Kaltura Configuration Guide

Version: 1.15

Release: 7 Mar 2012

Table of Contents

1 Introduction

This document provides a technical guide for the integration of Kaltura's video platform with our payment plug-in and a third-party Digital Rights Management (DRM) service. After integration you will be able to start charging for premium video content through your website.

We recommend you also reading the CarrotPay-Overview for Merchants document so you become familiar with the general concepts of CarrotPay payments.

2 Quick Start

1. Register at Kaltura video platform and get your Partner ID
2. Register as a CarrotPay Merchant and get your Merchant ID

   You may obtain a CarrotPay Merchant account in one of two ways;

   a. Register directly for a new Merchant account.
   b. or use an existing WebPurse and configure it with Merchant data.
3. Once your Merchant account has been created CarrotPay will allocate you with three important credentials.

Merchant ID	KPPW-KBCD-GDZD-JWMW (this is the same as your WebPurse ID)
Secret	e. g. csswlwclzgchcwch
Hash seed	e.g. jwwvkgkdksmskqcl

NOTE: For more information about the Hash seed and Secret, read document: CarrotPay-Overview for Merchants, chapter: CarrotPay Security.

4. Include the **Carrot JavaScript API**

```
For HTTP pages:    <script src="http://cdn.carrot.org/js/carrot.js"></script>
For HTTPS pages:   <script src="https://cdn.carrot.org/js/carrot.js"></script>
```

5. Create a Pay-Per-View profile in Kaltura – described in chapter 3 Creating a Pay-Per-View profile
6. Assign a Pay-Per-View profile to a video entry hosted in Kaltura – described in chapter 4 Assigning a Pay-Per-View profile to an entry.
7. Install the CarrotPay plug-in to your Kaltura Player – described in chapter 5 Installing the CarrotPay plug-in a Kaltura player.
8. Charge for video on a pay-per-view basis (see chapter 7 Charging for videos without a Digital Rights Management service) without using a DRM or integrate your own or a third party DRM with the CarrotPay plug-in (see chapter 8 Interfacing a Digital Rights Management service with CarrotPay).
9. Embed videos hosted on Kaltura Video platform within your website – described in chapter 10 Embedding Kaltura player in your website.
10. (Optional) Enable our free affiliate network service **share-n-earn** to increase sales through reward based social sharing – described in chapter 11.2 Enabling share-n-earn for video sales.

3 Creating a Pay-Per-View profile

If you plan to charge viewers to watch your videos, you will need to create a Pay-Per-View profile within the Kaltura console and select the 'Advanced Security' and 'Pay-Per-View' options.

1. Open the Access Control menu from the Kaltura Management Console (KMC).
2. Click **Add Profile**.
3. Make sure the **Secure viewing of this video with server side secret (KS)** box is checked.
4. (Optional) To allow a free preview of your content, check the **Free preview** box and set the preview time (minutes : seconds). You may wish to have different free preview times for selected videos, in this case you must create a PPV profiles for each different preview period.

4 Assigning a Pay-Per-View profile to an entry

1. Open the Manage menu from the Kaltura Management Console.
2. Click on the video entry **Name** to which you want to assign a specific Pay-Per-View profile.
3. Click the **Access Control** tab.
4. Select a profile from the **Access Control Profile** drop-down menu.

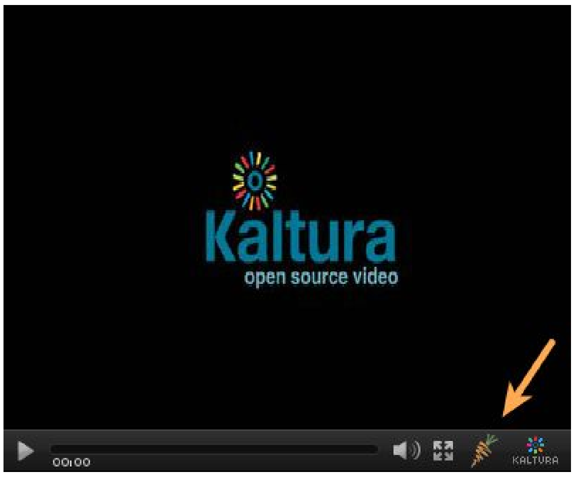
5 Installing the CarrotPay plug-in a Kaltura player

In order to charge for your content you will need to install the CarrotPay plug-in to your player.

1. Open the Studio menu from Kaltura Management Console (KMC) and select the player that you wish to install the plug-in.
2. Select to **Edit** your desired Kaltura player.
3. Click on the **Additional parameters and plugins** button.
4. Use the **Add** button to insert the following key-value pairs.

Key	Value
CarrotPay.plugin	true
CarrotPay.path	http://cdn.carrot.org/swf/CarrotPayPlugin.swf
CarrotPay.relativeTo	ControllerScreen
CarrotPay.position	lastChild
CarrotPay.width	24
CarrotPay.height	24
CarrotPay.entry	{mediaProxy.entry}

5. If you wish to offer more comprehensive viewing rights than simply PPV, you will need to specify the use of a Digital Rights Management (DRM) service to track users and their rights. This plug-in may be configured to use a specific DRM by setting a drm_url key-value pair - please read chapter 9, Option 1 - Specifying the DRM URL in the Kaltura Management Console. Alternatively, skip this step from now and later you can choose to use one of the other 3 available options to specify the DRM URL which are also described in chapter 9 Specifying the Digital Rights Management URL.
6. After saving the changes to the player the Carrot icon will be displayed in the player



6 Communicating with the CarrotPay plug-in

In chapter 5 Installing the CarrotPay plug-in a Kaltura player the basic set-up steps were explained and now we will elaborate on how to communicate through the Carrot library. When the CarrotPay plug-in is installed in the Kaltura player, it provides a declarative interface to the carrot.js JavaScript library by interpreting FlashVars parameters and calling Carrot functions which have been instantiated on host html page.

