" servlet to specify the error cause.
Consequently, no line is added to the new field "log_request_XXX.txt"

3rd case: user subscription acceptance

If the user clicks on the "Confirm your purchase" button and the subscription is accepted by the w-HA platform, **a subscription identifier is created with an authorized status**.

The user is redirected to the Content Provider "pos_bundle" server (merchant.callback.url) with a "**OfferAuthorizeSuccess**" message.

URL example to which the user is redirected in case of cancellation from the payment panel:

http://192.168.1.8/bundle/pos_bundle?m=h%3D66d1ce95dbd08060aa42c0f66696ca09%3Bp%3D515%3Bk%3D515%3Bv%3D3%3A%7Bc%3DOfferAuthorizationSuccess%3Bv%3D%7Bmp%3D%7B_ap_userId%3Dabcd%3B_ap_sessionId%3D1234%3Bts%3D2008-04-14+15%3A45%3A59.515%3Bcur%3DEUR%3B%7D%3Boid%3DO4%3Bru%3Dhttps%3A%2F%2Fwanadoo.w-ha.com%2Fapp-node-mct%2Fresponder%3Bg_amt%3D0.01%3Bz%3D92442%3Bco%3DFR%3Buoid%3D6-U5117575881274524%3Bst%3DFR%3Bci%3DISSY+LES+MOULINEAUX+CEDEX%3B%7D%7D

Decoded URL:

http://192.168.1.8/bundle/pos_bundle?
m=h=66d1ce95dbd08060aa42c0f66696ca09;
p=515;
k=515;
v=3:{
c=OfferAuthorizationSuccess;
v={
mp={
_ap_userId=abcd;
_ap_sessionId=1234;
ts=2008-04-14 15:45:59.515;
cur=EUR;};
oid=O4;
ru=https://wanadoo.w-ha.com/app-node-mct/responder;
g_amt=0.01;
z=92442;
co=FR;
uoid=6-U5117575881274524;
st=FR;
ci=ISSY LES MOULINEAUX CEDEX;}}

This URL is signed.

Information included in this URL are:

- an hmac to ensure message integrity
- "OfferAuthorizeSuccess" (c=)
- Subscription amount (g_amt=)
- additional Content Provider settings (mp=)
- The URL to confirm the subscription (ru=)
- the subscription identifier (uoid=)

Upon receipt of this request, the "pos_bundle" servlet then updates the "log_response_XXX.txt" log file.

Two cases may then appear regarding this offer depending on the value of the "autoConfirm" setting for the "productbundle_XXX.xml" file:

*** autoConfirm = true**

If the "autoConfirm" setting of the offer is set at "true", upon receipt of a "OfferAuthorizeSuccess" message, the "pos_bundle" servlet will **automatically and immediately** activate the **subscription confirmation**
(§ 2.1.3 - Step #3 : Subscription confirmation using the servlet : /pos_request?action=offerConfirm).

As for the w-HA, the subscription goes to a "confirmed" status.

For the Content Provider, the "pos_bundle" servlet redirects the user to the offer fulfillmentUrl, set in the "productbundle_XXX.xml" file.

The Content Provider must implement an executable file (CGI, php, jsp, coldfusion,…) for the fulfillmentUrl which will be responsible for acknowledging the subscription.

*** autoConfirm = false**

If the "autoConfirm" setting of the offer is set at "false", upon receipt of a "OfferAuthorizeSuccess" message, the "pos_bundle" servlet will redirect the user to the offer **fulfillementUrl** set in the "productbundle_XXX.xml" file.

The Content Provider must implement an executable file (CGI, php, jsp, coldfusion,…) for the fulfillmentUrl which will be responsible for acknowledging the subscription.

The Content Provider has **24h hours maximum** (from the offer authorization date) **to confirm the subscription** (and to make it effective on the w-HA platform) using an SDK supplied the w-HA (see annex I).

Caution !!

If the Content Provider does not confirm the subscription, it will be valid for the Content Provider, but not for the w-HA thus causing **desynchronization of the databases**.

3.1.1.7 – To take subscription into consideration (offer fulfillmentUrl)

When the "pos_bundle" servlet receives a positive subscription request reply form the w-HA platform, it redirects the user to the subscription acknowledgement URL ("**fulfillmentUrl**") set in the "productbundle_XXX.xml" file.

Example of subscription confirmation URL, created by the "pos_bundle" servlet:

```
http://192.168.1.8/demo/bundle/html/validOffre4.html?hmac=7633f0af106fbef2b0d644c0c9aefcd0&cur=EUR&oid=O4&sessionId=1234&ts=2008-04-14+15%3A54%3A26.093&uoid=6-U4645012218882221&userId=abcd
```

Decoded URL:

```
http://192.168.1.8/demo/bundle/html/validOffre4.html?
hmac=7633f0af106fbef2b0d644c0c9aefcd0
&cur=EUR
&oid=O4
&sessionId=1234
&ts=2008-04-14 15:54:26.093
&uoid=6-U4645012218882221
&userId=abcd
```

This URL is secure (https) or not (http) and signed.

Information included in this URL are:

- a hmac to ensure message integrity (h=),
- the subscription identifier (uoid=)
- the offer identifier (oid=)
- additional Content Provider settings (mp)

Warning!

To calculate the hmac, all of the settings must be considered including the subscription identifier (uoid).

For the subscription acknowledgement URL, **the Content Provider must associate the subscription identifier that has just been subscribed (uoid) with the username (ex: userId) in its own application (back-office).**

UserId=fmoreau12 ⟷ uoid=6-U7141248844587211

It can then **update its database with the subscription identifier (w-HA)** by specifying the **subscription status** (valid or not) and until when (§ 2.1.2 - Step #2 : 1st CP database update).

3.1.2 – To reload local configuration file / « pos_bundle?action=reload »

When the content provider modifies configuration files (.xml) he must rebout the “bundle” application in his servlet engine.

Calling the “pos_bundle” servlet with the parameter “action=reload” reloads configurations in xml files. (w-HA application is not available during the reboot)

Concerned files are:

- merchants.xml
- productbundle_XXX.xml

3.2 – Servlets for server-to-server requests

Functionalities bellow work with server-to-server requests **https**:

- deferred confirmation (“pos_request” servlet : CP → w-HA)
- subscription termination (unsubscription) (“pos_request” servlet : CP → w-HA)
- transaction refund (“pos_request” servlet : CP → w-HA)
- “push” termination notification (“responder” servlet : w-HA → CP)

(CP = Content Provider)

Warning!

“responder” and “**pos_request”** servlets functionalities **must NOT be avialable from public internet (port 80)**.

Warning!

Because those requests are https, network equipments need to be correctly configured for requests form/to w-HA platform.

3.2.1 - « Deferred » confirmation / pos_request?action=offerConfirm

Contrary to the subscription demand and its response, which are http(s) redirections, **the subscription confirmation is a server-to-server request (https)**.

It confirms the subscription on the w-HA platform: The subscription status switches from “authorized” to “confirmed”.

When the “**autoConfirm**” setting relative to an offer is equal to “**true**” in the “productbundle_XXX.xml” file, the **« pos_bundle » servlet automatically sends the confirmation request** to the w-HA platform.

When the “**autoConfirm**” setting relative to an offer is equal to “**false**” (recommended by Internet+ to avoid database desynchronizations) in the “productbundle_XXX.xml” file, the **Content Provider must send the confirmation request** to the w-HA platform. This request is called “deferred confirmation request”

3.2.1.1 – Calling the servlet

When the internet user subscribes to an offer, it is redirected to the subscription validation URL defined in the “products.xml” file (fulfillmentUrl).

When the internet user reaches this URL, this will activate a program (to be developed by a Content Provider in php, java, coldfusion) which will:

1) **collect** the subscribed package identifiers (**uoid**),
2) **update the Content Provider’s subscription database**
3) call the “**pos_request**” servlet (**action=offerConfirm**) to validate the subscription on the w-HA side.

3.2.1.2 - Example of calling the servlet

To confirm a subscription, the Content Provider will call the servlet **via his HTTP client (to be developed)** by transmitting the necessary settings.

In order to do this he will use a URL:

http://*wha.marchand.com*/bundle/pos_request?
action=offerConfirm
&mid=10
&uoid=6-U7141248844587211
&url=https://wanadoo.w-ha.com/app-node-mct/responder

In red: Mandatory parameters
In bleue: Surcharge parameters

3.2.1.3 – Calling parameters of the servlet

For delayed subscription confirmation, the “pos_request” servlet is called with the following settings:

action=offerConfirm: to indicate to the servlet that this is a subscription confirmation request.

mId=*shop identifier*: shop identifier affected by the “confirmation” request; the keyId values and the matching keyValue are collected by the servlet in the “merchants.xml” file.

uoid=*the subscription identifier*: subscription identifier that has just been subscribed by the internet user and informed in the Content Provider database.

url=*Url of the w-HA node (responder)*: the w-HA url node which will receive the subscription confirmation request.

The URL node to call depends on the ISP of the subscription.
This information is featured in the subscription identifier (first digits).

The identifier/operator/url matches are given in the table below:

Identifier	Operator	URL (responder)
6-XXX…XXX	Orange Internet	https://wanadoo.w-ha.com/app-node-mct/responder
7- XXX…XXX	Alice/Free	https://free.w-ha.com/app-node-mct/responder
25- XXX…XXX	Orbeo	https://cb.w-ha.com/app-node-mct/responder

3.2.1.4 – Servlet functioning

When the "pos_request” servlet is called with the parameter action=offerConfirm, it does actions bellow:

- Gets information of the merchant in the « web.xml » file (merchantId, URLs, …)
- Gets parameters of the service (merchantId, keyId, keyValue) in the « merchants.xml » file
- Creates a securised URL (https) and signed, containing the offer ID (uoid) to be confirmed, to the w-HA platform
- Sends an https request (in background) to the w-HA platform.

3.2.1.5 – URL example created by the servlet

URL example of a subscription confimation, created by the « pos_request » servlet:

```
https://wanadoo.w-ha.com/app-node-
mct/responder?m=h%3D8e94261071cba9dcaf53b81759f5ab2a%3Bp%3D515%3Bk%3D515%3Bv%3D3%3A%7
Bc%3Dm_offerConfirm%3Bv%3D%7Buoid%3D6-U2143613233868231%3B%7D%7D
```

Decoded URL:

```
https://wanadoo.w-ha.com/app-node-mct/responder?
m=h=8e94261071cba9dcaf53b81759f5ab2a;
p=515;
k=515;
v=3:{
c=m_offerConfirm;
v={
uoid=6-U2143613233868231;}}
```

This URL is not securised (http) and signed.

Information included in this URL are:
- a hmac to ensure message integrity (h=),
- the “m_offerConfirm” (c=) command
- the offer ID to be confirmed (uoid=)

When the w-HA platform receives a subscription confirmation, it sends an acknowledgment (ack) to the « pos_request » servlet.

3.2.1.6 – w-HA platform response to deferred confirmation request

When **the offer subscription confirmation (=>subscription) (uoid) is validated** the w-HA platform returns an acknowledgment “c=ack”.

