Master's Project: Micropayments Platform

Jose M. del Barco

September 30, 2009

EPFL-LBD Database Laboratory

Professor:
Stefano Spaccapietra

Assistant:
David Portabella

Abstract

Current web payment systems are not suitable for transferring very small amounts of money, such as 0.001 Euros. Because of that it is currently not possible to ask users to pay little prices for small services, such as searching once in an online catalog. In addition, current payment procedures require a large number of steps to be performed. This project consists in (1) designing and developing a platform targeted to micropayments, with the corresponding website and web API, and (2) create a user-friendly, fast and reliable payment procedure, easy to implement for merchants.

Contents

1 Introduction

It is undeniable that Internet has become one of the best human inventions. At the beginning of its life it was used mostly as a way of communication and information retrieval, but it has quickly evolved to become an all-mighty tool, something humans need to satisfy their social needs, and from not that long ago, even their consuming needs. As the rate of Internet utilization among the global population increases, also does the amount of business and markets around it. With the power of Internet to spread information, companies are starting to look at it as the best way to get new consumers, and so they create websites to get their attention and be able to multiply their benefits.

Using Internet as a selling tool is not that easy, though. People still do not trust Internet as a way of payment, due to the amount of hacking and the apparent fragility of their security methods. In addition, it is very difficult to make people trust a payment not performed with a human seller, where they can actually see the exchange of goods by money. On Internet there are no human sellers. All the process is performed by a computer program, bound to failures and bugs, making the consumer feel insecure and unprotected against those potentially wrong purchases.

Even overcoming those fears and using Internet as a tool to purchase goods, there are cases when it is very difficult to perform a sale. To perform a transaction over Internet merchants use "payment gateways", which are softwares (from separate companies or banks) that understand the language that real bank transaction servers speak so they can transform any payment data (a credit card number and the user data, for example) into real bank transactions. However, as a part of the payment business, both gateways and banks charge a fee for being used, which translates into a fixed cost per on-line transaction. This leads to a huge problem, because... What happens if the cost of the product is lower than its fixed cost? This is known as the *micropayment problematic*. Solving it can create new conceptions of Internet payments, and this is the goal of this project.

1.1 Definitions

The following are the definitions of the terminology used in this report regarding the different levels of interaction between human beings and the Internet application described in it.

MicroPayments Platform (MPP):
The subject of this thesis project. It can be referred by its capital letters or as the Service Provider in the OAuth scheme.

Merchant:
An e-commerce business that uses the MPP to charge its Users for using his services.

User:
A physical person who access Internet to use the Merchant e-commerce services, resulting in performing transactions in the MPP system. Note that for the MPP, merchants are also users.

Payment Service:
Services on which the Merchant bases his sales, used to charge Users to get a benefit.

2 Micropayments

Micropayments are generally defined as payments lower than 10$. Products or services with lower prices are much harder to exploit as the benefit per product is too low, even producing a cost instead of benefit, making the transaction not affordable. That is why this kind of payments are still not very implemented in actual e-commerce based services, and it is still a subject of study and research.

By reading different research papers [5][3][4], I observed that the actual efforts to solve the micropayment problematic were either by aggregating a specific amount of transactions into a single one to condense the transaction fees, or creating a virtual currency to work with. After investigating already existing Internet payment services such as PayPal, I concluded that a good approximation to solve the micropayment problematic could be combining those micropayments to a virtual account model, thus creating a virtual accounting system which would have to be used by both users and merchants. This is the way the MicroPayments Platform, the subject of this project, was born.

The MPP will act as an intermediary between users and merchants. It will create virtual accounts with which users will perform transactions in the accepted merchants. To do so, it will make users charge their MPP accounts with real money using a third service (for the moment PayPal), and it will make merchants accept transactions from the MPP accounts. Instead of performing real bank transactions between them, MPP will simply update their virtual accounts balance.

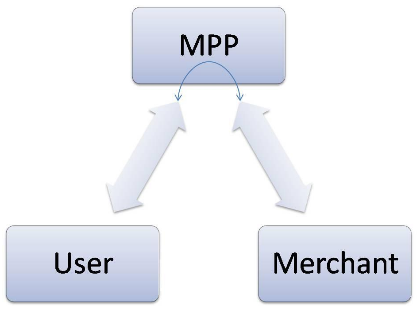


Figure 1: MPP interactions

The first part of this report makes a small overview of the most successfull payment platforms on Internet and explains the desired requirements to transform the MPP into a true competitor: its features, its effort for provide secured transactions and the web interface associated to it. The second part provides a detailed description of the technical implementation of those techniques and presents two examples of implementation. Finally the third part concludes the report with the results obtained and some thoughts about how to enhance the platform.