The table below shows the mapping between the programmatic interface provided by carrot.js and the declarative interface provided by the plug-in.

carrot.js programmatic interface	CarrotPay plug-in declarative interface
Carrot.pay({parameter:'value'})	carrot_pay_parameter=value
Carrot.purse({parameter:'value'})	carrot_purse_parameter=value
Carrot.get_id({parameter:'value'})	carrot_get_id_parameter=value
Carrot.DRM({parameter:'value'})	carrot_drm_parameter=value

Example of calling the Carrot.pay() function with JavaScript

```
Carrot.pay({
        //Merchant ID as allocated during CarrotPay registration
        merchant: "BWRV-JZHS-RQGZ-WLVL",
        price: "0.001:EUR"
        description: "Carrot Icon",
        return_url: "http://www.carrot.org/Carrot[Carrot].png"
});
```

Example of passing parameters to the Carrot.pay() function with FlashVars

```
 <object...
   <param name="flashVars"
```

```
    value="&carrot_pay_merchant=BWRV-JZHS-RQGZ-WLVL&carrot_pay_price=0.001:EUR&carrot_pay_description=Carro
    t Icon&carrot_return_url=http://www.carrot.org/Carrot[Carrot].png" />
</object...
```

NOTE: The plug-in will call Carrot.pay() with the set of supplied parameters and will also include some additional parameters such as entry-id and partner-id.

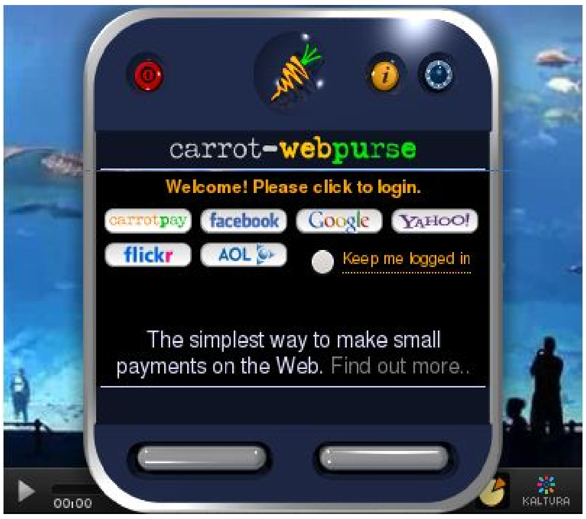
NOTE: In addition to being able to specify parameters for each of the four Carrot functions, you may also specify parameters that will be used with all four functions by prefixing the parameter name with 'carrot_'. However take case when specifying parameters like 'drm_url', as this may easily be mis-read if you are not careful. For example if you want to send the parameter 'drm_url' to the DRM function, you must actually use 'carrot_drm_drm_url' because if you simply use 'carrot_drm_url' it will pass a parameter with the key 'url' rather that 'drm_url' as you may have intended. A useful example of this method is 'carrot_merchant'.

7 Charging for videos without a Digital Rights Management service

You can charge for videos by applying a Kaltura pay-per-view policy:

1. Ensure you have assigned a Pay-Per-View profile entry for the video - see chapter 4 Assigning a Pay-Per-View profile to an entry.
2. Set up parameters, process and validate the payment - see chapter 6 Communicating with the CarrotPay plug-in.
3. Generate and return a Kaltura Session (KS) to the player.

A user has just clicked the **Play** button on the Kaltura player and the Carrot **WebPurse** is displayed and prompts for confirmation of the payment of US 20¢ (as specified by the **carrot_pay_price** parameter in FlashVars).


carrot-webpurse
Welcome! Please click to login.
carrotpay
facebook
Google
YAHOO!
flickr
AOL
Keep me logged in
The simplest way to make small payments on the Web. Find out more..
00:00
KALTURA

In this case the Kaltura player FlashVars will be something like this:

```
<object...
    <param name="flashVars"
    value="&carrot_pay_merchant=BWRV-JZHS-RQGZ-WLVL&carrot_pay_price=0.20:USD&carrot_pay_description=just
    a video&carrot_pay_other_parameters_here...=..." />
</object>
```

After the user has confirmed the payment, your server should process and validate the return_url based on your website's own criteria (e.g. did the price match the video and was there a valid hash).

Once validation is complete, your server must use the Kaltura API to generate the Kaltura Session (KS) and return this back to the player, thereby allowing the video to continue.

8 Interfacing a Digital Rights Management service with CarrotPay

8.1 Introduction

Digital Rights Management (DRM) in this context referrers to the 'Digital Management of Rights', with the intent to allow or restrict access an on-line digital asset (namely a video). It can be used to allow extended viewing periods and to permit access from multiple devices (e.g. computer, pad or phone).