If an error occurs, the platform returns a specific error code (e=code) based on the following nomenclature:

Code	Comment
0	Unknown subscription (uoid): « User offer not found »
1	Previously confirmed offer subscription (uoid): « Invalid user offer status ». Generic return code when a subscription (uoid) no longer has an active status (ex: the confirmation is sent after 24 hours from the time of subscribing)

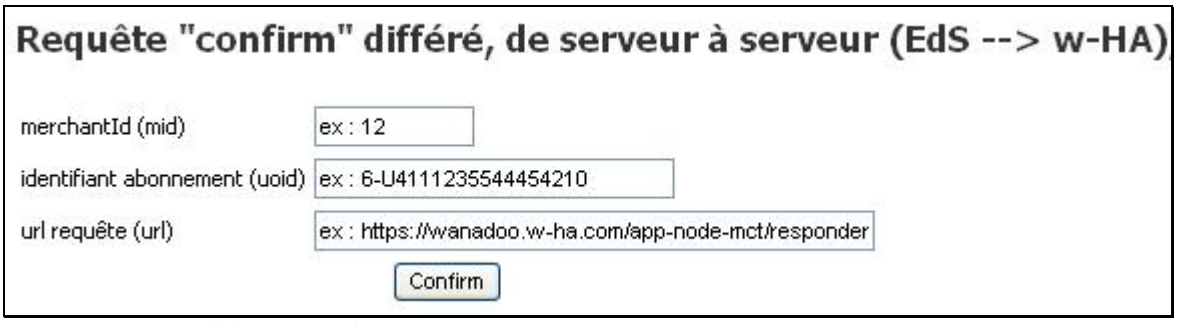
Example : e=1

3.2.1.7 - Example of using the servlet: web interface

A web interface, which uses the “pos_request” servlet (action=offerConfirm) is also available to the Content Provider.

It enables the Content Provider:
- to properly understand how the servlet works
- to perform access controls manually if necessary.

This web interface can be found a the following URL:
http://*wha.merchant.com*/demo/bundle/html/controle_acces.html (*):
(*) replace wha.merchant.com by the IP adress or public domain name of the server on which the w-HA application is installed.



Requête “confirm” différé de serveur à serveur = Delayed “confirm” request from server to server

3.2.2 – Subscription termination / « pos_request ?action=cancelOffer

The content provider can terminate a subscription with a server to server request to the w-HA platform.

The unsubscription can be :
- **with immediate effect (deferred=0=false)**
- **at expiry date (deferred=1=true)**

Note:
- In case of an unsubscription at expiry date, the subscription will be terminated at the end of the commitment period.
- In case of an unsubscription with immediate effect, the subscription will be terminated immediately even if there is a commitment period.

3.2.2.1 – Calling the servlet

When an internet user wishes to terminate his subscription, he can:
- Contact the Orange customer service (Internet)
- Contact the content provider customer service

The content provider can send to w-HA platform an unsubscription request (with the subscription identifier (uoid), the moment to terminate the subscription (immediately or deferred) and comments).

3.2.2.2 - Example of calling the servlet

To terminate a subscription, the Content Provider will call the servlet **via his HTTP client (to be developed)** by transmitting the necessary settings.

In order to do this he will use a URL:

```
http://wha.marchand.com/bundle/pos_request?
action=cancelOffer
&mid=10
&uoid=6-U1741248844587211
&url=https://wanadoo.w-ha.com/app-node-mct/responder
&deferred=0
&userComment=client comment
&adminComment=content provider comment
&reasonCode=110
```

3.2.2.3 – Calling parameters of the servlet

For unsubscription request, the “pos_request” servlet is called with the following settings:

action=cancelOffer: to indicate to the servlet that this is an unsubscription request.

mId=*shop identifier*: shop identifier affected by the unsubscription request; the keyId values and the matching keyValue are collected by the servlet in the “merchants.xml” file.

uoid=*the subscription identifier*: subscription identifier that has just been unsubscribed by the internet user and informed in the Content Provider database.

url=*Url of the w-HA node (responder)*: the w-HA url node which will receive the unsubscription request.
(ex: Orange (Internet) node is: https://wanadoo.w-ha.com/app-node-mct/responder)

deffered=0/1 or **deffered =false/true:** to indicate if the unsubscription must be done immediately (deferred=0=false) or deferred to the end of the subscription period (deferred=1=true)

userComment: internet user comment on the reason of his unsubscription demand to the content provider.

adminComment: content provider comment on the reason of his unsubscription demand to w-HA.

reasonCode:

Code (r=)	Comment
110	internet user unsubscription demand (else)
111	internet user unsubscription demand (multiple command)
112	internet user unsubscription demand (multiple billing)
113	internet user unsubscription demand (undelivery service)
114	internet user unsubscription demand (subscription contested)
115	internet user unsubscription demand (child subscription)
116	internet user unsubscription demand (transaction due to a terminated subscription or an unwanted one)
117	Subscription terminated: I would like to be refund
118	The service does not correspond to my expectations
119	I subscribed many times to this single offer

3.2.2.4 – Servlet functioning

When the servlet « pos_request » is called with the parameter action=cancelOffer, it does actions bellow:

- Gets information of the merchant in the « web.xml » file (merchantId, URLs, …)
- Gets parameters of the service (merchantId, keyId, keyValue) in the « merchants.xml » file
- Creates a securised URL (https) and signed, containing the offer ID (uoid) to be unsubscribed, to the w-HA platform
- Sends an https request (in background) to the w-HA platform.

3.2.2.5 – URL example created by the servlet

URL example of an unsubscription request, created by the « pos_request » servlet:

```
https%3A//wanadoo.w-ha.com/app-node-
mct/responder%3Fm%3Dh%3Dfd4685944ae52d91601e343f7f561d3c%3Bp%3D13%3Bk%3D13%3B
v%3D3%3A%7Bc%3Dm_userOfferUnsubscribe%3Bv%3D%7Badmincom%3DCommentaire%20edite
ur%3Bdc%3D0uoid%3D6-
U714124844587211%3Breason%3D110%3Busercom%3DCommentaire%20client%3B%7D%7D
```

Decoded URL:

```
https://wanadoo.w-ha.com/app-node-mct/responder?
m=h=fd4685944ae52d91601e343f7f561d3c;
p=13;
k=13;
v=3:{
c=m_userOfferUnsubscribe;
v={
admincom=Commentaire editeur;
dc=0
uoid=6-U714124844587211;
reason=110;
usercom=Commentaire client;}}
```

This URL is not securised (http) and signed.

Information included in this URL are:
- a hmac to ensure message integrity (h=),
- the “m_userOfferUnsubscribe” (c=) command
- the offer ID to be confirmed (uoid=)

3.2.2.6 - w-HA platform response to deferred confirmation request

When the subscription identifier (**uoid**) **exists** and the subscription has a **status** which **allows its unsubcription** :

- the w-HA platform returns:
h=46e73ffadc98fa7f29d5fcb119b198f9;p=5XXX;k=5XXX;v=2:{c=ack}

- the kit returns : **e=0**

If an error occurs, the platform returns a specific error code (e=code) based on the following nomenclature:

Code	Comment
1	unsubscription failure
3	wrong hmac

Example: **e=1**

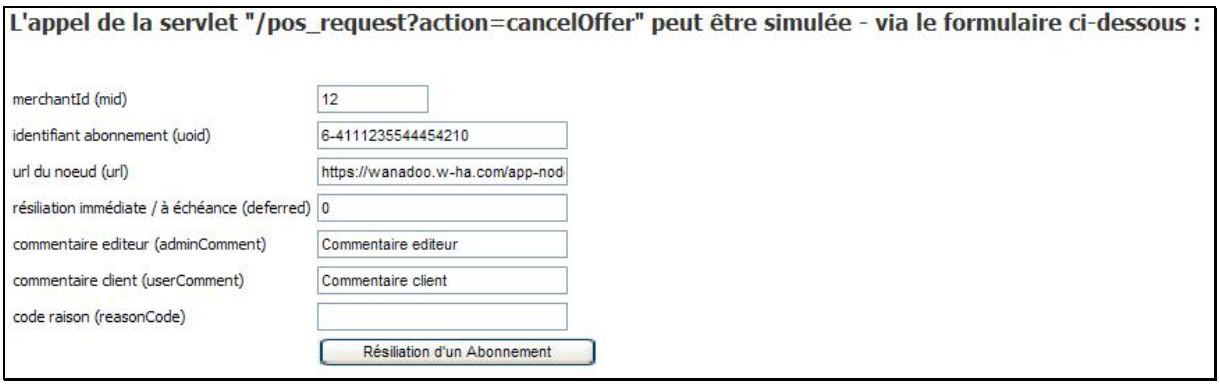
Note: If you try to **unsubscribe many times the same subscription**, the w-HA platform is **idempotent for unsubscription requests** (it will always return the same acknowledgment message) so the w-HA platform returns: **h=46e73ffadc98fa7f29d5fcb119b198f9;p=5XXX;k=5XXX;v=2:{c=ack}** (The kit will return e=1)

3.2.2.7 - Example of using the servlet: web interface

A web interface, which uses the “pos_request” servlet (action=cancelOffer) is also available to the Content Provider.

It enables the Content Provider:
- to properly understand how the servlet works
- to perform access controls manually if necessary

This web interface can be found a the following URL:
http://*wha.marchand.com*/demo/bundle/html/resiliation_abonnement.html (*):
(*) replace wha.merchant.com by the IP adress or public domain name of the server on which the w-HA application is installed.



Warning!

Access to this page must be retricted to the content provider and **must NOT be avialable to public internet users**.

Otherwise they could terminate subscriptions via this « demo » page !

3.2.3 – Transaction refund / « pos_request ?action=refund »

The content provider can refund a transaction with a server to server request to the w-HA platform.

Warning !

To respect the process the Orange customer service, you **must send a unsubscription request** with immediate effect (deferred=0=false) **after the refund** of a subscription.

3.2.3.1 – Calling the servlet

When an internet user wishes to terminate his subscription, he can:

- Contact the Orange customer service (Internet)
- Contact the content provider customer service

The content provider can send to w-HA platform a refund request (with the transaction identifier (**trxid**) to be refund, and **comments**).

3.2.3.2 - Example of calling the servlet

To refund a transaction, the Content Provider will call the servlet **via his HTTP client (to be developed)** by transmitting the necessary settings.

In order to do this he will use a URL:

http://wha.marchand.com/bundle/pos_request?
action=refund
&mid=10
&trxid=6-17141248844587211
&url=https://wanadoo.w-ha.com/app-node-mct/responder
&userComment=commentaire client
&adminComment=commentaire editeur
&reasonCode=110

3.2.3.3 – Calling parameters of the servlet

For refund request, the “pos_request” servlet is called with the following settings:

action=refund: to indicate to the servlet that this is a refund request.

mId=*shop identifier*: shop identifier affected by the refund request; the keyId values and the matching keyValue are collected by the servlet in the “merchants.xml” file.

trxid=*the transaction identifier*: transaction identifier to be refund.

url=*Url of the w-HA node (responder)*: the w-HA url node which will receive the refund request.
(ex: Orange (Internet) node is: https://wanadoo.w-ha.com/app-node-mct/responder)

userComment: internet user comment on the reason of his unsubscription demand to the content provider.

adminComment: content provider comment on the reason of his unsubscription demand to w-HA.

reasonCode:

Code (r=)	Comment
110	internet user refund demand (else)
111	internet user refund demand (multiple command)
112	internet user refund demand (multiple billing)
113	internet user refund demand (undelivery service)
114	internet user refund demand (subscription contested)
115	internet user refund demand (child subscription)
116	internet user refund demand (transaction due to a terminated subscription or an unwanted one)
117	Subscription terminated: I would like to be refund
118	The service does not correspond to my expectations
119	I subscribed many times to this single offer

3.2.3.4 – Servlet functioning

When the servlet « pos_request » is called with the parameter action=refund, it does actions bellow:

- Gets information of the merchant in the « web.xml » file (merchantId, URLs, …)
- Gets parameters of the service (merchantId, keyId, keyValue) in the « merchants.xml » file
- Creates a securised URL (https) and signed, containing the transaction ID (trxid) to be refund, to the w-HA platform
- Sends an https request (in background) to the w-HA platform.