Part I
The MicroPayments Platform (MPP)

3 Current Payment Platforms

There are plenty of payment platforms currently working on Internet, some of them already famous and commonly used. To design another payment platform that could compete with the existent ones, or at least get a share of the market, we need to know all the features of the competence, enhance them and/or create new ones. In this section I present this brief study, which will establish the basic features to implement in the MPP platform.

3.1 PayPal

Without a doubt, Paypal (http://www.paypal.com) is the most used of the current Internet payment platforms. Millions of people are purchasing goods every day with their services thanks to its integration with eBay, the most known auction service on Internet, and its wide use and easiness to use (the transactions are triggered by embedded buttons) makes it a reliable way to perform transactions for most users.

By creating a PayPal account the system offers a virtual account system from which the user is able to perform a variety of transactions. There are three different types of accounts, depending on the level of complexity of the transactions to perform:

***Personal*:** for users seeking simple checkout processes and basic transfers of money;

***Premier*:** for more advance users who seek to buy and sell goods over Internet;

***Business*:** for companies that want to integrate PayPal as the payment processor for their commerces;

To integrate their services, PayPal offers different API packages depending on the level or complexity of the transactions to perform. The basic package, called *Website Payments Standard* is a free API to create personalized buttons and perform basic transfers between PayPal accounts. The *Website Payments Pro* is a pay access that offers merchants capable of making credit card payments (PCI compliants) a variety of services to integrate PayPal accounts into their payment system. Finally, PayPal also offers a pure gateway service called *PayFlow*.

All those features together with its wide use makes PayPal the reference of how users want to purchase their goods, and thus one of the main references for this project.

3.2 Google Checkount

Inside the politic of Google to conquer all the possible markets on Internet, a payment system couldn't be let appart. Google Checkout (http://checkout.google.com) tries to be a direct competitor of PayPal in the Internet payments market, but it's limited to a checkout system. However, the integration with their OpenID system adds anonymity to the transactions, a really important feature nowadays. Unfortunatelly, Google Checkout is currently limited to the United States, and does not provide bank interfaces or any kind of virtual accounts, but just credit card payments.

3.3 MoneyBookers

MoneyBookers (http://www.moneybookers.com) is a clear example of complementing the PayPal services while trying to get a market share in the checkout and auction services. Also under a virtual account system it offers mobile messaging and fax services, but most importantly it implements a mobile payment system. It also offers auction and checkout services similar to PayPal.

3.4 VirtueMart

VirtueMart (http://virtuemart.net/) is a different concept of payment processor specially designed for the Joomla and Mambo Content Management Systems (CMS). Instead of limiting their services to offer Internet payments, they provide a full payment system to be easily integrated into the CMS as a plugin, with a complete backend to configure all aspects of the payment process including a product catalog manager. VirtueMart uses other third party services for performing the transactions, such as PayPal, but it can transform any website made by the supported CMS to a full chekout-based e-shop in a short amount of time.

3.5 Others

There are a high amount of other payment processors and gateways working nowadays. Most of them offer similar services as the ones presented here, others try to offer different, more advanced and specific services. Other payment processors to have into consideration could be CCAvenue (http://www.ccavenue.com) and regSoft (http://www.regsoft.com), overall payment processors with multiple services, or 2Checkout (http://www.2checkout.com), based in checkout processes.

4 Features of the MPP

Now that we know the general features expected for a payment processor, we can start talking about what we want to achieve with the MicroPayment Platform. The MPP is intended to be an advanced Internet payment system, with the main goal of being user-friendly and highly automated for users while being really easy to implement for merchants. The project was designed to integrate most of the already existing features in payment processors while adding some new, advance ones. Those features are:

- Uses PayPal accounts as the way to deposit and/or withdraw money in the MPP virtual accounts;
- Allows different kinds of payment services, without any amount restriction:
  - *e-shop*: payment links and buttons, shop carts, etc;
  - *Pay-Per-Service*: pay-per-time, pay-per-bandwidth, pay-per-token;
  - *Custom Payments*: fully configurable (e.g. combination of various services);
- Handles different currencies;
- Lets the user set up automation behaviors to get agile payments;
- Performs anonymous payments (merchants do not have access to user's payment data);
- Provides user and merchant configuration back-ends in the MPP website;
- Provides a new in-site, real-time, merchant-independent control panel for users.