When selling videos with extended viewing rights (e.g. 24hr access or monthly subscription etc.), there is a need for a DRM service to identify the viewer and the content, check for any previously purchased rights and to authorize access or request payment as appropriate. CarrotPay provides a convenient way to add a DRM phase to the 'in-video' sales process . When selling a video on a purely pay-per-view basis, there is no need for a DRM as the viewer must make a payment every time and the viewer's identity is likely to be unimportant.

8.2 Your DRM Service

CarrorPay makes no assumptions about the nature of your particular DRM other than;

1. It can be accessed with a public URL
2. This URL can process a few well defined parameters and
3. It can return an HTML result.

If this is not the case then you must arrange for a regular web service (e.g. PHP or Java etc.) to forward the parameters to the actual DRM and then to return a suitable result in HTML format.

8.3 How it works - Overview

When the user is required to pay for a video, the CarrotPay plug-in calls Carrot.DRM() passing the entry_id of the video and the partner_id of the merchant. The CarrotPay library adds the entry_id, parter_id, element_id and webpurse_id (if known), to the drm_url and creates an iframe on the host HTML page. The iframe 'src' is set to the drm_url thereby getting the browser to call the DRM.

When called, the DRM must determine if the viewer owns an existing right to view the video and if so it must generate a Kaltura Session (KS) (using various Kaltura server-side functions), and return a KS to the player via the Carrot JavaScript library.

Using JSP the returned HTML will look something like this example:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<!-- DRM/approved -->
<head>
<title>DRM approval</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<script type="text/javascript" src="https://cdn.carrot.org/js/carrot.js" ></script>
<script type="text/javascript">
<!-- <![CDATA[
$(function() {
       Carrot.initialise_host(function() {
              Carrot.host.handle_return("KS=${KS}");
       });
});
// ]]> →
</script>
</head>
<body style="background-color: transparent">
</body>
</html>
```

NOTE: The important bit here is the call to Carrot.initialise_host() and Carrot.host.handle_return(). This code will cause the **KS** to be returned to the plug-in as a key-value pair (KS=<actual KS string>), and then on to the player, which will then continue to play the video. In this case there is no need to display anything to the viewer so the <body> is empty.

If the viewer does not currently own the right to view the video, the DRM may;

1. return payment data directly to the plug-in if there is only one option or
2. return an HTML page presenting the available payment options to the viewer (e.g. pay-per-view, 24hr rental or package subscription).

To directly return payment data use something like this:

```
 Carrot.initialise_host(function() {
       Carrot.host.handle_return(
       "merchant=<merchantID>&price=0.2:USD&description=Cool video&return_url=<url>"
       );
  });
```

NOTE : Payment parameters must be passed to the plug-in in standard URL format (i.e. &param=value).

NOTE: The return_url must be URL encoded so that it forms a valid URL parameter. This may easily be achieved by using the “escape()” function in JavaScript or a similar server-side function.

When returning multiple payment options to the viewer it's a little more complex because the choices need to be displayed before the viewer selects one. The follow code shows the essentials element of this process:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<!-- DRM/multiOptionSale -->
<head>
<title>DRM sale</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<script type="text/javascript" src="https://cdn.carrot.org/js/carrot.js" ></script>

<script type="text/javascript">
<!-- <![CDATA[
function cancel() {
if(typeof Carrot.host != 'undefined') {
Carrot.host.popup_message('Sale cancelled',4000);
Carrot.host.handle_failure('cancelled');
 }
}
function revealSalesOffer(area) {
Carrot.initialise_host(function() {
if(typeof Carrot.host != 'undefined') {
Carrot.host.resize_iframe($(area).width(),$(area).height());
Carrot.host.show_iframe("${param['element_id']}");
   }
  });
}
$(function() {
$("li#ppv").click(function() {
var url = "${AUTH_URL}?entry_id=${param['asset_id']}";
        url += "&product_code=ppv&h=[${param['asset_id']} ppv]";
var item = "price=0.2:USD&description=Cool video";
var params = "merchant=BBBB-VVVV-GGGG-QQQQ&"+item;

Carrot.host.handle_return(params+"&return_url="+escape(url));
});
$("li#day").click(function() {


var url = "${AUTH_URL}?entry_id=${param['asset_id']}";
    url += "&product_code=day&h=[${param['asset_id']} day]";
var item = "price=0.5:USD&description=Cool video day pass";
var params = "merchant=BBBB-VVVV-GGGG-QQQQ&"+item;

Carrot.host.handle_return(params+"&return_url="+escape(url));
});
revealSalesOffer($("#container"));
});
// ]]> -->
</script>
</head>
<body class="sale-body" >
<div id="container">
<ul>
<li id="ppv">Pay-per-view</li>
<li id="day">Day pass</li>
<li><a href="Javascript:cancel()">Close</a></li>
</ul>
</div>
</body>
</html>
```

Code example – 8.1

Points of interest in the code example 8.1

1. The Carrot library is initialised with Carrot.initialise_host(). This takes a callback function that will be called once a

connection with the host page has been established by the Carrot library.