3.2.3.5 – URL example created by the servlet

URL example of a refund request, created by the « pos_request » servlet:

https%3A%2F%2Fwanadoo.w-ha.com%2Fapp-node-mct%2Fresponder%3Fm%3Dh%3Dfd4685944ae52d91601e343f7f561d3c%3Bp%3D13%3Bk%3D13%3Bv%3D3%3A%7Bc%3Dm_fullRefund%3Bv%3D%7Badmincom%3DCommentaire%20editeur%3Brid%3Drq004442%3BtrxId%3D6-17141248844587211%3Breason%3D110%3Bd%3D0%3Busercom%3DCommentaire%20client%3B%7D%7D

Decoded URL:

https://wanadoo.w-ha.com/app-node-mct/responder?
m=h=fd4685944ae52d91601e343f7f561d3c;
p=5XXX;
k=5XXX;
v=3:{
c=m_fullRefund;
v={
admincom=Commentaire editeur;
rid=rq004442;
trxId=6-17141248844587211;
reason=110;
d=0;
usercom=Commentaire client;}}

This URL is securised (https) and signed.

Information included in this URL are:

- a hmac to ensure message integrity (h=),
- the “m_fullRefund” (c=) command
- the transactionID to be refuned (trxId=)

3.2.3.6 – w-HA platform response to deferred confirmation request

When the transaction identifier (**trxid**) **exists** and the transaction has a **status** which **allows its refund**, the w-HA platform returns:
The refund demand identifier: trxid=RR1234567891234567

If an error occurs, the platform returns a specific message based on the following nomenclature:

Message	Comment
h=a0e198418d3a05ed657c1cfce78c0964;p=5XXX;k=5XXX;v=2:{c=ex;v={m=INVALID_MERCHANT_INFO;t=com.ipin.core.api.node.pos.MerchantTransactionRefundManager$RefundException;c=2;}}	Refund fail
e=3	Wrong hmac
e=15	Error in the transmitted parameters (example : "trx**i**d=" instead of "trx**I**d=")

Example :
h=a0e198418d3a05ed657c1cfce78c0964;p=5XXX;k=5XXX;v=2:{c=ex;v={m=INVALID_MERCHANT_INFO;t=com.ipin.core.api.node.pos.MerchantTransactionRefundManager$RefundException;c=2;}}

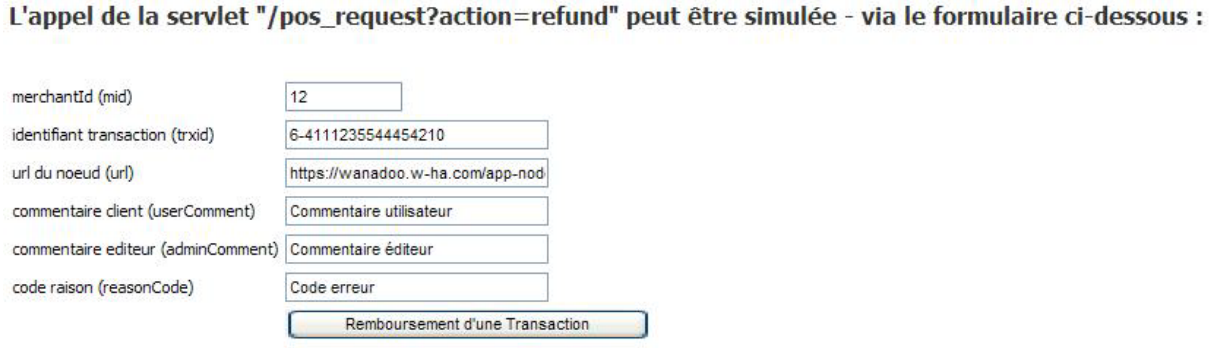
Note: If you try to refund **many times the same transaction**:
The w-HA platform is **not idempotent for refund requests**, for the first request the w-HA Kit will return the **refund demand identifier** (trxid=RR1234567891234567) then for followings request the w-HA Kit platform return:
"**h=a0e198418d3a05ed657c1cfce78c0964;p=5XXX;k=5XXX;v=2:{c=ex;v={m=INVALID_MERCHANT_INFO;t=com.ipin.core.api.node.pos.MerchantTransactionRefundManager$RefundException;c=2;}}**" (the refund is already effective).

3.2.3.7 - Example of using the servlet: web interface

A web interface, which uses the “pos_request” servlet (action=refund) is also available to the Content Provider.

It enables the Content Provider:
- to properly understand how the servlet works
- to perform access controls manually if necessary

This web interface can be found a the following URL:
http://*wha.merchant.com*/demo/bundle/html/remboursement_transaction.html (*):
(*) replace wha.merchant.com by the IP adress or public domain name of the server on which the w-HA application is installed.



Warning!

Access to this page must be retricted to the content provider and **must NOT be avialable to public internet users**.

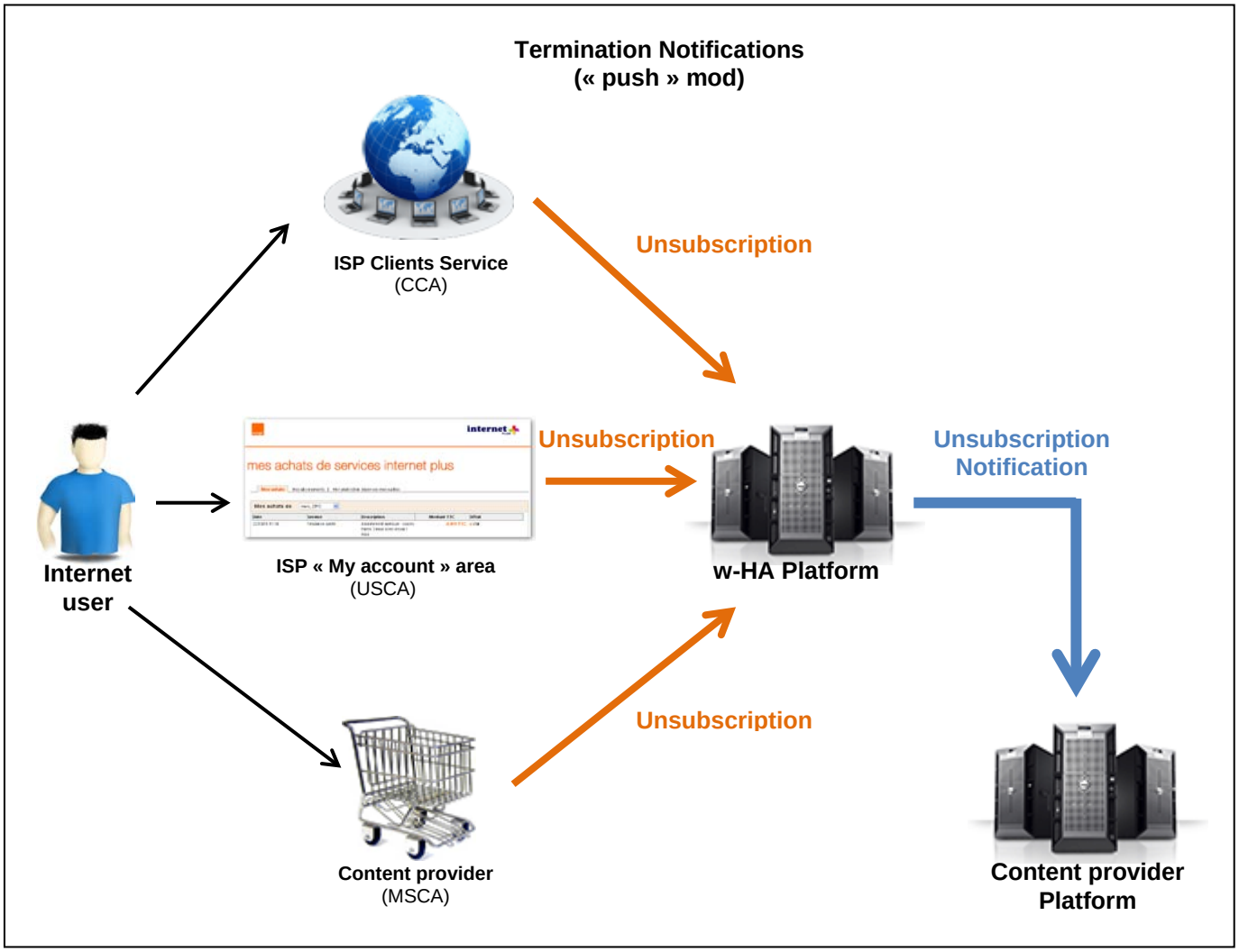
Otherwise they could refund transactions via this « demo » page !

3.2.4 – Termination notification / « responder ?c=NMPOC_NEW »

Access control is performed by the Content Provider for Internet+ offers, communication regarding any event connected to a current subscription (cancellation, refund,…) must be made in real time.

The w-Ha (v3.5) platform includes a **"push" mechanism** (w-HA to the Content Provider) enabling the Content Provider to immediately update its database with the new subscription status.

For the application installed on the Content Provider server, a **"responder" servlet** (or a similar development around the SDK) **is in charge of collecting this information**.



3.2.4.1 – Calling the servlet

A client can, at any time, ask for the unsubcritpion of one of his services.
For that, there are 4 tools which permit to proceed to an unsubscrption:

1. The “espace Mon compte" of the client
2. The service provider’s MSCA
3. The service provider servlet pos_request (action = cancelOffer)
4. The CCA (ISP tool)

Depending on the tool used, the service provider wil not receive the same unsubscription notifications.

If the unsubscription is asked through the **“espace “Mon Compte“**of the client, the service provider will receive 2 notifications:

- The first when the unsubscription has been asked
- The second when the unsubscription will come into effect.

In the case of an unsubscription coming from one of the 3 others tools, the service provider will receive only one notification: when the unsubscription will come into effect.

CASE 1 : Unsubscription from the "espace Mon Compte"

A notification will be sent by W-ha to the **servlet Responder** of the service provider (server to server request) :

*** When the termination is asked by the internet user**

The code used for this notification is: **r=200**.
This notification gives the **date and time** of the internet user demand in the “comments” field (c=).

http://www.w-ha.com/kit/responder.php?m=h=4f3d343dfca552db2e65e9f306cde62f;p=515;k=515;v=3:{c=NMPOC_NEW;v={uo=6-U2151766123994976;**r=200**;p=@515@Merchant515-1269871784986-product|;o=test;**c=29/03/10 16:53:28**;}}

*** When the cancellation comes into effect:**

The code used for this notification is: **r=200**.
This notification gives in the “comments” field (c=), the **date and time** when the unsubscription came into effect.

http://www.w-ha.com/kit/responder.php?m=h=4f3d343dfca552db2e65e9f306cde62f;p=515;k=515;v=3:{c=NMPOC_NEW;v={uo=6-U2151766123994976;**r=201**;p=@515@Merchant515-1269871784986-product|;o=test;**c=05/04/10 16:34:28**;}}

Note: You must previously communicate to w-HA the URL for the “responder” servlet

CASE 2: Unsubscription through the CCA, the MSCA or the pos_request Servlet of the service provider

A notification will be sent by W-ha to the **servlet Responder** of the service provider (server to server request) **only when the unsubscription will come into effect.**

The comment field (c=) is used to give the date and time when the notification came into effect.

http://www.w-ha.com/kit/responder.php?m=h=994e932713c6e621fa57dc3039cfd38a;p=515;k=515;v=3:{c=NMPOC_NEW;v={uo=6-U8149894128200400**;r=110**;p=@515@acces_limite2|;o=acces_limite_2;c=**08/06/10 12:35:06**;}}

Each notification received by the service provider has a resiliation code **(r=)** giving the reason of the unsubscription, cf the following table:

Code (r=)	Comments
100	Subscription cancellation via the CSD (ISP Customer Service Department) reason = interrupted connection or incomplete download
101	Subscription cancellation via the CSD (ISP Customer Service Department) reason = faulty item OR **Subscription not confirmed in 24 hours**
102	Cancellation at the end of the subscription validity (license expiration)
103	Cancellation connected to subscription invoicing failure
104	Cancellation following user account closure by the operator (via the provisionning bus)
105	Cancellation following user account closure by the CSD (ISP Customer Service Department tool)
106	Cancellation following non-payment of AAR by invoice
107	Cancellation due to inability to invoice the user
110	Subscription cancellation via the CSD (ISP Customer Service Department) reason = other
111	Cancellation asked by the internet user (multiple command)
112	Cancellation asked by the internet user (ticket invoiced many times)
113	Cancellation asked by the internet user (undelivered service)
114	Cancellation asked by the internet user (contested purchase)
115	Cancellation asked by the internet user (child purchase)
116	Cancellation asked by the internet user (transaction linked to a canceled subscription or an unwanted subscription)
117	Subscription terminated: I would like to be refund
118	The service does not correspond to my expectations
119	I subscribed many times to this single offer
200	Termination of the subscription **demanded** by the internet user (via the customer area of his internet provider) (USCA)
201	Termination of the subscription **effective** (via the customer area of his internet provider) (USCA)

3.2.4.2 - Example of calling the servlet

When a subscription cancellation occurs, the URL called by the w-HA platform is in the following form (encoded URL):

```
http%3A%2F%2Fwha%2Emarchand%2Ecom%2Fbundle%2Dresponder%2Fresponder%3Fm%3Dh%3Df5938cd
e8dd6bfcb67da921b4b8bf350%3B%0D%0Ap%3D10%3Bk%3D10%3Bv%3D3%3A%7Bc%3DNMPOC%5FNEW
%3Bv%3D%7Buo%3D3D6%2DU7141248844587211%3Br%3D103%3Bp%3D%4010%40P1%7C%4010%40P2
%7C%4010%40P4%7C%3B%0D%0Ao%3DCR005%3Bc%3DCancelled+due+to+charge+processing+functional+
failure%3B%7D%7D%0D%0A
```

URL decoded:

```
http://wha.marchand.com/bundle-responder/responder?
m=
h=f5938cde8dd6bfcb67da921b4b8bf350;
p=10 ;
k=10 ;
v=3 :{
c=NMPOC_NEW;
v={
uo=6-U7141248844587211;
r=103;
p=@10@P1|@10@P2|@10@P4|;
o=CR005;
c=Cancelled due to charge processing functional failure;}}
```

This URL is secure (https) or not (http) and signed.

Information included in this URL are:

- a hmac to ensure message integrity (h=),
- the “NMPOC_NEW” (c=) command
- the subscription identifier that has just been cancelled (uo=)
- the cancellation code (r=)
- the offer affected by the cancellation (o=)

[Optional] Development of a « responder » : precision on the hmac calcul

If the content provider wish to develop his own responder using w-HA protocol explained above, he needs to calculate a hmac in order to ensure message integrity.

The hmac is calculated as bellow:

The “termination notification” message is:

m=h=f5938cde8dd6bfcb67da921b4b8bf350;p=10 ;k=10 ;v=3 :**{**
c=NMPOC_NEW;v={uo=6-U7141248844587211;r=103;p=@10@P1|@10@P2|@10@P4|;o=CR005;c=Cancelled due to charge processing functional failure;}
}

The hmac (h=) is calculated as bellow:

h= hmac_MD5("**c=NMPOC_NEW;v={uo=6-U7141248844587211;r=103;p=@10@P1|@10@P2|@10@P4|;o=CR005;c=Cancelled due to charge processing functional failure;}**","*Value_of_the_KeyValue*")

where hmac_MD5 is the calculation function of the HMAC/MD5.

A description of the algorithm HMAC/MD5, including examples, is available at the URL :
http://www.ietf.org/rfc/rfc2104.txt?number=2104

An example of calcul of HMAC/MD5 in php is available at the paragraph:
§4.8 - Content protection (using the hmac)

3.2.4.3 – Servlet functioning

The “responder” servlet works in 3 modes:

The « Logs » mode
The parameter <**urlRedirect**> in /bundle-responder/WEB-INF/merchants.xml **is NOT filled in**.
The parameter <**logFileName**>, in /bundle-responder/WEB-INF/merchants.xml, **is filled in**.

The « Redirection » mode
The parameter <**urlRedirect**>, in /bundle-responder/WEB-INF/merchants.xml, **is filled in**.
The parameter <**logFileName**>, in /bundle-responder/WEB-INF/merchants.xml, **is NOT filled in**.

The « Redirection+Logs » mode
The parameter <**urlRedirect**>, in /bundle-responder/WEB-INF/merchants.xml, **is filled in**.
The parameter <**logFileName**>, in /bundle-responder/WEB-INF/merchants.xml, **is filled in**.

When the servlet received a termination notification, it does following actions:

- writing information of the termination in the « log_notification_XXX.txt » file
  (Parameter « logFilename » in /bundle-responder/WEB-INF/merchants.xml)
- redirection to the « urlRedirect » URL
  (Parameter « urlRedirect » in /bundle-responder/WEB-INF/merchants.xml)

3.2.4.4 - Parsing of the « log_notification_XXX.txt » file

The content provider should regularly analyze the created log files (log_notifications_5XXX.txt),extract subscription identifier (uoid) which are terminated and update his database.

3.2.4.5 - Redirection of termination notifications to an URL

If the parameter « urlRedirect » in /bundle-responder/WEB-INF/merchants.xml is filled in, information about termination notification **will be send to this URL** which must be an executable (servlet, cgi, asp, php, …)

The URL called by the Kit is in the following form (encoded URL):

```
http://wha.marchand.com/notification_resiliation.php?hmac%3Df5938cde8dd6bfcb67da921b4b8bf350%26c%3DCommentaire%2Bpar%2Bdefaut%26o%3DO1%26p%3D%405XXX%40P1%26r%3D110%26uo%3D6-U5112074834229514
```

URL decoded:

```
http://wha.marchand.com/notification_resiliation.php?
hmac=f5938cde8dd6bfcb67da921b4b8bf350
&c=Commentaire+par+defaut
&o=O1
&p=@5XXX@P1
&r=110
&uo=6-U5112074834229514
```

3.2.4.6 - Response from the servlet to w-HA : Acknowledgement receipt

In « Logs » mode (parameter <***urlRedirect***> is not filled in):
The **responder** send automatically an acknowledgment « ack » to the w-HA platform.

In « Redirection » mode (parameter ***urlRedirect***> is filled in):
The **content provider** send an acknowledgment « ack » to the w-HA platform.

The “program” parametered in <urlRedirect> must do this acknowledgment. This “program” is also in charge of getting termination notification and updating in real time the content’s provider database.

The acknowledgment receipt is as bellow:

```
h=a45b921b4321fcb67da921b4b8bf350;p=5XXX;k=5XXX;v=3:{c=ack}
```

Example in php to send an acknowledgment (ack) to w-HA (termination notification):

```
<?php
//Definition of a function which caculates hmac
   function hmac($string, $key)
   {
      $byte = 64; // byte length for md5
      if (strlen($key) > $byte) {
         $key = pack("H*",md5($key));
      }
      $key  = str_pad($key, $byte, chr(0x00));
      $in_pad = str_pad('', $byte, chr(0x36));
      $out_pad = str_pad('', $byte, chr(0x5c));
      $k_in_pad = $key ^ $in_pad ;
      $k_out_pad = $key ^ $out_pad;

      return md5($k_out_pad  . pack("H*",md5($k_in_pad . $string)));
   }

//Ceation of the message (ack) to send to w-HA
$hmac = hmac("c=ack","your Keyvalue");
$message = "h=".$hmac.";p=your Mctid;k=your Keyid;v=3:{c=ack}";
echo $message;
?>
```

Information contained in the acknowledgment of the termination notification are:

- h= : the hmac to ensure message integrity
- p= : the value of the parameter « mctId » of the « merchants.xml » file (cf. § 0)
- k= : the value of the parameter « mctKeyId » of the « merchants.xml » file (cf. § 0)
- v=3 : the version of the format of the request
- c=ack : the acknowledgment receipt {c=ack}

Values in bleue, must be replaced by merchant parameters of the content provider.

The message (acknowledgment of the termination notification) sent is:

h=a45b921b4321fcb67da921b4b8bf350;p=10;k=10;v=3:{c=ack}

The hmac (h=) is calculated as bellow:

h=hmac_MD5("c=ack","VALUE_OF_THE_MCTKEYVALUE")

Where hmac_MD5 is the function of calcul of the HMAC/MD5 and VALUE_OF_THE_MCTKEY is equal to the value of the parameter « MctKey » of the « merchants.xml » file (cf. §7.2.2 - Configuration file « merchants.xml »).

A description of an algorithm HMAC/MD5, including examples, is available at the URL:
http://www.ietf.org/rfc/rfc2104.txt?number=2104

An example of calcul of HMAC/MD5 in php is available at the paragraph:
§4.8 - Content protection (using the hmac)

3.2.4.7 - In case the Content Provider "responder" servlet does not respond

If the w-HA platform does not receive a cancellation acknowledgement receipt, it will activate new attempts: in total 4 attempts with an hour interval.

Furthermore, all the notifications that failed after 4 attempts will be stored **on the w-HA platform : notifications are buffered** while waiting for the next attempt to send.

The next attempt occurs after a new notification is successfully sent.
In this case, all the awaiting notifications are present.

Please note that buffered notifications are not automatically unblocked once the Content Provider platform is operational once again. Actually, while the w-HA platform does not receive any new notification requests, the Content Provider platform status will not be tested and the notifications will therefore not be resent.

4 - What the merchant must implement to use Kit v3.5

4.1 - Creation of offers and products via the MSCA

The first stage consists of connecting to the Web interface (MSCA) enabling the Content Provider to create offers and products on the w-HA platform.

This web interface can be found at the following URL: https://mps.w-ha.com/app-msca/

The content provider must **create** at least 1 **ADMNISTRATOR account** (he chooses his login and password) at his first connection to the MSCA. Once this first ADMINISTOR account is created, the content provider **must not** use the login and the password given by w-HA.

There are 3 account types:

ADMINISTRATOR: Access to the 3 sections: "Offers"/"Activities"/"Users"

The administror account manages other users.
It can create and/or delete other accounts: Administrator, Client Service, Webmaster.

CLIENT SERVICE: Access only to the section: "Activities"

WEBMASTER : Access only to the section: "Offers"

4.2 - "*.xml" configuration file settings

The Content Provider will have to set the configuration files of the « pos_bundle » servlet

web.xml
The "web.xml" file should only have to be set once by the Content Provider before going into production.

merchants.xml
The "merchants.xml" file will be modified by the Content Provider each time a new shop is opened (merchantId) using the Kit v3.5 subscription function.

Productbundle_XXX.xml
The "productbundle_XXX.xml" file will be modified by the merchant at each product addition, modification or deletion **coherently with the offers and products created on the MSCA**.

Note: For each **modification** carried out in the "web.xml" or "products.xml" files (or even "server.xml"), it is necessary to stop and **restart the Servlet engine** in order to acknowledge the change.

4.3 - Subscription confirmation

If the Content Provider decides to manage a delayed supply confirmation (autoConfirm= false) he must:

- either **develop the "confirm" request to the w-HA platform** himself using API Java as supplied by w-HA (see Annex I)
- either **develop a client http calling the "pos_request" servlet (action= offerConfirm)**

4.4 - Using the "responder to receive "push" type info

If the Content Provider decides to use the "responder" servlet, part of Kit v3.5, he must:

In "logs" mode:
Regularly analyze the created log files (log_notifications_5XXX.txt) in order to update his subscription database.

In "redirection" mode:
Get "termination notification" using the urlRedirect parameter (in bundle-responder/WEB-INF/merchant.xml) and send an acknowledgement (c=ack) to the w-HA platform for each "termination notification" recieved.

