

# **CarrotPay: Overview for Merchants**

Version 0.1

Copyright © RMP Protection Limited 2008

---

## Contents

Introduction.....	2
What is a Carrot?.....	2
Carrots in spam prevention: CarrotMail.....	2
Carrots as micropayments: CarrotPay.....	2
The Carrot Purse.....	2
The Purchase Process.....	3
1: Register merchant.....	3
2: Configure Website.....	3
3: Visit Website.....	3
4: Pay with Carrots.....	4
5: Collect content.....	4
6: Collect Carrots.....	4
7: Redeem Carrots.....	4
Purchase Security.....	4
What's the problem?.....	4
Other solutions.....	5
CarrotPay security.....	5

---

## Introduction

This document gives an overview of the CarrotPay payment system for merchants wishing to accept Carrots as a currency on their website.

This overview is largely non-technical and does not deal with specific programming issues, for these, see:

- CarrotPay: HTML Integration Guide
- CarrotPay: Script Integration Guide

## **What is a Carrot?**

A Carrot is “digital coin” which represents real monetary value in a currency called Carrots. Carrots are issued by carrot.org (and in the future, other certified licensees of the technology), and can be purchased by credit card or Paypal, and *redeemed* for cash paid into a Paypal account. Carrots are anonymous and do not record anything about the original purchaser.

## **Preventing forgery**

Each Carrot carries an unforgeable unique identity, so they cannot be created other than by a licensed operator. However, being a digital item, Carrots could in theory be copied and an attempt made to use them multiple times. To prevent this, carrot.org provides a verification service which records all the Carrots issued. Any individual Carrot can only be verified once, after which the original Carrot becomes invalid, and a new Carrot of the same value replaces it. When Carrots are redeemed, they are made invalid but the value is transferred into cash instead.

Hence the rule is: Only the first person to present a Carrot for verification or redemption will succeed. This prevents copies being used, but it also requires them to be kept securely, otherwise an attacker could verify them and claim the value themselves.

## **Carrots in spam prevention: CarrotMail**

One early use for Carrots is in preventing spam. The CarrotMail system provides a number of anti-spam features based around a “whitelist” of known contacts, but will allow e-mail to be received from anyone if it has Carrots attached. These Carrots can then be reused on further mails that the receiver sends out. The aim is to allow free interchange of e-mail between people who send about as many e-mails as they receive, but to make it economically non-viable for spammers to send millions of broadcast e-mails.

As a result of use of Carrots in CarrotMail, there will be a large population of users with the capability of spending Carrots for goods and services, which is the main subject of this document. You can find more information about CarrotMail at [www.carrot.org](http://www.carrot.org).

## **Carrots as micropayments: CarrotPay**

The aim of CarrotPay is to provide a low-cost, frictionless method of payment for Web-based goods and services, in particular various forms of paid-for online content such as service subscriptions, premium documents and reports, video, music and images.

The CarrotPay service is designed so that payment can be made with a single click, or – at the user's option (typically for very small values), completely automatically with only a notification popping up that payment has been made. We call this zero-click. The received value can then be collected by the merchant as Carrots on their own computer, and either spent for other services or redeemed for cash at any point.

## **Charge-backs**

CarrotPay will never and can never reverse a transaction or force a charge-back. Once a Carrot has been collected by the merchant it is impossible for CarrotPay or anyone else to take back that Carrot without the agreement of the merchant.