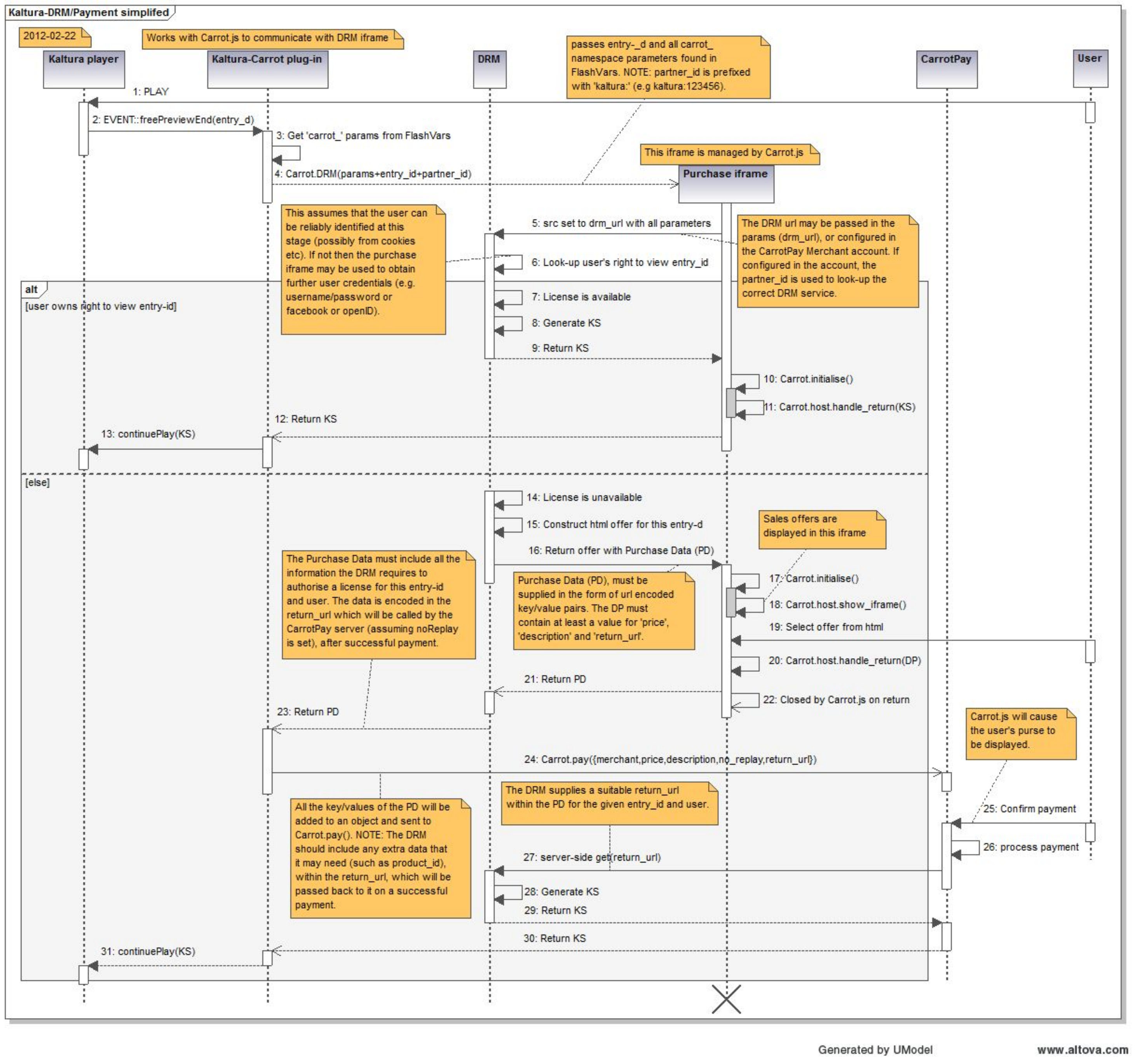
2. An iframe may then be displayed with Carrot.host.resize_iframe() and Carrot.host.show_iframe(). The resize_iframe requires you to set the size of the display area and the show_iframe allows you to position the iframe on the host page. In this example the iframe will be positioned centrally over the player as the plug-in automatically creates an element_id parameter when the DRM is called.

3. Once a selection has been made, payment parameters must be passed back to the plug-in using Carrot.host.handle_return(). The plug-in will recognize that there is no KS parameter defined and will initiate a call to Carrot.pay() with the actual parameters supplied.

4. The return_url has encoded in its parameters a product_code (i.e. ppv & day), which will determine the correct price to be charged. The product_code and asset_id (i.e. the Kaltura entry_id), are also placed between square brackets '[' & ']' and passed as a parameter called “h”. This is the mechanism used by CarrotPay to pass a cryptographic hash (a kind of digital signature), to the return_url so that the return_url can determine if the payment is genuine. Note that when checking the hash, you must use 'price' data exactly as it is presented in the price parameter.

5. You should always provide a cancel option in case the viewer decides not to follow through with a purchase. If selected cancel must use Carrot.host.handle_failure() to cause the Carrot library to close the iframe from the host page. You may optionally display a message to the viewer using Carrot.host.popup_message().