4.1 PayPal Funding

To become a payment platform the MPP needs a way to interact with real money from both users and merchants. As I have explained before, banks use specific protocols to perform transactions over Internet, so the first thing to do is being able to use those protocols. The easiest way to do that is by using Payment Gateways, which are computer programs that are able to use those protocols to send user and credit card data, i.e. they are able to order money transfers from one bank account to another by using just the credit card information (among other functionalities).

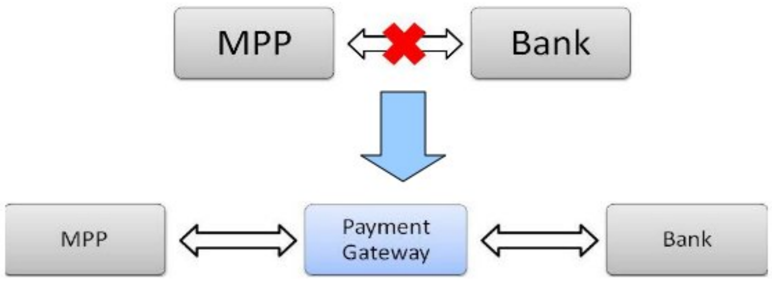
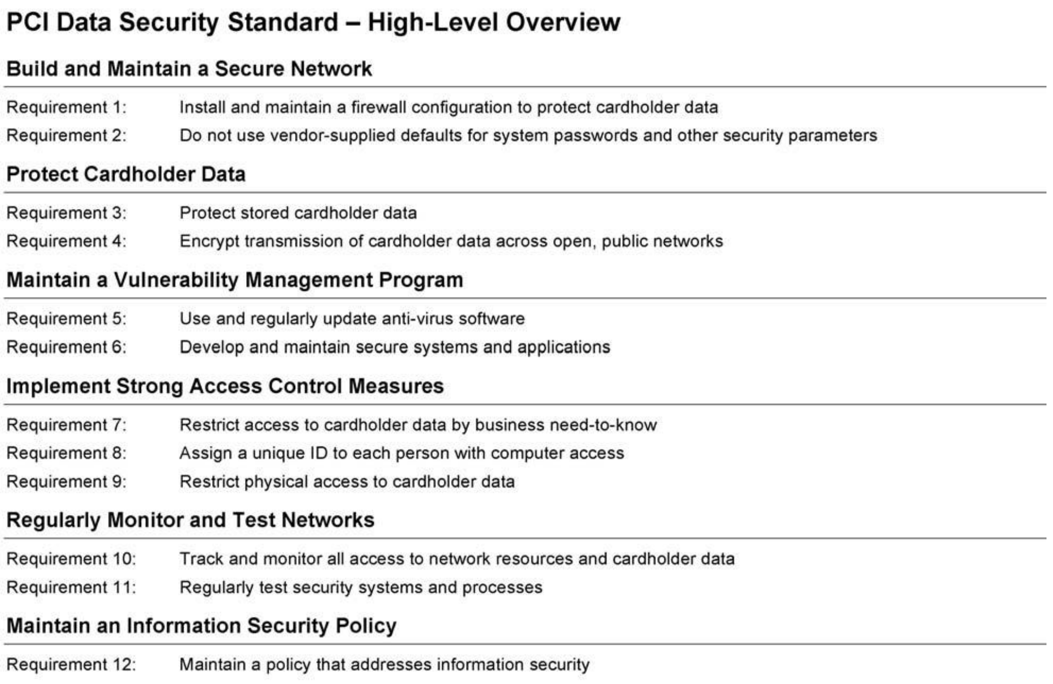


Figure 2: Payment Gateways

However, allowing users to pay directly by introducing their credit card information in an web requires a high level of security. Because of the vulnerability of Internet to hacking attacks, which in this case would translate into potential money robbery, the Payment Card Industry Security Standards Council, or PCI-SC, formed by the main credit card companies (Visa, MasterCard, American Express, JCB and Discover Network) created the PCI - Data Security Standard or PCI-DSS, which is a set of 12 mandatory rules regarding all the aspects of the electronic card



PCI Data Security Standard – High-Level Overview

Build and Maintain a Secure Network	
Requirement 1:	Install and maintain a firewall configuration to protect cardholder data
Requirement 2:	Do not use vendor-supplied defaults for system passwords and other security parameters
Protect Cardholder Data	
Requirement 3:	Protect stored cardholder data
Requirement 4:	Encrypt transmission of cardholder data across open, public networks
Maintain a Vulnerability Management Program	
Requirement 5:	Use and regularly update anti-virus software
Requirement 6:	Develop and maintain secure systems and applications
Implement Strong Access Control Measures	
Requirement 7:	Restrict access to cardholder data by business need-to-know
Requirement 8:	Assign a unique ID to each person with computer access
Requirement 9:	Restrict physical access to cardholder data
Regularly Monitor and Test Networks	
Requirement 10:	Track and monitor all access to network resources and cardholder data
Requirement 11:	Regularly test security systems and processes
Maintain an Information Security Policy	
Requirement 12:	Maintain a policy that addresses information security

Figure 3: PCI-DSS Requirements

transaction handling (see figure 3). Every business that wants to handle any credit card from the PCI-SC companies on Internet must follow these rules.