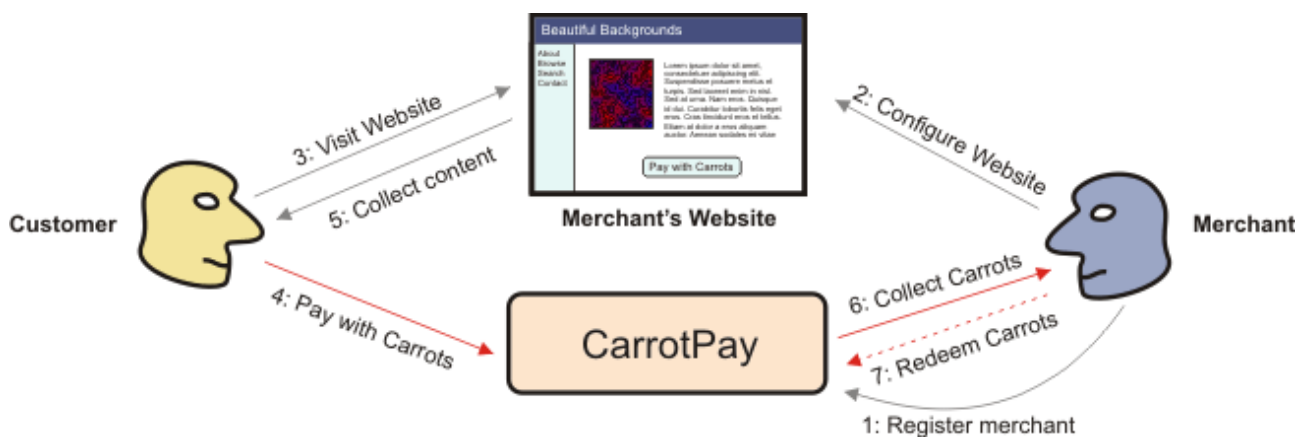
## **The Carrot Purse**

To use Carrots, customers need to install Carrot Purse software on their computer which stores their Carrots and responds to payment requests from Websites. For merchants, the same software also accumulates incoming Carrots for reuse or redemption.

The CarrotMail Client used for spam prevention includes a Carrot Purse, and we are also developing a standalone Purse which will be available on all standard platforms.

## The Purchase Process

We'll use a simple image library website to show the process of purchasing goods or services with CarrotPay, although the system can be used with any type of goods or services. The overall process is as follows:



1. The merchant installs a Carrot Purse, and registers as a merchant with CarrotPay
2. The merchant configures their website to use CarrotPay, and adds the paid-for content
3. A customer visits the Website, browses through the content, and clicks on a "Pay with Carrots" button
4. The Website button leads to CarrotPay, which requests the required number of Carrots from the customer's Carrot Purse, verifies them, stores the value in the merchant's account and redirects the customer back to the merchant's Website through a secure URL
5. The customer collects their content from the Website
6. The merchant's own Carrot Purse automatically retrieves Carrots to the value paid, minus a small processing fee
7. The merchant can then optionally redeem Carrots and have cash paid into their Paypal account

We'll now look at each step in detail:

### **1: Register merchant**

To accept CarrotPay payments, the merchant first has to install a Carrot Purse themselves. Currently, this means installing the full CarrotMail software, although it's not necessary to enable the anti-spam features. The software is freely available for download from the Carrot.org web site and is provided under the terms an end user licence.

Next, the merchant visits the CarrotPay website and asks to register as a merchant. This generates a new account with a unique identifier and other security information (discussed in more detail in the Integration Guides), and configures the merchant's Carrot Purse to receive the payments. The process is anonymous and the merchant does not have to provide any personal information. However, if the merchant wishes to be included in Carrot.org's promotional activities they may provide their web site URL and a short description of their products and services.

### **2: Configure Website**

The merchant then configures their own website to accept CarrotPay payments. At its simplest this can be putting an HTML form on a page and renaming some files – the system is specifically designed to work without needing scripting or back-end Web development, and hence can be used on very basic (or free) Web hosts.

It is also possible to integrate CarrotPay into new or existing Web shopping carts and other e-commerce systems. Our Script Integration Guide gives more details and examples of this. In either case the Carrot Purse software provides a few simple tools to aid in the site development process.

### **3: Visit Website**

What the customer sees on the Website is entirely under the control of the merchant. To pay with Carrots, the customer simply clicks on a "Pay with Carrots" button which has been configured with the merchant's ID, the price required and a return URL for the content to be delivered.