4.5 - Creating offer presentation web pages

The Content Provider will show various subscription offers on his website:

authorizeOffer(O1) authorizeOffer(O2) authorizeOffer(O3)

Cette formule offre un accès illimité à = this package offer unlimited access to
Mon Site Web = My Website
Abonnement hebdomadaire à tacite reconduction, sans engagement de durée, et résiliable à tout moment = Weekly subscription with tacit renewal, no length committement and may be cancelled at any time
mensuel = monthly
trimestriel = quarterly
Suite = Next

4.6 - Display Internet+ web pages in “current page”

Content provider side :

Due to browser evolution (Internet Explorer, Firefox, Google Chrome), and protection services (antivirus, pop-up killer, ….) more and more efficient, you **must display Internet+ web pages (choice of the Internet Service Provider, authentication, billing panel) in “current page”.**

From the billing panel, on your web site, the html code you have to use to call the billing servlet w-HA (pos_bundle) becomes:

```
<a href="/bundle/pos_bundle?action=authorizeOffer&oid=O1&wha_desc2=current">
<img src="../images/bt_validate_gd.jpg" border="0">          (1)
</a>
```

A classical *href* link is used.

The following html code could also be used:

```
<input type="image" src="../button_payment_internetplus.gif”
OnClick="window.location.href='
/bundle/pos_init?action=authorizeOffer&oid=O1&wha_desc2=current">     (1)
```

The method *location.href* allow the opening of the billing panel w-HA in the current page.
However, in that implementation, a javascript code is used to display screens (function *OnClick* and *location.href*).

The Kit redirects the internet user to"route.w-ha.com" with "wha_desc2=current" as additional parameter. Those parameters appear in merchant properties (mp) and "_ap_" is added before the parameter so we can read in mp: "_ap_wha_desc2=current"

Example :
https://route.w-ha.com/app-bundlepurchase/node?m=h=9db95d8f245ab1ceed02d114ffdbb45e;p=5XXX;k=5XXX;v=3:{c=OfferAuthorizeReq;v={mUrl=http://localhost:8080/bundle/pos_bundle;oid=O5;mp={_ap_userId=fmoreau;**_ap_wha_desc2=current;**cur=EUR;amt=0.01;};}}

w-HA platform side :

The w-HA server does a control on the parameter wha_desc2=current, to optimize the billing cinematic in « current page » mode.

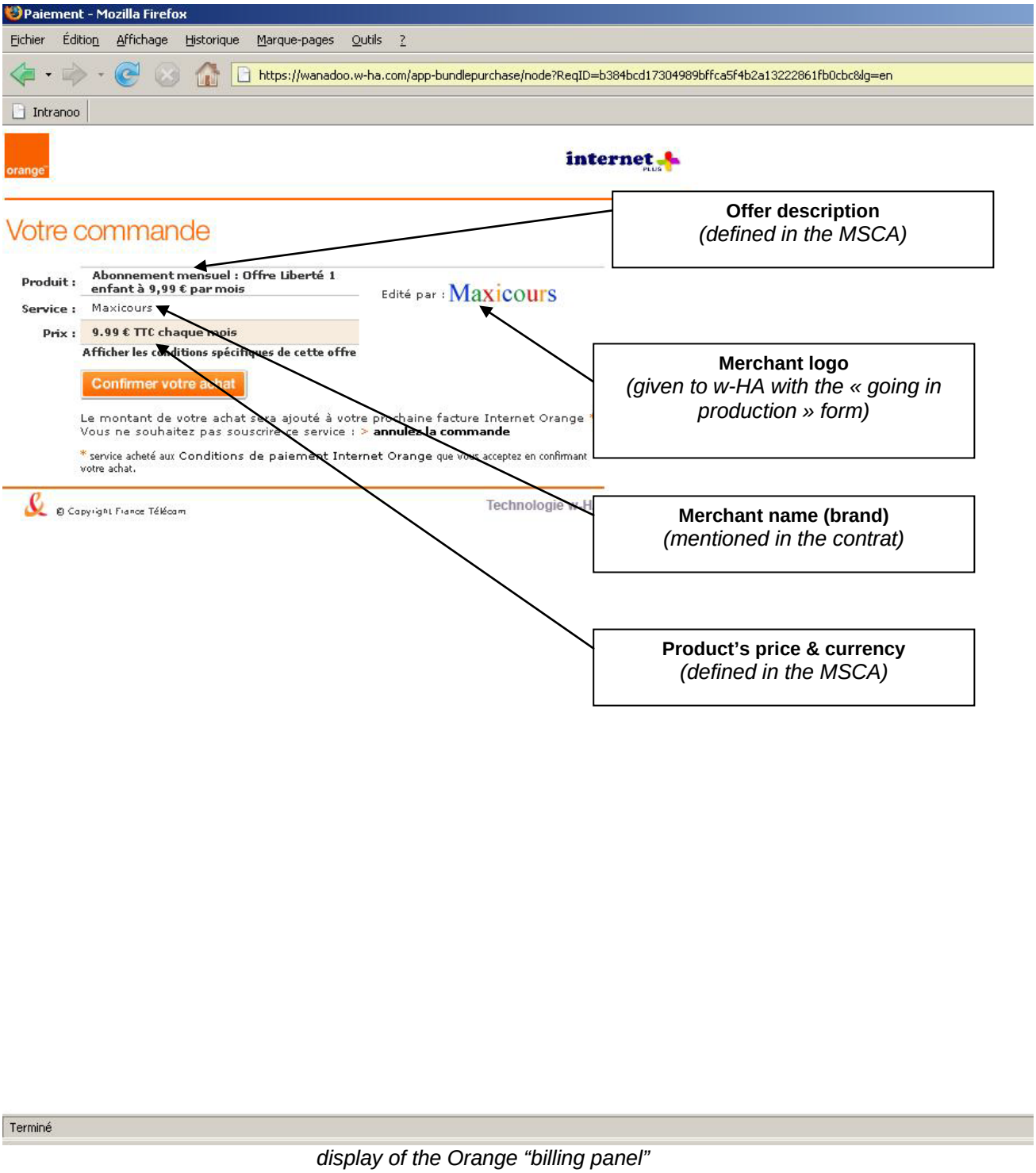
So the merchant must associate the “*wha_desc2=current*” parameter to a billing cinematic in *current page*.

Warning:
You have to check that the new parameter "wha_desc2=current" is used in the hmac calculation when the client is redirected to :

- the **fulfillmenturl** (url parametered in tomcat/webapps/bundle/WEB-INF/prodcutbundle_5XXX.xml)
- the **mcttrxCancelFromPaymentPanelUrl** (url parametered in tomcat/webapps/bundle/WEB-INF/merchant.xml)

(1) The file « ../images/bouton_paiement_internetplus.gif » contains the picture of the internet+ billing button, which should be displayed on the merchant web site.
All usable buttons are available on www.internetplus.fr > « tool box »

Example of the Orange billing panel in current page:



display of the Orange "billing panel"

4.8 - Content protection (using the hmac)

All information exchanged between the service Content Provider platform and the w-HA will be **signed** (HMAC-MD5) which will **guarantee platform authentification and the integrity of exchanged information**.

The electronic signature is based on a **shared symmetrical crypt encryption key** by the merchant platform and that of the w-HA: the **mctKey (or KeyValue)**.

Once this information is sent, the “A” platform uses an algorithm to create a MAC (Message Authentification Code) based on the symmetrical key and its information, which will then be transmitted simultaneously to platform “B” in the URL.

Upon receipt of this information and the MAC, platform “B” uses the same algorithm to created another MAC based on the same symmetrical key and its information.

If it is identical to the one read, no changes to the information has been made: the transaction is authorized.
If it is different to the one read, the information has been modified: the transaction is denied.

To prevent the Internet User from using a service for free and accessing it directly through its URL (fulfillmentUrl), the following must be checked:

- he integrity of additional settings (calculating a HMAC)
- the value of one or several additional settings (ex: session identifier)

The HMAC is created using a HMAC MD5 algorithm with the following settings:

- the settings collected in the fulfillmentUrl (as a unique “string”),
- the value of the “mctKey” (or KeyValue) setting for the “merchants.xml” (as a unique “string”),

For example, if the return to the end URL (fulfillment Url) is done in the following manner:

http://marchand.com/payant/paiement_ok.php?sessionId=1234&commandeId=abcd&userId=toto&hmac=891284e23faa662c033a41dd9905cc10&uoid=U5151292477228013

The program (here “paiement_ok.php”) must check the integrity of settings by calculating the HMAC relative to these settings prior to checking their value and displaying the purchased product/service (or by refusing access if necessary).

If the “mctKey” setting value is “a1b2c3d4e5”, then the HMAC calculation id done in the following manner:

my-hmac = H-MAC ("**commandeId=abcd&sessionId=1234&uoid=6-U5151292477228013&userId=toto**","**a1b2c3d4e5**")

Caution 1:
The additional settings (or “merchant properties” or “mp”) sent to the w-HA servlet must not contain any accents or special characters.

The URL encoding of these characters will result in an error in the HMAC calculation and in this case the product/service delivery will not be carried out even though the end user will be debited.

Caution 2:
The order of settings for consideration for the HMAC calculation by the w-HA application is an **alphabetical order**.

Caution 3:
The subscription identifier (uoid), collected on the fulfillementUrl, **must also be taken into account when re-calculating the hmac.**

A calculation example for HMAC in PHP:
If the fulfillment URL is:
http://marchand.com/payant/paiement_ok.php?sessionId=1234&commandeId=abcd&userId=toto&hmac=891284e23faa 662c033a41dd9905cc10&uoid=6-U5151292477228013

The hmac has to be calculated this way:

```
$hmac_verif = bin2hex(mhash(MHASH_MD5,"commandeId=abcd&sessionId=1234& uoid=6-U5151292477228013&userId=toto","a1b2c3d4e5f");
        if ($hmac_verif == $hmac)
        print "ok"; else
        print "no";
```

Check of the value of one (or many) additional parameters.

In order to prevent internet user to use many times the fulfillment (ffl) (and doing it access freely to the product/service), **the content provider should check that the "delivery" of a specific transaction has not been done yet**.

To do it the content provider can use the subscription identifier (uoid) which is unique or an additional parameter of his choice (calling the servlet for subscription authorization demand, see 3.1.1 - Subscription demand / « pos_bundle?action=authorizeOffer ») to check the delivery URL (ffl) uniueness.

When the ffl is called the content provider can check that unique parameter (uoid or one parameter of his choice) defines a "delivery which was already done or not.

For the first call of the URL, the content provider redirects the client to the ffl.
If there are other calls, the content provider blocks the access to the ffl and display an error message.

5 – ANNEXE I : SDK

5.1 - Developing a subscription “confirm” via the SDK

If the service Content Provider want to activate the “confirm” (or the “cancel”) of the subscription after ensuring that it has been acknowledged (or not) in the database, he must use the API Java contained in the SDK.

A Javadoc (“javadoc.zip” is available to the Content Providers who request it from their w-HA technical support.