To get the PCI-DSS compliance certificate the business has to audit all those rules with a supported auditing company. Unfortunatelly, this makes the direct credit card handling out of the scope of this project, although it should definitely be considered when using the application in a real production environment. The consequence of this is that to be able to handle real money from users and merchants the MPP has to use the services of another company. It must be able to perform transactions by itself, with reduced cost, with a public API to automate them and that is trusted enough by Internet users to be used as the main resource for money transfers. For its widely utilization and trust among Internet users, we decided to use the PayPal services.

PayPal is a PCI-DSS compliant company that allows business to use its public API to perform any kind of transactions over Internet. Although advanced functionalities of that API are only available by paying monthly fees, the basic functionalities are free and enough for the scope of this project.

The way to proceed is simple. The MPP acts as a business (merchant) in PayPal, owning different accounts with different currencies and having access to a limited part of their API. A user that wants to use the MPP is asked to use PayPal to charge their MPP Virtual Account in the currency he wants by clicking on the respective button in their user area in the MPP website. This redirects them to the PayPal site, which asks them the amount to charge and the payment method (credit card or using their own PayPal account). Once the user has charged his MPP Virtual Account, he will be able to use all the features of the MPP system explained in this section. If the user wants to retrieve back his money from the MPP Virtual Account he will need to have a PayPal account, where the MPP will transfer the amount he desires.

This way, the MPP handles the money of both users and merchants by saving it in its own PayPal account and then performing virtual transactions between them (figure 4). No real money is transferred between users and merchants, so the transactions do not have a fixed cost and thus the amount has no limits, solving the micropayment problematic. So, by paying a small fee to PayPal for performing the real transactions, the MPP becomes a payment platform.

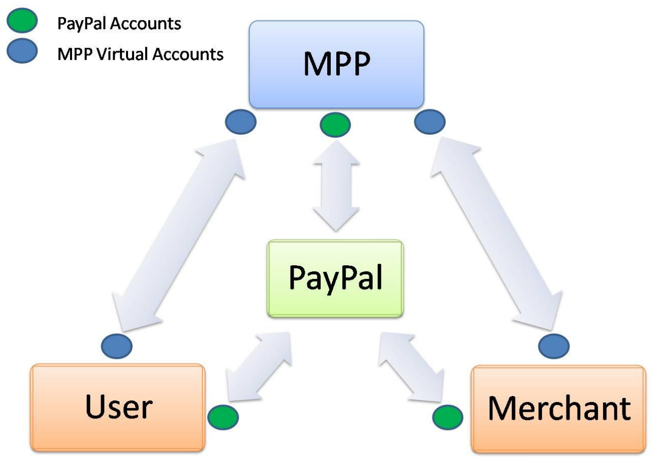


Figure 4: MPP Account Model

4.2 Payment Services

The Virtual Account system used in the MPP, together with the micropayments capabilities and the zero cost of the virtual transactions allows almost unlimited ways of charging users for using merchant services. Not to say that this is a really powerful marketing weapon to attract merchants to use the MPP system, as its versatility makes it able to scope almost every Internet business in the actual e-commerce market.

For the prototype of the MPP system presented in this project two different payment models have been implemented: a basic e-shop (without checkout) and a web-based trivia game. However other models such as charging for time, bandwidth or any specific service could be easily implemented by doing small modifications and/or additions to the MPP API, or even in the merchant side by using the actual API in a convenient way (for instance, instead of having an API call for doing periodical payments, a merchant could program the periodicity in his server and call the payment function every period).

The Payment Service is configured during the signup process of the merchant or in its personal area in the MPP website. It allows to select the basic behavior of the system from the different payment models available, and then it lets the merchant specify a list of currency-dependant prices for the products or services the Payment Service will cover.

The following is a small description of the example models. The technical implementation is explained in detail in the section 13.2.