NOTE: For furthered details see the Kaltura DRM/Payment sequence diagram in figure 8.2


Kaltura-DRM/Payment simplifed
2012-02-22
Works with Carrot.js to communicate with DRM iframe
Kaltura player
Kaltura-Carrot plug-in
DRM
CarrotPay
User
1: PLAY
2: EVENT::freePreviewEnd(entry_d)
3: Get 'carrot_' params from FlashVars
4: Carrot.DRM(params+entry_id+partner_id)
This iframe is managed by Carrot.js
Purchase iframe
5: src set to drm_url with all parameters
6: Look-up user's right to view entry_id
alt
[user owns right to view entry-id]
7: License is available
8: Generate KS
9: Return KS
10: Carrot.initialise()
11: Carrot.host.handle_return(KS)
12: Return KS
13: continuePlay(KS)
[else]
14: License is unavailable
15: Construct html offer for this entry-d
16: Return offer with Purchase Data (PD)
Sales offers are displayed in this iframe
17: Carrot.initialise()
18: Carrot.host.show_iframe()
19: Select offer from html
20: Carrot.host.handle_return(DP)
21: Return PD
22: Closed by Carrot.js on return
23: Return PD
24: Carrot.pay({merchant,price,description,no_replay,return_url})
Carrot.js will cause the user's purse to be displayed.
25: Confirm payment
26: process payment
27: server-side get(return_url)
28: Generate KS
29: Return KS
30: Return KS
31: continuePlay(KS)
Generated by UModel
www.altova.com

passes entry-_d and all carrot_ namespace parameters found in FlashVars. NOTE: partner_id is prefixed with 'kaltura:' (e.g kaltura:123456).

This assumes that the user can be reliably identified at this stage (possibly from cookies etc). If not then the purchase iframe may be used to obtain further user credentials (e.g. username/password or facebook or openID).

The DRM url may be passed in the params (drm_url), or configured in the CarrotPay Merchant account. If configured in the account, the partner_id is used to look-up the correct DRM service.

The Purchase Data must include all the information the DRM requires to authorise a license for this entry-id and user. The data is encoded in the return_url which will be called by the CarrotPay server (assuming noReplay is set), after successful payment.

Purchase Data (PD), must be supplied in the form of url encoded key/value pairs. The DP must contain at least a value for 'price', 'description' and 'return_url'.

All the key/values of the PD will be added to an object and sent to Carrot.pay(). NOTE: The DRM should include any extra data that it may need (such as product_id), within the return_url, which will be passed back to it on a successful payment.

The DRM supplies a suitable return_url within the PD for the given entry_id and user.

Figure 8.2 - Kaltura DRM/Payment sequence diagram

4 What should the DRM expect

When called, the DRM should expect to receive a number of parameters which must be processed to determine if a viewer should be granted access to the video immediately or if a payment should first be requested. The CarrotPay plug-in for the Kaltura player will ensure that the following parameters are passed to the DRM:

Parameter	Description
asset_id	This will be set to the Kaltura entry_id for the particular video that has been requested to play.
element_id	This will be set to the id of the player that is playing the video.
webpurse_id	If it's available, this will be set to the unique id of the viewer's WebPurse, otherwise it will be empty.

The DRM should have it's own method of knowing who the viewer is. Perhaps the viewer must first login or maybe the identity is already held in a server session or in a Cookie in the same domain as the drm_url or perhaps the identity is established through OpenID or Facebook etc. If the viewer needs to login but has not already done so, the DRM may return an HTML login page to capture the viewer's credentials before determining if the viewer currently owns viewing rights or not. In this case it must use Carrot.initialise_host() and Carrot.host.show_iframe() to display the login page.

8.5 Releasing a KS after payment

If payment is required and the plug-in is passed appropriate payment data, it will call Carrot.pay() with all the payment parameters and set the no_replay parameter to true. With this option set, the payment process will perform a server-side fetch of the processed return_url.

Once called the return_url should check the hash to ensure that payment has been properly made and if so it may need to update its records to record the fact that the viewer now has future viewing rights for one or more videos in the catalog. It also needs to calculate and return the KS for the specific video identified by the enrty_id which will then be passed back to the plug-in through the CarrotPay system.

NOTE: In this case the KS should NOT be prefixed with “KS=” as is the case when the KS is returned from the DRM process.

9 Specifying the Digital Rights Management URL

Specifying a DRM service to be used with the Kaltura Video platform and CarrotPay payments can be achieved by using one of the following four options.

NOTE: You should installed the CarrotPay plug-in before continuing this section. For more information on how to do that, please read chapter 5 Installing the CarrotPay plug-in a Kaltura player.

NOTE: You are advised not to enable the Kaltura Player feature Share Button (found under KMC > Studio > Edit a Kaltura Player > Features > Share Button) because in most social media websites, e.g. Facebook the player will not be able to display successfully a payment request to the end user and that is because some websites do not allow JavaScript to communicate with the player, (i.e <param name="allowScriptAccess" value="never"/>). Alternatively, you are encouraged to enable the share-n-earn functionality (see chapter 10 Enabling share-n-earn for video sales) which will allow users to copy and share the video link to any website. When the link is shared the video content will still not be viewed within the shared website, e.g. on a Facebook Wall but instead the user will have to click on the displayed link which will redirect him to the merchant’s website where from there the video will be possible to be played and successfully display the payment request to the end user.

9.1 Option 1 - Specifying the DRM URL in the Kaltura Management Console

This solution is suggested if you are planning to enable the Kaltura player "Share Button". The "Share Button" allows users to copy the Embed Code of your video entries from your page and paste them in other pages (e.g. a blog).

NOTE: As mentioned earlier in chapter 9 Specifying the Digital Rights Management URL, when the video is shared using the Share Button the end user when is trying to play the video content may not be able to successfully receive a payment request prompt due to the allowScriptAccess parameter set to never by some websites.

You can specify the DRM URL for your desired Kaltura player by adding the following key-value pair under the Additional parameters and plugins panel in Kaltura Management Console (KMC).

1. From the Studio menu in KMC , select to edit the player you wish to add the DRM URL.
2. Open the **Additional parameters and plugins** panel.
3. Use the **Add** button to insert the following key-value pair.