## **4: Pay with Carrots**

When the customer clicks on the “Pay with Carrots” button, the CarrotPay service requests the required number of Carrots from the customer's Carrot Purse. If the amount is below a user configurable limit, the Carrot Purse may engage zero-click mode and make the payment automatically, simply notifying the customer that payment has been made. By default, though, the customer will be asked to authorise or reject the payment with a single-click.

When the payment has been authorised, the Carrot Purse returns the required number of Carrots. The CarrotPay service then verifies they are valid and if so credits the value to the merchant's account.

The CarrotPay service then redirects the customer back to the merchant's Website through the configured return URL, modified for security as discussed below.

NOTE: CarrotPay will NOT send the buyer a purchase confirmation email as no personal details are ever collected from the buyer. It is the merchant's responsibility to collect buyer details and to send a confirmation email if they feel this is appropriate. However, for many small-values transactions with no physical delivery, such data collection will not be justified.

## **5: Collect content**

The customer's Web browser will then fetch the content (or another page which leads to the content) as usual, using the secured URL. This is the end of this purchase for the customer.

## **6: Collect Carrots**

All the time the merchant's computer is on, their Carrot Purse will be checking with the CarrotPay service for new payments. Once sufficient payments have been received, the Carrot Purse will automatically download the Carrots for storage locally.

There is a small processing charge applied by CarrotPay at this stage, currently 2% of the value collected, subject to a minimum of 0.02 Carrots. The merchant can adjust the minimum value to check for, trading off the benefits of immediate receipt with the minimum fee charged for each collection.

## **7: Redeem Carrots**

Once the merchant's Carrot Purse has received the Carrots, the merchant can simply use them to buy other online services in the manner described above. Alternatively, they could use the Carrots to attach to email messages to their own customers or redeem the Carrots for cash through the carrot.org Website. Redemption for cash of any value may be made to a Paypal account and larger sums may be transfer directly to a merchant's bank account.

---

## **Purchase Security**

One of the major issues integrating simple HTML pages with Web-based payment systems is providing proof against attempts to obtain the content or goods without paying, by 'spoofing' the URL which leads to the content or authorises delivery, without going through the genuine payment process.

Payment services such as Paypal have a number of ways of dealing with this, including encrypted buttons, IPN, PDT and full back-end integration. CarrotPay has a single, much simpler mechanism which provides security all the way from simple HTML pages to full e-commerce backends. The details of the system are given in the Integration Guides, but we can give an overview here.

### **What's the problem?**

The problem with 'spoofing' arises because in simple integrations there is no direct communication between the merchant's Website and the payment service – everything happens through pages being fetched by the customer's browser.

Hence a typical payment button contains information which makes the customer's browser go to the payment service website, saying something like “*Please take payment of \$X, for product Y, credited to merchant account Z, and return the user the page at URL U*”. The payment service then goes through its normal login procedure to identify the customer, takes the payment, and tells the customer's browser to return to URL U. The Web page then takes the request for U to allow delivery of the content directly, or authorise the physical delivery of goods.

The problem which arises is that with a little knowledge of the payment system and HTML, a fraudster can go through the purchase up to the point where they have to pay, but then just skip that and point their browser at URL U, making the merchant's website think that payment has been made when it has not.

### **Other solutions**

Some of the solutions to this problem provided by other payment services include:

- Encrypted buttons: You need to have the HTML for the button specially created by the payment service

Website, in such a way that only they can decrypt it and work out the return URL. This only works for fixed prices and products, but it does work without any scripting.