4.2.1 E-shop Model: www.buyflowers.com

It is a basic e-commerce shop that sells flowers. It has an HTML structure with PHP only for retrieving data from a small database with the product information. It allows different payment options depending on the currency, which can be modified in the database as well. In this case prices are around the micropayment definition, but can be whatever. The scope is to show the behavior of the MPP system under a low number of more expensive transactions.

4.2.2 Web Game Model: www.trivia.com

A PHP web game consisting in answering a series of questions about general or specific topics. It is based in the “PHP Dynamic Trivia” code from Robin van de Vusse (r.vandevusse@rogers.com) and published under the GNU General Public license. The MPP system has been integrated to charge the user in real-time for every formulated question and reward him in case the answer is correct. As the rate of the transactions is higher, a Oneclick transaction automation (see 4.5) is suggested. Other examples with the same implementation than this model could be a gambling website or a flash web videogame.

4.3 Currency Handling

It is obvious that one of the main characteristics of Internet is to be able to access any service from any country in the world. Currency handling is, for that reason, a must for not limiting the amount of users to access a given service.

The MPP implements an advanced currency handling that allows users to purchase in the currency they prefer and also provides the merchants with an easy way of setting up the prices of their products or services depending on the currency the user wants to pay with.

As the real money accounts are based in PayPal, the MPP system has one PayPal account for every possible currency it can handle. Users and merchants are not forced to have a specific currency in their PayPal account, but it is suggested to have at least the same currencies in both PayPal and MPP, as the PayPal system forces the exchange of the currency if it does not match the transaction's.

Despite using a currency-based platform such as PayPal, the currency handling is not problem-free: PayPal has a limited behavior when managing currencies from the public API. In addition, the MPP has to take into account that the transactions between users and merchants are virtual. Users and merchants only get real money when they decide to transfer it from their virtual accounts to their PayPal accounts, and this transfer is performed in the currency of the account they are using for it. The PayPal API does not let forcing a currency in a transaction nor knowing beforehand the exchange rate it will apply in case of currency exchange, so an alternative way to handle currencies had to be implemented.

Having that limited currency behavior, it was decided that for the prototype model of the MPP the currency handling would be internal, meaning that the MPP system would use a FOREX table to get the exchange rates of the currencies it can handle and perform the exchange between the virtual accounts. However, there are two factors that make this a potentially dangerous behavior which could even translate into an undesired cost:

1. The exchange rate PayPal is applying does not have to be the same than the MPP's (and it is not, as they apply a fee in it);

2. The FOREX table fluctuates every day. The exchange rates applied when doing virtual transfers can vary from one day to another, and transferring money from MPP Virtual Accounts to PayPal accounts can use a much lower exchange rate than the one applied when the user did the transactions, so the MPP system can lose (or even win) money just by allowing those exchanges. The only solution could be knowing a-priori the exchange rates that PayPal applies at a given moment, but this data is not directly available using the API. However, there is one way to manually create the forex table PayPal is using: as the MPP system has different accounts for every currency, it could perform a transaction between all its accounts (all the possible combinations of them) and check the exchange rate and fee applied in every transaction, thus constructing the PayPal forex table. However, every transaction has a fixed cost associated to it, so this method should be used weekly or daily as much. In addition, depending on the number of currencies the system handles this method can be very slow, limiting even more the update process and even avoiding it to be real-time.

4.4 Price Optimizer

The internal currency management allows a more precise handling of transactions. For instance, as the exchange is DB based and the transactions are virtual a currency optimizer could be easily implemented. The idea is using the fact that merchants can specify different prices for their products or services depending on the currency used to pay. The payment control system (called User Panel) could let the user click on an option to calculate all the possible currency exchange combinations between his accounts and the payment options the merchant gives and automatically select the one that provides the minimum price to pay. This allows the user to pay the amount the merchant specifies while ensuring that it is the lowest price, avoiding for example abusive exchange rates or, why not, making profit of a favorable exchange rate in a specific period of time.

4.5 Oneclick Payments

Most of the actual Internet payment platforms, such as PayPal or Google Checkout, have slow step-by-step procedures to perform Internet transactions. Most of the times those procedures are favorable for the user, as they give him confidence about the whole process, but if the user trusts the Service Provider and the Merchant it seems logical to speed up the whole process by automating most of the payment steps.

The MPP system, thanks to the virtualization of the transactions, allows the user to configure certain parameters to do that. This feature is called Oneclick payment, as the user only has to trigger the purchase (or transaction) from the merchant website to automatically perform it. The name “oneclick” is well known in the Internet payment industry: Amazon.com had the full patent (sub-licensed to Apple), i.e. the idea and its implementations, until in 2008 was narrowed to Amazons specific implementation (in a shopping cart) because the patent included so many broad definitions that although that functionality seemed a logical evolution of any e-commerce system, no one could implement it.