Key	Value
carrot_drm_drm_url	https://yourvideowebsite.com/yourDRM?

NOTE: The value of the carrot_drm_drm_url key should point to your DRM URL.

9.2 Option 2 - Specifying the DRM URL in your website with JavaScript

This solution is suggested if you are not planning to enable the Kaltura player **Share Button**, i.e. your videos will be viewed only within your own site.

1. Include the Carrot API JavaScript library:

```
For HTTP pages:  <script type="text/javascript" src="http://cdn.carrot.org/js/carrot.js"></script>
For HTTPS pages: <script type="text/javascript" src="https://cdn.carrot.org/js/carrot.js"></script>
```

2. Set the Carrot.drm_url parameter to point to your DRM URL.

   Here is the simplest way to achieve this:

```
<script> Carrot.drm_url="https://secure.yourwebsite.com/yourDRM?"; </script>
```

However, the following jQuery method may be more reliable because it ensures the Carrot library has been loaded and initialised.

```
<script>
$(document).ready(function() {
//The Carrot.drm_url will be set-up when the page is loaded
Carrot.drm_url="https://secure.yourwebsite.com/yourDRM?";
});
</script>
```

NOTE: The jQuery library is automatically loaded buy the carrot.js JavaScript library.

9.3 Option 3 - Specifying the DRM URL in FlashVars

This solution is suggested if you are planning to use a different DRM URL for different sets of video entries while using the same Kaltura player for all entries. An alternative solution to this is to create as many Kaltura players as required and set the DRM URL for each individually through the **Additional parameters and plugins** panel in Kaltura Management Console (KMC).

NOTE: If you have the player's "Share Button" enabled, and a viewer shares the video on a third-party site (e.g. facebook), Kaltura will recreate the embed code from its internal records. In this way any manually added FlashVars parameters will not be included in the shared player.

NOTE: As mentioned earlier in chapter 10 Embedding Kaltura player in your website, when the video is shared using the Share Button the end user when is trying to play the video content may not be able to successfully see a payment request prompt due to the allowScriptAccess parameter set to never by many websites.

In the highlighted line below, an example is given of how the **carrot_drm_drm_url** attribute has been set-up to point to a DRM URL using FlashVars.

```
<object>
 id="kaltura_player_1328581297"
 name="kaltura_player_1328581297"
 type="application/x-shockwave-flash"
 allowFullScreen="true"
 allowNetworking="all"
 allowScriptAccess="always" height="330"
 width="400"
 bgcolor="#000000"
 xmlns:dc="http://purl.org/dc/terms/"
 xmlns:media="http://search.yahoo.com/searchmonkey/media/"
 rel="media:video"
 resource="http://www.kaltura.com/index.php/kwidget/cache_st/1328581297/wid/_757252/uiconf_id/7124832/entry
 _id/0_cj9cah63"
 data="http://www.kaltura.com/index.php/kwidget/cache_st/1328581297/wid/_757252/uiconf_id/7124832/entry_id/
 0_cj9cah63"
 wmode="opaque">

 <param name="wmode" value="opaque"/>
 <param name="allowFullScreen" value="true" />
 <param name="allowNetworking" value="all" />
 <param name="allowScriptAccess" value="always" />
 <param name="bgcolor" value="#000000" />
 <param name="flashVars" value="&carrot_drm_drm_url=http://yourwebsite.com/yourDRM?" />
 <param name="movie"
 value="http://www.kaltura.com/index.php/kwidget/cache_st/1328581297/wid/_757252/uiconf_id/7124832/entry_id
 /0_cj9cah63" />
</object>
```

9.4 Option 4 - Specifying the DRM URL in the CarrotPay Account Control Panel

This solution is suggested when for some reason you are unable to set up the DRM URL by using any of the previously mentioned options or simply because you feel this way might be more suitable to what you are trying to achieve. When the DRM URL is set in the CarrotPay Account Control Panel the URL will always be retrieved through Carrot.

You can specify the DRM URL in the Control Panel by executing the following steps:

1. Open the CarrotPay Account Control Panel
2. Under **Configuration** panel, click **Enter optional data**
3. Set the **DRM url** e.g., https://secure.yourwebsite.com/drm
4. Set the **Partner ID**. Enter 'kaltura:' followed by your Kaltura partner ID with no spaces.

```
kaltura:123456
```

5. Click **Save**

10 Embedding Kaltura player in your website

After you have successfully installed the CarrotPay plug-in you can start embedding premium videos that can be played with the Kaltura player in your site.

1. Open the Manage menu from the KMC.
2. Click the **Preview & Embed** link for the video you want to embed.
3. Under the **Select Player** drop-down menu choose the Kaltura player that you have installed CarrotPay plug-in.
4. Copy the text from the **Embed Code** and paste it in your page.
5. Within your page, edit the embedded code by adding the wmode attributes to be set to opaque for the object as well as the param as seen below in the highlighted lines.

```
<object id="kaltura_player_1328581297"
   name="kaltura_player_1328581297"
   type="application/x-shockwave-flash"
   allowFullScreen="true"
   allowNetworking="all"
   allowScriptAccess="always" height="330"
   width="400"
   bgcolor="#000000"
   xmlns:dc="http://purl.org/dc/terms/"
   xmlns:media="http://search.yahoo.com/searchmonkey/media/"
   rel="media:video"
   resource="http://www.kaltura.com/index.php/kwidget/cache_st/1328581297/wid/_757252/uiconf_id/7124832/en
   try_id/0_cj9cah63"
   data="http://www.kaltura.com/index.php/kwidget/cache_st/1328581297/wid/_757252/uiconf_id/7124832/entry_
   id/0_cj9cah63"
   wmode="opaque">
```

```
    <param name="wmode" value="opaque"/>
    <param name="allowFullScreen" value="true" />
    <param name="allowNetworking" value="all" />
    <param name="allowScriptAccess" value="always" />
    <param name="bgcolor" value="#000000" />
    <param name="flashVars" value="&" />
    <param name="movie"
    value="http://www.kaltura.com/index.php/kwidget/cache_st/1328581297/wid/_757252/uiconf_id/7124832/entry
    _id/0_cj9cah63" />
</object>
```