The service Content Provider must use the following Java classes:

- sdk-valista0.jar
- CORE-API-1-18-sp0r3-MERCHANT-SDK-lib.jar

5.1.1 - Implementation

Implementation is carried out in the following manner:

The Java class for construction must in inherit the following class:

Proxy

The methods for implementation for confirmation or cancellation are:

CancelOffer

```
public void cancelOffer(java.lang.String nodeResponderURL,
                        long merchantId,
                        long keyId,
                        java.lang.String userOfferId)
                throws com.valista.core.api.node.pos.MerchantOfferManager.CancellationException,
                java.rmi.RemoteException
com.valista.core.api.node.pos.MerchantOfferManager.CancellationException
java.rmi.RemoteException
```

ConfirmOffer

```
public void confirmOffer(java.lang.String nodeResponderURL,
                         long merchantId,
                         long keyId,
                         java.lang.String userOfferId)
                throws com.valista.core.api.node.pos.MerchantOfferManager.ConfirmationException,
                com.valista.core.api.node.pos.MerchantOfferManager.ReplacementException,
                java.rmi.RemoteException
com.valista.core.api.node.pos.MerchantOfferManager.ConfirmationException
com.valista.core.api.node.pos.MerchantOfferManager.ReplacementException
java.rmi.RemoteException
```

5.1.2 – Code example

```
Proxy proxyValista = new Proxy(keyTable) ;

//CancelOffer
try
{
proxyValista.cancelOffer(nodeURL, merchantId, keyId, uoid);
forwardToMessageURL(request, response, “CancelSuccess.UserOfferId: “ + uoid);
}
catch(com.valista.core.api.node.pos.MerchantOfferManager.CancellationException e)
{
forwardToMessageURL(request, response, “Cancel for” + uoid + “failed<br/>” +
 e.toString());
}

//ConfirmOffer
try
{
proxyValista.confirmOffer(nodeURL, merchantId, keyId, uoid);
forwardToMessageURL(request, response, “ConfirmSuccess.UserOfferId: “ + uoid);
}
catch(com.valista.core.api.node.pos.MerchantOfferManager.ConfirmationException e)
{
forwardToMessageURL(request, response, “Confirm for” + uoid + “failed<br/>” +
 e.toString());
}
catch(com.valista.core.api.node.pos.MerchantOfferManager.ReplacementException e)
{
forwardToMessageURL(request, response, “Replacement Confirmation failed for” + uoid +
“failed<br/>” +  e.toString());
}
```

5.2 - Development of a specific “responder” via the SDK

If the service Content Provider wants to create its own “responder” in order to collect cancellation information and include them directly into his database, he must use the API Java contained in the SDK sdk-valista.jar.

A Javadoc (“javadocResponder.zip”) is available to the Content Providers who request it from their w-HA technical support.

The service Content Provider must make his Java class inherit the following APIs:

- core-api-merchant-sdk-lib.jar
- core-util-common.jar

5.2.1 - Implementation

The implementation is carried out in the following manner:

The Java class for construction must in inherit the following class:

AbstractNotificationManagerCommand :

Type de commande traitée :

NMPOC_NEW: NotificationManagerPushOfferCancellationData(data);

La méthode à implémenter pour ce type de commande est :

pushOfferCancellation(java.lang.String userOfferId,java.lang.String offerId, com.valista.core.api.merchant.node.NotificationManager.CancelledProductInfo[] products, java.lang.String reasonCode, java.lang.String comment)

Avec les interfaces suivantes :

All Implemented Interfaces:
com.valista.command.Command, com.valista.util.manager.Manager, com.valista.core.api.merchant.node.NotificationManager, java.rmi.Remote

5.2.2 – Code example:

```
// NotificationManagerPushOfferCancellationData,

// Interface à implementer NotificationManager

if(cmdType.equals(« NMPOC_NEW »)) {

        NotificationManagerPushOfferCancellationData data =
(NotificationManagerPushOfferCancellationData)cmdData;

                String reasonCode = data.gerReasonCode();
                String comment = data.getComment();

                pushOfferCancellation(data.getUserOfferId(),data.getOfferId(),data.getCancelledProducts(),
reasonCode, comment);
        return MessageUtils.createAcknowledgementMessage(null);
}
```

Description of the NotificationManager interface:

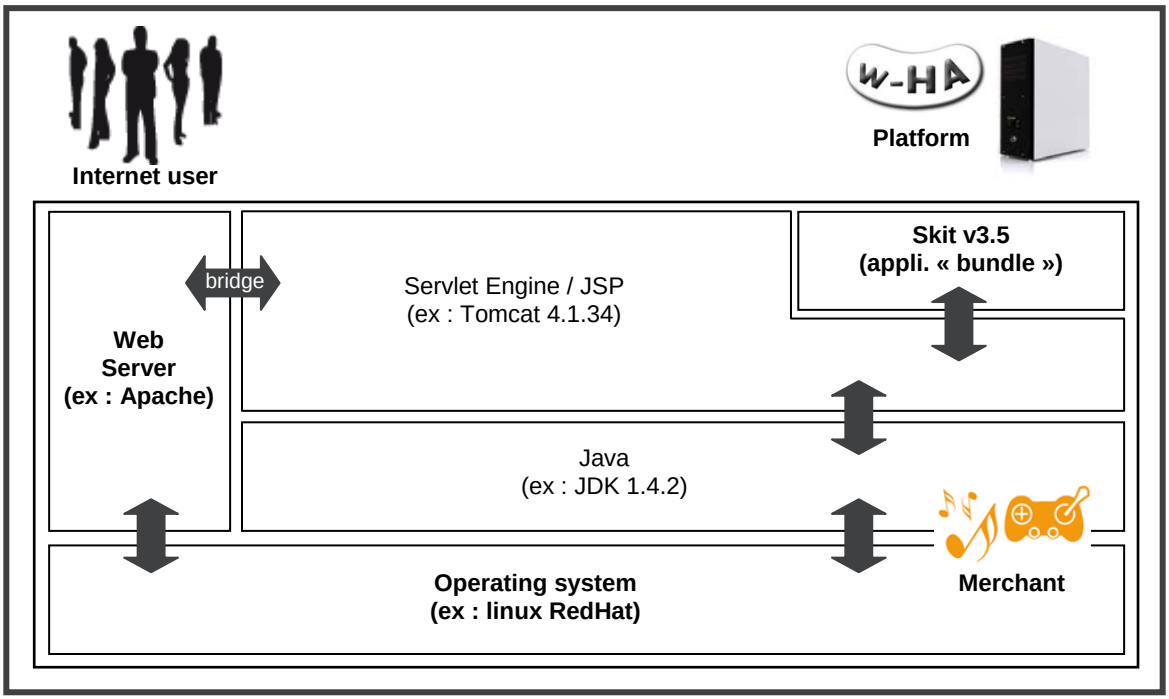
```
package com.valista.core.api.merchant.node ;
import com.valista.util.manager.Manager ;
import com.valista.message.ValistaMessageableException ;
import java.rmi.RemoteException ;
import java.io.Serializable ;

public interface NotificationManager extends Manager {
        public abstract void pushOfferCancellation(String s, String s1,
CancelledProductInfo[]cancelledProductInfos, String s2, String s3) throws RemoteException ;
}
```

6 - ANNEXE II : Technical pre-requirements for operating Kit v3.5

Implementing the w-HA solution **requires the installation of a Java "bundle" application on the merchant's server**.

The following diagram illustrates the relationships between the different software and material components necessary for operation the w-HA "bundle" application.



Components necessary to the « bundle » application

The w-HA payment system operation requires installation of the “w-HA” (Java application) application on the service Content Provider hosting platform.

6.1 - Hosting platform

The technical environment (operating system and web server) of the Content Provider’s website hosting platform must be compatible with the “w-HA” application.

This application relies on the use of Java Servlets enabling compatibility with most platforms on the market.

6.1.1 - Operating system

W-HA guarantees that Kit v3.5 will work properly and ensure technical support for the environments described below.

If the operating system is:

Sun Solaris version 2.6, version 2.7 or above, w-HA will ensure technical support for the web servers:

- Apache Web Server version 1.3.9 or above (with support for DSO modules)
  If Apache is not installed, install a recent version of 1.3.x or 2.x.

Linux Kernel version 2.2.x (glibc/libc6 must be installed) w-HA will ensure technical support for the web servers:

- Apache Web Server version 1.3.9 or above (with support for DSO modules)
  If Apache is not installed, install a recent version of 1.3.x or 2.x.

Windows NT version 4.0 (Service Pack 5 or above must be installed) w-HA will ensure technical support for the web servers:

- Apache Web Server version 1.3.9 or above (with support for DSO modules)
  If Apache is not installed, install a recent version of 1.3.x or 2.x.
- Internet Information Server version 4.x (IIS4)

Windows 2000, w-Ha will ensure technical support for the web servers:

- Apache Web Server version 1.3.9 or above (with support for DSO modules)
  If Apache is not installed, install a recent version of 1.3.x or 2.x.
- Internet Information Server version 5.x (IIS5)

Windows 2003, w-Ha will ensure technical support for the web servers:

- Apache Web Server version 1.3.9 or above (with support for DSO modules)
  If Apache is not installed, install a recent version of 1.3.x or 2.x.
- Internet Information Server version 6.x (IIS6)

For other environments (other Operating System / Web Server couples), operating the w-HA application is quite possible, but in this case the installation of Java J2SDK 1.3.x version (or above) and the creation of the “bridge” between the web server and the Servlet engine will be performed by the merchant and will be his sole responsibility.

6.1.2 - Java Virtual Machine

Java 2 SDK version 1.4.x or above must be installed prior to or during the integration.

W-HA recommends installing version "j2sdk1.4.x" (for example: j2sdk1.4.2).

Note:
If the platform is a Linux/UNIX type, the patches required for each component must be installed (for example, when installing j2sdk1.3.x on Solaris 7, the installation document specifies that patches No. xxxxx must be installed).

6.1.3 - SSL protocol management module : "JSSE"

JSSE version 1.0.2 must be installed prior to or during the integration.

Note:
JSSE is included by default in J2SDK Java versions 1.4.x and above.
Installation is therefore not necessary.

6.1.4 - Servlet engine

A Servlets/JSP engine complying with the following SUN requirements must be installed prior to or during the integration:

- **JavaServlet 2.2**
- **JavaServlets Pages (JSP) 1.1**

W-HA recommends the Servlets/JSP engine "Tomcat 4.1.x" (for example: Tomcat 4.1.34).

If the Servlets/JSP engine is Tomcat, the Web server must be one of the following:

- Apache Web Server version 1.3.9 or above (with support for DSO modules). If Apache is not installed, install a recent version of 1.3.x or 2.x
- Internet Information Server 4 - (IIS4)
- Internet Information Server 5 - (IIS5)
- **For all other Web Servers, or if the merchant's Servlet engine is not Jakarta-Tomcat**, operating the w-HA application is quite possible on the condition that the Servlet's engine complies with the SUN requirements cited above.

In this case, integrating the application within the Servlet engine is the merchant's sole responsibility. W-HA ensures technical support for the w-HA application operation and settings.

6.2 – Network Considerations

For Kit v.35 to work, some network equipment settings must be considered (port opening, firewalls, proxies)

6.2.1 - During the w-HA application installation

The w-HA application-hosting platform must be:

- **accessible** from the outside by the **existing Web Server port** (by default: port 80)
- if possible, accessible from the outside by the Servlet Engine port (by default: port 8080)
- **Can initiate a SSL communication (on port 443) with the w-HA production platform**
- The different network equipments (Firewalls, Proxies,…) must therefore be properly configured prior to installation to allow access as mentioned above.

6.2.2 - In production

The w-HA application-hosting platform must be:

- **accessible from the outside by the existing Web Server port (by default: port 80)**
- **Can initiate a SSL communication (on port 443) with w-HA platform nodes:**

Domain name	Operator	IP	Port	Request type
https://wanadoo.w-ha.com	Orange Internet	193.28.205.2	443	Outgoing
https://free.w-ha.com	Alice/Free	193.28.205.52	443	Outgoing
https://sfr.w-ha.com	SFR	193.28.205.17	443	Outgoing
http://secure.w-ha.com	Orange Internet, Alice	193.28.205.186	80	Ingoing

The different network equipments (Firewalls, Proxies, ...) must therefore be properly configured prior to installation to allow access as mentioned above.