The MPP Oneclick payment concept differs from the others (such as the Amazon’s) in that the user does not need to register in the merchant’s website to make a payment. Moreover, the merchant does not get the identity of the user, making oneclick anonymous payments possible (see section 4.6).

The generic MPP Oneclick setting has two different sets of parameters: the first set establishes the type of oneclick, as it sets the trigger for the automation, while the second set establishes the duration of the oneclick by different parameters.

The user can set the validity of the oneclick according to three different parameters:

1. **By time:** The Oneclick will be valid during that session, a fixed amount of time (from minutes to one month), always or during a custom amount of time set by the user;
2. **By number of transactions:** The Oneclick will be valid for a fixed or custom number of transactions;
3. **By maximum amount to spend:** The system monitorizes the amount spent during a specific time window and compares it with the selected threshold, which can also be user-defined.

As fully automating a payment process is a delicate matter for the user, the MPP system gives a way to cancel the procedure. When the user triggers a transaction matching the Oneclick configuration, the MPP control panel (see 4.7) shows a countdown of five seconds. This countdown is in fact a button where the user can click to cancel that specific automatic transaction, while keeping active the Oneclick setting.

The following is a description of the types of Oneclick depending on the trigger (subject to the duration explained before).

4.5.1 Express Oneclick

This method consists in saving in the MPP database the account and currency used in the last transaction performed and automatically set those parameters as the default ones for the following transactions while the Oneclick setting is active.

4.5.2 Normal Oneclick

This method consists in letting the user chose between different “payment behaviors”. The MPP system asks the user for what to do if:

1. The merchant specifies different payment options based on the currency:
    (a) Use the same currency than the default user account;
    (b) Use the price optimization engine to get the lowest price/account combination.
2. If the previous behavior fail (due to lack of funds or currency mismatch):
    (a) Use the best next valid account
    (b) Use the account with more funds
    (c) Use a specific account (lets the user choose it)
    (d) Cancel the transaction

4.6 Anonymous Payments

The main concern for users in an Internet payment system is obviously security. When performing transactions over Internet, the identity of the user and his data has to be completely obfuscated, only used when strictly necessary and subject to as many security layers as possible. Users do not necessarily have to trust all the on-line services they find on Internet, but they may want to be able to use them without giving any information to the merchant behind it, not to speak about giving credit card information even to merchants they may trust. In addition, the user has to be sure that only him can access to his data, or grant access to it to a third party, and that the data is secured so they are the only ones that can view it.

For e-shop models anonymity it is not that important, as most of them require shipping information that forces the user to give personal information (and even they may require a signup process to save it), but for other payment models such as on-line games, gambling, etc. it may be of great importance.

The MPP system uses two different techniques to ensure the anonymity of their users while using it, both based in the use of an API to encapsulate all the transaction processes. This way the system is able to retrieve both the user and the merchant information needed to perform the transaction separately, directly from each one of the parts, without exchanging any personal information. The two different techniques are OAuth and iFrame Cross-Domain AJAX (XDA in short), and both are explained with plenty of detail in the sections 5.2 and 5.3 of this report.

4.7 MPP User Panel

The MPP User Panel was born as a logical consequence of deciding to create an agile payment platform. The idea is to eliminate the part that makes most of the common Internet payment procedures so slow: the redirections. A way to do that could be to have a window integrated to the merchant web (not a pop-up window, mostly captured by anti-spam filters) which could present all the steps of a normal transaction, and some additional options. That simple idea has become the most important feature of the MPP system and also the most challenging part of the whole project. Moreover, it brings Internet payments to a new level of speed, versatility and usability, becoming the main marketing tool for the MicroPayment Platform.

As a result of the investigation on real-time data retrieval, the MPP User Panel has become the tool for implementing the iFrame Cross-Domain AJAX technique, explained in detail in section 5.3, allowing advanced usability. Its features are:

- It is JavaScript based, so it is compatible with most of the current web browsers (tested in Firefox 2 & 3, Internet Explorer 7 & 8, Google Chrome and Opera);
- It is merchant-independent, meaning that can be loaded in different merchants at the same time;
- It intrinsically provides anonymous payments, not giving any user information to merchants;

- Allows the user to:
  - See the balance of all his accounts in real-time;
  - See the transactions he is performing, confirming or canceling them, in real-time;
  - Set up (create, edit, delete and real-time cancel) oneclick configurations for the next transactions the user will perform in that specific merchant website;
  - See the history of transactions performed.