11 Enabling share-n-earn for video sales

11.1 Introduction

share-n-earn (S-N-E), is a free ad-hoc affiliate network service for web sites (merchants), who wish to promote selected sales items, (e.g. video, music, games) or their site as a whole. A merchant signifies their willingness to share sales revenue with an affiliate by enabling the S-N-E button of CarrotPay plug-in after it has been installed in the Kaltura player. The use of this button by a potential affiliate creates a custom version of a URL supplied by the merchant (known as an 'affiliate URL'), to be posted on the affiliate's social network page, sent in an email or posted on the affiliate's own web site. If subsequent use of this URL results in a sale for the merchant, the merchant agrees to split the sales revenue with the affiliate. We recommend you also read the CarrotPay:share-n-earn document so you become familiar with the general concepts of share-n-earn and find out more technical information and further examples of how share-n-earn can be integrated in your site.

11.2 Enabling share-n-earn via Kaltura Management Console

This option is ideal if you wish to operate a site wide S-N-E scheme where every sne_link is the same. Alternatively, if you would prefer to promote individual videos, you may wish to dynamically set the parameters through FlashVars, thereby having the flexibility to vary the terms of the scheme. Chapter 11.3 Enabling share-n-earn via FlashVars parameters has more details of this.

After you have successfully installed the CarrotPay plug-in, you may enable S-N-E by adding parameters in the Kaltura Management Console (KMC).

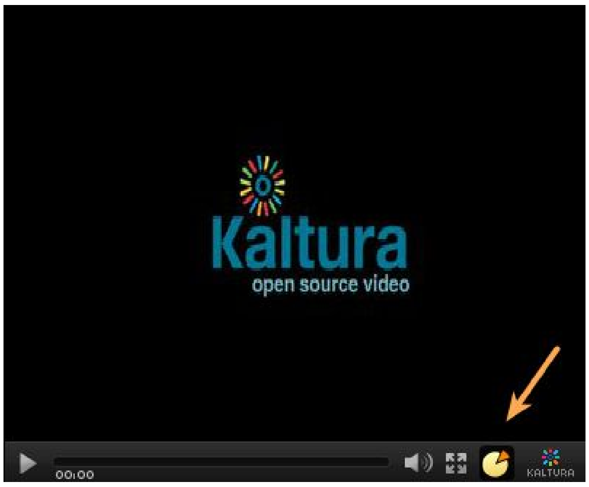
1. Open the **Additional parameters and plugins** panel.
2. From the Studio menu in KMC , select to edit the player you wish to enable S-N-E.
3. Use the **Add** button to insert the following required key-value pairs.

Key	Value (Example)
carrot_purse_se_link	A URL that allows the merchant to determine what is being promoted. e.g. http://www.yoursite.com/affiliate-sale?prod=1234
carrot_purse_se_title	A short description of the content being promoted. e.g. A funny video

4. (Optional) Use the **Add** button to insert the following optional key-value pairs.

Key	Value (Example)
carrot_purse_se_cobrand	A short name to be displayed during the sharing process. Defaults to the domain name of the host site.
carrot_purse_se_earn	A very brief description of the sharing scheme. Defaults to blank, e.g. Earn 10%
carrot_purse_se_scheme	The URL of a page describing the merchant's share-n-earn scheme. This URL may be activated by an affiliate by clicking an information icon provided in the share-n-earn widget. If not provided the affiliate will be sent to a generic description of the service hosted by carror.org. e.g. http://www.yoursite.com/shareNearn.html

5. After saving the changes to the player the S-N-E button will be displayed in the player



6. Include on your host page the following two libraries.

```
<script src="http://cdn.carrot.org/js/carrot.js" type="text/javascript"></script>

<script src="http://cdn.carrot.org/js/shareNearn.js" type="text/javascript"></script>
```

7. Embed a video player in your page using the embedded code – described in chapter 10.
8. Add a server function to process the affiliate URL in your site. The S-N-E service will add an extra parameter called **'aff'** to the carrot_purse_se_link URL before an affiliate can post it. This attribute holds the id of the affiliate and is ultimately used by CarrotPay when splitting the payment. When a buyer clicks an affiliate URL and lands on the merchant site, the **'aff'** parameter must be recorded and used to construct a CarrotPay payment object with the appropriate affiliate id included.

11.3 Enabling share-n-earn via FlashVars parameters

This option is ideal if you prefer your pages to set the **share-n-earn** parameters through FlashVars for each specific video, independently of the Kaltura player(s) being used on your page. Alternatively, if you would prefer to use a specific

Kaltura player for a specific set of videos then read chapter Enabling share-n-earn via Kaltura Management Console

After you have successfully installed the CarrotPay plug-in you can enable S-N-E by adding S-N-E parameters to the FlashVars already in the embedded code.