Note:
To know the IP address of a domain name with windows (*in start>execute>cmd*) you can use the *nslookup* commande (ex: nslookup wanadoo.w-ha.com)

6.3 - Other pre-requisites for integration

6.3.1 - Decompression Utility program

As the Kit v3.5 application is delivered in "**zip**" or ".**tar.gz**" form, a decompression utility program such as **WinZip, unzip or tar is required**.

6.3.2 - Web Server re-boot

Application installation requires a modification of the server(s) configuration (Http server and Servlet Engine).
These modifications require a re-boot of the server(s), or even of the machine in some cases.

6.3.3 - System administrator presence

During the application installation and the Web server restart, the hosting platform system administrator's presence ("root" rights) is required.
(1/2 to 1 Day.)

7 - ANNEXE III : Components of the Kit v3.5

Kit v3.5 uses the following components for subscription payments:

Application « bundle »	
Servlet « pos_bundle »	Manages: - subscription requests on the w-HA platform - subscription request replies - subscription confirmation
Servlet « pos_request »	Manages: - delayed subscription confirmation - transaction refund - subscription termination
Fichier de configuration « web.xml »	Contains overall and default settings for Kit v3.5
Fichier de configuration « productbundle_XXX.xml »	Contains the Product Catalogue
Fichier de configuration « merchants.xml »	Contains settings for the different merchant shops
Un fichier de log : « log_request_XXX.txt »	Writes the information regarding subscription requests
Un fichier de log : « log_response_XXX.txt »	Writes information on subscriptions validated by the w-HA platform

Application « bundle-responder »	
Servlet « responder »	Manages information receipt from the w-HA platform ("push") for "Termination" type events
Fichier de configuration « web.xml »	Mainly contains the shop “merchantId”
Fichier de configuration « merchants.xml »	Contains parameters of the shop of the content provider
Un fichier de log des erreurs : « log_error_XXX.txt »	Writes errors information on "push" type events
Un fichier de log des notifications : « log_notification_XXX.txt »	Writes information on "push" type events

Application « demo »	
/demo/bundle/html/index.html	Enables to check if the "pos_bundle" servlet for an offer subscription works.
/demo/bundle/html/confirm_differe.html	Enables to check if the "pos_request" servlet for delayed confirmation subscription request works.
/demo/bundle/html/remboursement_transaction.html	Enables to check if the "pos_request" servlet for refund requests
/demo/bundle/html/resiliation_abonnement.html	Enables to check if the "pos_request" servlet for unsubscription requests.

Note: When the content provider completes settings in ".xml" files, he can add **only one parameter in urls** defined.

Here is the list of urls to be set in ".xml" files :

» in tomcat/wepapps/bundle/WEB-INF/web.xml :
- merchantHomeUrl
- productUnavailableUrl

» in tomcat/wepapps/bundle/WEB-INF/productbundle_5XXX.xml :
- fulfillmentUrl

» in tomcat/wepapps/bundle/WEB-INF/merchant.xml :
- mcttrxCancelFromPaymentPanelUrl

» in tomcat/wepapps/bundle-responder/WEB-INF/merchant.xml :
- urlRedirect

Note:
If the url contains 0 parameter:
(example of an url set in an ".xml" file :
http://wha.marchand.com/page.php)
Then at the time of the redirection to the set url, parameters are added after a "?" and are separated by "&".

If the url contains 1 parameter:
(example of an url set in an ".xml" file :
http://wha.marchand.com/page.php?parameter1=abc)
Then at the time of the redirection to the set url, parameters are added after a "&" and are separated by "&".

If the url contains 2 parameters or more:
(example of an url set in an ".xml" file :
http://wha.marchand.com/page.php?parameter1=abc&parameter2=cba)
Then there is an error 500 (IllegalArgumentException) because the Kit can not read the url.

In order to avoid theses error due to settings, content providers should add **additional parameters** when they **call the Kit** and **not in urls set in ".xml" files**.

7.1 - Servlets « pos_bundle » and « pos_request »

7.1.1 - Configuration of the « web.xml » file

7.1.1.1 - Structure of the « web.xml » file

Parameter	Comments
« pos_bundle »	
merchantId (mctid)	**Shop identifier used by default (supplied by w-HA).** It enables the w-HA platform to identify the shop. The other settings (key.id, key.value) connected to this shop are present in the file defined by the xmlMarchadDatabase settings.
messageUrl	**Access path to JSP that manages the html format error messages (do not modify)** The different error cases are sent to the JSP "message.jsp" in the "bundle" application.
messageWmlUrl	**Access path to JSP that manages the wml format error messages (do not modify)** The different error cases are sent to the JSP "message.jsp" in the "bundle" application.
merchantCallbackUrl	**URL where the "pos_bundle" servlet is located on the merchant server (to be modified)** URL towards which the user is redirected after accepting the offer subscription on the w-HA payment panel. Adapt with the public IP address or the server public domain name where the "bundle" application is installed.
nodePaymentPanelUrl	**URL of the w-HA platform node (supplied by w-HA)** The URL for the production : https://route.w-ha.com/app-bundlepurchase/node
merchantCurrency	**Merchant currenty: EUR (do not modify)** The Euro is the only currency accepted on the w-HA platform for Internet+. The Content Provider currency must therefore by set to the EUR value. The offer amount is displayed in Euros on the w-HA payment panel.
xmlAllProductDatabase	**Access path to the "productbundle.xml" file (do not modify)** The "productbundle.xml" file contains the Content Provider's offer.
xmlMarchandsDatabase	**Access path to the "merchant.xml" file (do not modify)** The "merchant.xml" contains the settings (key.id, key.value, ...) for the different Content Provider shops.
timestampDelay	**Maximum delay for offer confirmation (in milliseconds, do not modify)** Maximum delay between the subscription request and the subscription confirmation

« pos_request »	
merchantId	**Shop identifier used by default (supplied by w-HA).** It enables the w-HA platform to identify the shop. The other settings (key.id, key.value) connected to this shop are present in the file defined by the xmlMarchadDatabase settings.
messageUrl	**Access path to JSP that manages the html format error messages (do not modify)** The different error cases are sent to the JSP "message.jsp" in the "bundle" application.
messageWmlUrl	**Access path to JSP that manages the wml format error messages (do not modify)** The different error cases are sent to the JSP "message.jsp" in the "bundle" application.
merchantCallbackUrl	**URL where the "pos_request" servlet is located on the merchant server (to be modified)** Adapt with the public IP address or the server public domain name where the "bundle" application is installed.
xmlMarchandsDatabase	**Access path to the "merchant.xml" file (do not modify)** The "merchant.xml" contains the settings (key.id, key.value, ...) for the different Content Provider shops.
timestampDelay	**Maximum delay for offer confirmation (in milliseconds, do not modify)** Maximum delay between the subscription request and the subscription confirmation

7.1.1.2 - Example of the « web.xml » file

```
<!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>

// pos_bundle //

   <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>801</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>merchantCallbackUrl</param-name>
         <param-value>http://wha.marchand.com/bundle/pos_bundle</param-value>
      </init-param>
      <init-param>
         <param-name>nodePaymentPanelUrl</param-name>
         <param-value>https://route.w-ha.com/app-bundlepurchase/node</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/productbundle_801.xml;/WEB-INF/produtcbundle_802;</param-value>
         </init-param>
         <init-param>
            <param-name>xmlMarchandsDatabase</param-name>
            <param-value>/WEB-INF/merchants.xml</param-value>
         </init-param>
         <init-param>
            <param-name>timestampDelay</param-name>
            <param-value>300000</param-value>
         </init-param>
   </servlet>
```

// pos_request //

```
<servlet>
      <servlet-name>MerchantConfirmWHAServlet</servlet-name>
      <servlet-class>com.wha.core.merchant.simulator.MerchantConfirmWHAServlet</servlet-class>
      <init-param>
         <param-name>merchantId</param-name>
         <param-value>801</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>merchantCallbackUrl</param-name>
         <param-value>http://wha.marchand.com/bundle/pos_confirm</param-value>
      </init-param>
      <init-param>
         <param-name>xmlMarchandsDatabase</param-name>
         <param-value>/WEB-INF/merchants.xml</param-value>
      </init-param>
      <init-param>
         <param-name>timestampDelay</param-name>
         <param-value>300000</param-value>
      </init-param>
</servlet>

<servlet>
   <load-on-startup></load-on-startup>
</servlet>

   <servlet-mapping>
      <servlet-name>MerchantServlet</servlet-name>
      <url-pattern>/pos_bundle</url-pattern>
   </servlet-mapping>

<servlet-mapping>
            <servlet-name>MerchantConfirmWHAServlet</servlet-name>
            <url-pattern>/pos_request</url-pattern>
</servlet-mapping>

</web-app>
```

7.1.2 - Configuration of the « productbundle_XXX.xml » file

7.1.2.1 - Structure of the « productbundle_XXX.xml » file

Parameter	Comments
Products	
productId	**(NOT USED)** *Note : this field must not empty.*
fulfillmentUrl	**(NOT USED)** *Note : this field must not empty.*
purchaseCase	**(NOT USED)**
autoConfirm	**(NOT USED)** *Note : this field must not empty.*

Offer	
offerId	**Offer identifier (as defined in the MSCA)** Unique offer identifier used by the "pos_bundle" servlet to perform subscription requests (redirection)
fulfillmentUrl	**URL for subscription acknowledgement (to be modified)** The Content Provider URL towards which the user is redirected after accepting the offer subscription.
purchaseCase	**(NOT USED)**
autoConfirm	**Automatic access confirmation (by default: true)** - **true** : the "pos_bundle" servlet automatically confirms access then redirects the subscriber to the fulfillementUrl - **false** : The "pos_bundle" servlet redirects the subscriber to the fulfillmentUrl. The Content Provider has 24 hours to confirm access to the w-HA platform

7.1.2.2 - Example of the « productbundle_XXX.xml » file

```
<?xml version='1.0'?>
<valista-product-database>

        <info>WEB Merchant Product Database</info>

        <product-list>
                <product>
                <productId>P1</productId>
                <fulfillmentUrl>http://www.marchand.com/homepage/acces.php </fulfillmentUrl>
                <purchaseCase></purchaseCase>
                <autoConfirm>true</autoConfirm>
                </product>
        </product-list>

        <offer-list>
                <offer>
                <offerId>O1</offerId>
                <fulfillmentUrl>http://www.marchand.com/abonnement/souscription.php</fulfillmentUrl>
                <purchaseCase></purchaseCase>
                <autoConfirm>true</autoConfirm>
                </offer>
                <offer>
                <offerId>O2</offerId>
                <fulfillmentUrl>http://www.marchand.com/abonnement/souscription.php</fulfillmentUrl>
                <purchaseCase></purchaseCase>
                <autoConfirm>true</autoConfirm>
                </offer>
                <offer>
        </offer-list>

</valista-product-database>
```

7.1.3 - Configuration of the « merchants.xml » file

7.1.3.1 - Structure of the « merchants.xml » file

Parameter	Comments
mctId (merchantId)	**Merchant ID (supplied by w-HA)** Allows the w-HA platform to identify the Merchant.
mctKeyId (keyId)	**Secret key ID (supplied by w-HA)** The key value is shared between Kit v3.5 and the w-HA platform. It is connected to the "mctKey" and ensures information integrity.
mctKey (keyValue)	**Secret key value (supplied by w-HA)** The key value is shared between Kit v3.5 and the w-HA platform. It is connected to the "mctKeyId" and ensures information integrity.
mcttrxCancelFromPaymentPanelUrl	**URL to redirect the client if he clicks on the cancel button on the billing panel.**
requestFileName	**Access path *(absolute)* to the "log_request_5XXX.txt" file.**
responseFileName	**Access path *(absolute)* to the "log_response_5XXX.txt" file.**

7.1.3.2 - Example of the « merchants.xml » file

```
<?xml version='1.0'?>
<valista-key-database>
<info>WEB Merchant key Database</info>
        <mct-list>
        <mct>
                <mctId>801</mctId>
                <mctKeyId>801</mctKeyId>
                <mctKey>Key for 801</mctKey>
                <mcttrxCancelFromPaymentPanelUrl>
                http://www.marchand1.com/abonnement/refus_paiement.php</mcttrxCancelFromPaymentPanelUrl>
                <requestFileName>/usr/local/tomcat/webapps/bundle/WEB-
                INF/log_request_801.txt</requestFileName>
                <responseFileName>/usr/local/tomcat/webapps/bundle/WEB-
                INF/log_response_801.txt</responseFileName>
        </mct>
        <mct>
                <mctId>802</mctId>
                <mctKeyId>802</mctKeyId>
                <mctKey>Key for 802</mctKey>
                <mcttrxCancelFromPaymentPanelUr>
                http://www.marchand2.com/abonnement/refus_paiement.php</mcttrxCancelFromPaymentPanelUrl>
                <requestFileName>/usr/local/tomcat/webapps/bundle/WEB-
                INF/log_request_802.txt</requestFileName>
                <responseFileName>/usr/local/tomcat/webapps/bundle/WEB-
                INF/log_response_802.txt</responseFileName>

        </mct>
        </mct-list>
```

7.1.4 - Log file « log_request_XXX.txt »

Access path to the **log file « log_request_XXX.txt »** is defined in the « merchants.xml »file of the Kit v3.5.