The technical implementation of the User Panel can be reviewed in section 8, while concrete examples in the testing models can be observed in section 13.2.

4.8 User and Merchant Areas

As most of the payment platforms over Internet, the MPP website has a specific area for users and merchants to configure all their settings, like their personal information, currency handling and Normal Oneclicks. It also allows them to view their account balances, perform transfers between MPP and PayPal accounts and view the history of transactions performed. To give the users a familiar interface the user area has also a similar look & feel than the user panel.

Merchants on the other hand do not have something like the user panel, but their area lets them have the same functionalities as a user have, not including Oneclick configurations obviously, but adding the specification of the currency-based payment options for their products or creating new payment services.

User area is explained with detail in section 6.1.2. Due to time constraints the Merchant area actually implemented in the system does not include the functionalities mentioned, although they can be activated directly from the MPP database.

5 Payment Authorization Modes

As described in section 4.6, one of the main concerns for an Internet user when performing on-line transactions is security. User authentication takes an important role to make the user feel secure while protecting him from impersonating attacks, so a lot of investigation and effort was needed in this project to provide the best authentication and third-party access (authorization) capabilities.

The basic way of performing transactions securely is by redirecting the user to the payment processor website, perform the transaction, and then redirect him back to the merchant website. In this case no authorization is needed from the user, as it is him who triggers the transaction and performs it in the website he trusts. This is the way PayPal and Google Checkout act, for example. However, this method is slow and assumes that the user does not trust the merchant. If he does, the logical procedure could be to let the merchant trigger transactions in the name of the user.

With the exponential proliferation of the social networks, based on accessing other people's content, it seems logical to point their authentication methods as the possible way to implement this kind of authorizations. After some investigation we found an emerging open protocol that is starting to be used, in its pure form or just based on it, on most of the main social networks such as Facebook, Twitter, Yahoo or Google Apps: the OAuth protocol. The companies behind those social networks have specific teams dedicated to implement and improve OAuth, websites dedicated to discuss about it, and plenty of examples of integration and implementation.

OAuth is a token-based authentication method used to allow users granting access to their data to third-party websites or services. It is simple, robust, and the amount of documentation related makes it reasonable easy to implement. Translated to the MPP case, it allows merchants to trigger transactions in the name of the users without having to redirect them every time they need to perform a transaction.

The first versions of the MPP, prior to the integration of the real-time user panel, relied on OAuth as the authentication method by using a PHP implementation based on the TwitterOAuth code by Abraham Williams[8], found in the OAuth website. However, the necessity to provide the user with real-time transaction capabilities and the implementation of the iFrame Cross-Domain AJAX technique modified the whole transaction flow, making specific authentication methods no more needed and thus limiting OAuth to situations in which the new technique is not implementable.

Therefore, as a result of the investigations on authorization schemes, the MPP system implements three different authorization modes, easily set up using the MPP API. Assuming that the user triggers a transaction (for example clicks on a "purchase" button) in any merchant website, the modes are:

Redirect Mode: The user is redirected to the MPP website, where he is presented all the information related to the transaction and has to choose whether to accept it or not. After the transaction is performed the user is redirected back to the merchant website. As it is the user who controls the transaction flow, no specific authorization has to be provided.

OAuth Mode: The MPP system starts the OAuth authentication process and checks if the user has a Oneclick configured for that merchant (meaning he trusts him). If he does it automatically lets the merchant perform the transaction using the required token. If he does not, it redirects the user to the MPP website and asks for authorization to let the merchant perform that transaction in the user name, giving the option to automatize further payments.

User Panel Mode: The user uses the MPP interactive User Panel (that uses the iFrame Cross-Domain AJAX technique) to confirm the transactions, not having to be redirected to any website (if he is logged into the MPP system) and performing the transactions directly in the MPP system without any merchant interaction (thus no authorization is needed).

The following sections will explain those modes in general terms. The technical implementation in the MPP system is explained in Part II of this report.

5.1 Redirect Mode

The Redirect authorization mode is no more than a direct redirection to the MPP website. When the user triggers a transaction, the merchant redirects him to a specific website in the MPP domain, including all the required data from the transaction. From that point, all the process is performed in the MPP website. The user is presented all the transaction data and asked to confirm it. Once done, the MPP system confirms the state of the transaction to the user and redirects him to the merchant website (even if the user canceled it), including the state of the transaction for further processing (shipping, for instance).