- IPN: The payment service will fetch a special page from your Website telling you that a payment has been authorised. You need to have some script and database on the site to tie this together with the purchase process.
- PDT: The payment service provides a way of checking whether a payment has been made properly or not before you deliver the goods. Again, this requires scripting.
- EWP: Like with Encrypted Buttons, but the buttons are created dynamically. Very powerful, but it requires a lot of cryptographic knowledge and scripting to work.
- Secure hashes: The payment service adds an additional 'hash' parameter to the return URL created using a secret shared between the payment service and the merchant's website (but not revealing it), which proves the request has been through the proper payment system and hasn't been modified. This requires scripting on the merchant's Website to verify the hash.

The CarrotPay system is most like the last one, secure hashes, but simplified so it can be used without any scripting, as with encrypted buttons.

## **CarrotPay security**

The CarrotPay solution to the 'spoofing' problem is very simple, yet secure and powerful. When a merchant registers with CarrotPay, as well as their merchant ID they are also given a secret 'seed' value which is known only to CarrotPay and the merchant. This is automatically stored in the Carrot Purse.

When a customer clicks on a “Pay with Carrots” button, the CarrotPay service modifies the return URL that is encoded in the button by replacing any parts in square brackets “[xxx]” with a group of letters, or 'hash'. This hash represents a combination of the original contents of the square brackets, the price quoted and the secret seed value, in such a way that changing any of them changes the hash, but which does not reveal what the secret is<sup>1</sup>.

### **Simple static URLs**

In the simplest case, if the URL configured into the payment button was originally:

```
http://www.example-image-library.com/images/[butterfly].jpg
```

The process might turn it into

```
http://www.example-image-library.com/images/bzcmpwxj.jpg
```

Hence to hide the content from people who haven't paid, you simply rename “butterfly.jpg” as “bzcmpwxj.jpg”. Someone trying to fetch the content without paying cannot guess this, because they don't know the secret seed. Neither can they modify the payment button to pay less, because changing the price changes the text.

How does the merchant know what group of letters to rename each file to? The Carrot Purse (which knows the secret seed) has a feature where you can enter a URL or a single word and have it translated for you. Hence at this level all you need to do is add some square brackets to the return URL in the payment button, and rename a file.

The only disadvantage of this mechanism (which it shares with encrypted buttons and is really unavoidable unless some scripting is used) is that the return URLs will be the same for everyone. Hence if they become public knowledge (for example, if someone posts them to an online forum) the security is lost. If you become aware of this, all you have to do is generate a different filename and change the payment button, but in general we only recommend this mechanism for relatively low-value content which might easily be copied anyway.

### **Dynamically created URLs**

To avoid the problem of URLs being copied, and avoid the small amount of manual processing required with static URLs, you need to use an element of scripting (e.g. PHP, Java) and database or file to record each transaction individually, as is common with most shopping carts and e-commerce solutions.

In this case, the script and database back-end will most likely generate a unique transaction ID for each purchase. The payment button is then dynamically generated to include the transaction ID in the return URL – e.g.:

```
http://www.example-ecommerce.com/scripts/payment-return?tx=12345
```

This payment return URL then checks the transaction hasn't already been completed, and completes it – for example, by delivering the content, or authorising physical delivery of the goods.

Of course, as we saw before this return URL can be spoofed, so we need to prove that it has been passed through the payment system with the right price. The same system of URL modification can provide this as well. All we need to do is include the same transaction ID in another parameter, but this time in square brackets:

---

<sup>1</sup> Technically, the hash is a safe encoding of the lowest 32 bits of an MD5 digest. The exact algorithm is given in the Script Integration Guide.

`http://www.example-ecommerce.com/scripts/payment-return?tx=12345&hash=[12345]`

The CarrotPay service will then turn this into something like:

`http://www.example-ecommerce.com/scripts/payment-return?tx=12345&hash=[gbjwzcbz]`

The payment return script then needs to recalculate the hash word using the original transaction ID, the price quoted and the secret seed (which it will need to have configured into it in some secure way). It can then compare this with the hash quoted in the return URL to verify that payment has completed properly and complete the purchase.

The exact details of the hash algorithm and example code in various common scripting languages are given in the Script Integration Guide.