NOTE: If you haven't yet embedded any video in your page, please read chapter 10 of this document.

1. Include to your page the following two libraries.

```
<script src="http://cdn.carrot.org/js/carrot.js" type="text/javascript"></script>

<script src="http://cdn.carrot.org/js/shareNearn.js" type="text/javascript"></script>
```

2. Add a server function to process the affiliate URL in your site. The S-N-E service will add an extra parameter called **'aff'** to the carrot_purse_se_link URL before an affiliate can post it. This attribute holds the id of the affiliate and is ultimately used by CarrotPay when splitting the payment. When a buyer clicks an affiliate URL and lands on the merchant site, must the **'aff'** attribute must be recorded and used to construct a CarrotPay payment object with the appropriate affiliate id included.
3. Embed a video entry in your page using the Embedded Code – described in chapter 10 Embedding Kaltura player in your website.
4. Add the following required FlashVars in the Embedded Code.

Parameter	Value (Example)
carrot_purse_se_link	A URL that allows the merchant to determine what is being promoted. e.g. http://www.yoursite.com/affiliate-sale?prod=1234
carrot_purse_se_title	A short description of the content being promoted. e.g. A funny video

5. (Optional) Add the following optional FlashVars in the embedded code.

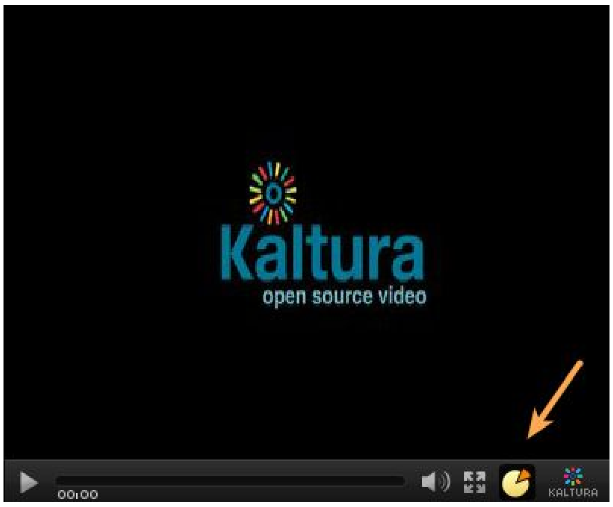
Parameter	Value (Example)
carrot_purse_se_cobrand	A short name to be displayed during the sharing process. Defaults to the domain name of the host site.
carrot_purse_se_earn	A very brief description of the sharing scheme. Defaults to blank e.g. Earn 10%
carrot_purse_se_scheme	The URL of a page describing the merchant's share-n-earn scheme. This URL may be activated by an affiliate by clicking an information icon provided in the share-n-earn widget. If not provided the affiliate will be sent to a generic description of the service hosted by carror.org. e.g. http://www.yoursite.com/shareNearn.html

6. In the highlighted lines below, there is an example given with all the required FlashVars typed within the embed code of Kaltura player.

```
<object
```

```
    id="kaltura_player_1328581297"
    name="kaltura_player_1328581297"
    type="application/x-shockwave-flash"
    allowFullScreen="true"
    allowNetworking="all"
    allowScriptAccess="always" height="330"
    width="400"
    bgcolor="#000000"
    xmlns:dc="http://purl.org/dc/terms/"
    xmlns:media="http://search.yahoo.com/searchmonkey/media/"
    rel="media:video"
    resource="http://www.kaltura.com/index.php/kwidget/cache_st/1328581297/wid/_757252/uiconf_id/7124832/e
    ntry_id/0_cj9cah63"
    data="http://www.kaltura.com/index.php/kwidget/cache_st/1328581297/wid/_757252/uiconf_id/7124832/entry
    _id/0_cj9cah63"
    wmode="opaque">
    <param name="wmode" value="opaque"/>
    <param name="allowFullScreen" value="true" />
    <param name="allowNetworking" value="all" />
    <param name="allowScriptAccess" value="always" />
    <param name="bgcolor" value="#000000" />
    <param
    name="flashVars"value="&carrot_purse_se_link=http://www.yoursite.com/affiliate-sale?prod=1234&carrot_p
    urse_se_title=A funny video&carrot_aff=${param['aff']}"/>
    <param name="movie"
    value="http://www.kaltura.com/index.php/kwidget/cache_st/1328581297/wid/_757252/uiconf_id/7124832/entr
    y_id/0_cj9cah6
</object>
```

7. After having set up all the required FlashVars correctly you should be able to see the S-N-E button in the player as seen below.



11.4 Add share-n-earn functionality directly to a page

If you wish to enable the share-n-earn functionality in your site for the promotion of any content (including video), then you may wish to read the CarrotPay-share-n-earn document which describes how you can achieve that.

For your reference, below there is a simple example of how to instantiate a share-n-earn button for a mobile phone review.

```
<a class="carrot-share-n-earn-button"
data-link="http://www.yoursite.com/affiliate?review=xyz-1625"
data-scheme="http://www.yoursite.com/carrot-affiliate-scheme.html"
data-title="Mobile Phone – Model xyz"
data-earn="Earn 10% for sale of this mobile phone review"
data-cobrand="Your Brand"></a>
```