It is an absolute path.
By default : [TOMCAT_HOME]/webapps/bundle/WEB-INF/log_request_XXX.txt

It is a text file (*.txt) which is updated **(a new line is added) for each going out request (answer to a subscription)** received by the servlet « pos_bundle ».

Caution !

If many requests are received, the size of this file may be important (several MB) and it might lower the performances' application.
So the setting up of rotating logs advised.

7.1.4.1 - Structure of the «log_request_XXX.txt » file

Parameter	Description
Date	Date of the going out request (sent by the KITv3.5 to the w-HA platform)
Identifier (Subscription)	When the request is sent, the w-HA platform did not allocate an offer ID (uoid) to the subscription. This field is always setted to : **NULL**
Status (Subscription)	When the request is sent, the w-HA platform did not allocate an offer ID (uoid) to the subscription. This field is always setted to : **3**
URL	Possible values : - URL of the w-HA platform to redirect the client (**https://*InternetProvider*.w-ha.com/app-bundlepurchase/node**)
Merchant Identifier	Merchant ID (defined in the « merchants.xml » file
Identifier (of the Offer)	Possible values : - Offer ID concerned (defined in the « productbundle_5XXX.xml » file) (ex : **O1**)
Merchant Propertie #1	Additional parameter from the content provider (#1)
…..	
Merchant Propertie #N	Additional parameter from the content provider (#N)
Purchase	Possible values : - Subscription to an offer : **2**

7.1.4.2 - Example of the « log_request_XXX.txt » file

Each new subscription request adds a new line containing different fields separated by the "Pipe" character ('|'):

2008-04-14 14:38:15|NULL|3|https://route.w-ha.com/app-bundlepurchase/node|515|O1|NULL|_ap_userId=abcd|_ap_sessionId=1234|ts=2008-04-14 14:38:15.25|2|
2008-04-14 15:54:26|NULL|3|https://route.w-ha.com/app-bundlepurchase/node|515|O4|NULL|_ap_userId=abcd|_ap_sessionId=1234|ts=2008-04-14 15:54:26.093|2|

7.1.5 - Log file : « log_response_XXX.txt »

Access path to the **log file « log_response_XXX.txt »** is defined in the « merchants.xml »file of the Kit v3.5.

It is an absolute path.
By default : [TOMCAT_HOME]/webapps/bundle/WEB-INF/log_response_XXX.txt

It is a text file (*.txt) which is updated **(a new line is added) for each entering request (answer to a subscription)** received by the servlet « pos_bundle ».

Caution !

If many requests are received, the size of this file may be important (several MB) and it might lower the performances' application.
So the setting up of rotating logs advised.

7.1.5.1 - Structure of the «log_response_XXX.txt » file

Parameter	Description
Date	Date of the entering request (received by the KITv3.5 from the w-HA platform)
Identifier (of the Offer)	Possible values : - Subscription : offer ID *(uoid)* (**6-UXX….XXXX**) - Cancellation from the client on the billing panel (**void**)
Status (of the Offer))	Possible values : - **Cancel** - **Confirm** - Waiting for the confirmation (**NOT AUTO CONFIRM**)
URL	Possible values : - Cancellation from the client on the billing panel : URL to redirect the client (**trxCancelFromPaymentPanelUrl**) - Unconfirmed offer : URL to redirect the request « confirm » (**https://*operateur*.w-ha.com/app-node-mct/responder**)
Merchant Identifier	Merchant ID (defined in the « merchants.xml » file
Identifier (of the Offer)	Possible values : - Offer ID concerned (defined in the « productbundle_5XXX.xml » file) (ex : **O1**)
Merchant Propertie #1	Additional parameter from the content provider (#1)
…..	
Merchant Propertie #N	Additional parameter from the content provider (#N)
Purchase	Possible values : - Subscription to an offer : **2**

7.1.5.2 - Example of the « log_response_XXX.txt » file

Each new subscription response adds a new line containing different fields separated by the "Pipe" character ('|'):

2008-04-11 10:30:25|void|Annulation|http://192.168.1.8/demo/bundle/html/panel_cancel1.html|5231|void|_ap_userId=fmoreau|ts=2008-04-11 10:30:11.671|cur=EUR|amt=0.01|2|2007-10-04 11:17:19|6-2008-04-14 15:14:24|6-U4142633227858431|Confirm|https://wanadoo.w-ha.com/app-node-mct/responder|515|O1|_ap_userId=abcd|_ap_sessionId=1234|ts=2008-04-14 15:14:08.656|cur=EUR|amt=0.01|2|2008-04-14 15:46:13|6-U5117575881274524|NOT AUTO CONFIRM|https://wanadoo.w-ha.com/app-node-mct/responder|515|O4|_ap_userId=abcd|_ap_sessionId=1234|ts=2008-04-14 15:45:59.515|cur=EUR|amt=0.01|2|

7.2 - Servlet « responder »

7.2.1 - Configuration of the « web.xml » file

7.2.1.1 - Structure of the « web.xml » file

The « responder » servlet has only one configuration file « web.xml ». Its parameter are :

Parameter	Comments
xmlMarchandsDatabase	**Access path to the « merchants.xml » file (do not modify)** Access path *(relative to the installation directory)* of the servlet (/bundle-responder).

7.2.1.2 – Example of the « web.xml » file

```
<?xml version="1.0" encoding="UTF-8"?>
<!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>
<context-param>
        <param-name>keyProviderClassName</param-name>
        <param-value>com.valista.message.SimpleKeyProvider</param-value>
</context-param>
<servlet>
<servlet-name>Responder</servlet-name>
<servlet-class>com.valista.communication.responder.HttpCommunicationResponder</servlet-class>
<init-param>
        <param-name>xmlMarchandsDataBase</param-name>
        <param-value>/WEB-INF/merchants.xml</param-value>
</init-param>
<init-param>
        <param-name>merchantCurrency</param-name>
        <param-value>EUR</param-value>
</init-param>

<!-- NOTIFICATION_MANAGER_PUSH_OFFER_CANCELLATION_CMD -->
<init-param>
<param-name>NMPOC_NEW</param-name>
<param-value>com.valista.core.api.merchant.node.impl.VoidNotificationManagerCommand</param-value>
</init-param>

<!-- NOTIFICATION_MANAGER_IS_USER_OFFER_AUTHORIZED_FOR_BILLING_CMD -->
<init-param>
<param-name>NMIUOAFB</param-name>
<param-value>com.valista.core.api.merchant.node.impl.VoidNotificationManagerCommand</param-value>
</init-param>

</servlet>
<servlet-mapping>
<servlet-name>Responder</servlet-name>
<url-pattern>/responder</url-pattern>
</servlet-mapping>
</web-app>
```

7.2.2 - Configuration file « merchants.xml »

7.2.2.1 - Structure of the « merchants.xml » file

Parameter	Comments
mctId (merchantId)	**Merchant ID (supplied by w-HA)** Allows the w-HA platform to identify the Merchant.
mctKeyId (keyId)	**Secret key ID (supplied by w-HA)** The key value is shared between Kit v3.5 and the w-HA platform. It is connected to the "mctKey" and ensures information integrity.
mctKey (keyValue)	**Secret key value (supplied by w-HA)** The key value is shared between Kit v3.5 and the w-HA platform. It is connected to the "mctKeyId" and ensures information integrity.
logErrorFileName	**Access path *(absolute)* to the "log_error_5XXX.txt" file.**
logFileName	**Access path *(absolute)* to the "log_notifications_5XXX.txt" file.**
urlRedirect	**URL to redirect termination information (optional)**

7.2.2.2 - Example of file « merchants.xml »

```
<?xml version='1.0'?>
<valista-key-database>
<info>WEB Merchant key Database</info>

        <mct-list>
        <mct>
                <mctId>801</mctId>
                <mctKeyId>801</mctKeyId>
                <mctKey>Key for 801</mctKey>
                <logErrorFileName>/usr/local/tomcat/webapps/bundle-responder/WEB-INF
                /log_notificationError_801.txt</logErrorFileName>
                <logFileName>/usr/local/tomcat/webapps/bundle-responder/WEB-INF/log_notification_801.txt
                </logFileName>
                <urlRedirect>http://wha.marchand.com/notification_resiliation.php</urlRedirect>
        </mct>
        <mct>
                <mctId>802</mctId>
                <mctKeyId>802</mctKeyId>
                <mctKey>Key for 802</mctKey>
                <logErrorFileName>/usr/local/tomcat/webapps/bundle-responder/WEB-INF
                /log_notificationError_802.txt</logErrorFileName>
                <logFileName>/usr/local/tomcat/webapps/bundle-responder/WEB-INF/log_notification_802.txt
                </logFileName>
                < urlRedirect >http://wha.marchand.com/notification_resiliation.php</urlRedirect >
        </mct>
        </mct-list>
```

7.2.3 - Log file : « log_notification_XXX.txt »

7.2.3.1 – Log file structure « log_notification_XXX.txt »

The file « log_notification_XXX.txt » logs information coming in the servlet « responder » :

Parameter	Description
Date	Date and time of the termination notification
uo = <user offer id>	Subscription ID (uoid)
r = <reason>	Termination code : cf. § 2.3.6.3
P = <product list>	Product liste : @<mctid>@<pid>|@<mctid>@<pid>|…
o = <external offer id>	Offer ID (oid)
c = <comment>	Termination reason

7.2.3.2 - Example of the log file « log_notification_XXX.txt » :

Fri Oct 05 19:25:03 2007: uo=6-U6141335211845451;r=200;p=@515@P1|;o=O2;c=05/10/07 17:20:46
Sat Oct 06 15:59:16 2007: uo=6-U4643143178635305;r=101;p=@515@P1|;o=O2;c=User billable timeout
Sat Oct 06 16:59:17 2007: uo=6-U4643143178635305;r=101;p=@515@P1|;o=O2;c=User billable timeout
Sat Oct 06 17:59:19 2007: uo=6-U4643143178635305;r=101;p=@515@P1|;o=O2;c=User billable timeout
Sat Oct 06 18:59:21 2007: uo=6-U4643143178635305;r=101;p=@515@P1|;o=O2;c=User billable timeout
Mon Oct 08 15:15:38 2007: uo=6-U6141344148455402;r=107;p=@515@P1|;o=O1;c=08/10/07 15:09:52