5.2 OAuth Mode

5.2.1 OAuth Specification

OAuth is the first feature that distinguishes the MPP payment system from the typical conception of an Internet payment. The specification allows users to share their private resources (photos, videos, contact list, bank accounts, etc.) stored on one site with another site without having to hand out their username and password credentials. To understand the problem clearly, giving an email account password to a social network site so they can look up the friends associated to it could be the same as going to a restaurant and giving the ATM card and PIN code to the waiter when it is time to pay. Any restaurant asking for a PIN code will go out of business, but when it comes to the web, users put themselves at risk sharing the same private information.

The ATM card is a good metaphor for OAuth from a user perspective. Instead of giving his ATM card and PIN code, the card can be used as a credit card with a signature authorization. Just like a username and password provide full access to the user resources, an ATM card and PIN code provide great control over his bank accounts, usually much more than just charging goods. But when the PIN code is replaced with the user's signature, the card becomes very limited and can only be used for limited access.

Instead of using a single site for all their online needs, users use one site for their photos, another for videos, another for email, and so on. In order to enable this kind of integration, sites need to access the user resources from other sites, and those are many times protects (private family photos, work documents, bank records), so they need a key to get in.

The key used by users is normally a combination of username and password, but this is too powerful and unrestrictive to share around. It also cannot be unshared once handed out except for changing it which will void access to every site, not just the one the user intends to block. OAuth addresses that by allowing users to hand out tokens instead. Each token grants access to a specific site (a video editing site) for specific resources (just videos from last weekend) and for a defined duration (the next 2 hours).

One of the best features of OAuth is that it is completely transparent to the users. In many cases the end-user will not know anything about OAuth, what it is or how it works. The user experience will be specific to the implementation of both the site requesting access and the one storing the resources, and adjusted to the device being used (web browser, mobile phone, PDA, set-top box).

A typical example is when a user wants to print a photo stored on another site. The interaction goes something like this: the user signs into the printer website and place an order for prints. The printer website asks which photos to print and the user chooses the name of the site where her photos are stored (from the list of sites supported by the printer). The printer website sends the user to the photo site to grant access. At the photo site the user signs into her account and is asked if she really wants to share her photos with the printer. If she agrees, she is sent back to the printer site which can now access the photos. At no point did the user share her username and password with the printer site.

It is important to understand that security and privacy are not guaranteed by the protocol. In fact, OAuth by itself provides no privacy at all and depends on other protocols to accomplish that (such as SSL). With that said, OAuth can be implemented in a very secure manner and the specification includes a good amount of security considerations to take include account when working with sensitive resources. Just like using passwords together with usernames to gain access, sites will use tokens together with secrets to access resources. And just like passwords, secrets must be protected.

5.2.2 Notation and Conventions

Service Provider:
A web application that allows access via OAuth.

User:
An individual who has an account with the Service Provider.

Consumer:
A website or application that uses OAuth to access the Service Provider on behalf of the User.

Protected Resource(s):
Data controlled by the Service Provider, which the Consumer can access through authentication.

Consumer Key:
A value used by the Consumer to identify itself to the Service Provider.

Consumer Secret:
A secret used by the Consumer to establish ownership of the Consumer Key.

Tokens:
A Token is generally a random string of letters and numbers (but not limited to) that is unique, hard to guess, and paired with a Secret to protect the Token from being abused. Tokens are used instead of User credentials to access resources. OAuth defines two different types of Tokens: Request and Access.

Request Token:
A value used by the Consumer to obtain authorization from the User, and exchanged for an Access Token.

Access Token:
A value used by the Consumer to gain access to the Protected Resources on behalf of the User, instead of using the Users Service Provider credentials.

Token Secret:
A secret used by the Consumer to establish ownership of a given Token.

OAuth Protocol Parameters:
Parameters with names beginning with *oauth*.

Timestamp:
Unless otherwise specified by the Service Provider, the timestamp is expressed in the number of seconds since January 1, 1970 00:00:00 GMT. The timestamp value MUST be a positive integer and MUST be equal or greater than the timestamp used in previous requests.

Nonce:
A random string uniquely generated for each request. The nonce allows the Service Provider to verify that a request has never been made before and helps prevent replay attacks when requests are made over a non-secure channel (such as HTTP).

5.2.3 Basic Schema

The basic schema, also called "Authentication Flow", has the following steps:

Obtaining a Request Token Phase

Step 1: A merchant (Consumer in the OAuth specification) registers to the Service Provider (from now on SP), who gives him a unique Consumer Key and Consumer Secret strings.

Step 2: When the Consumer wants to use the account from that User, it has to request to the SP a Request Token (no user-specific).

Obtaining User Authorization Phase

Step 3: When the Consumer receives the Request Token, it redirects the User to the SP login page with the Request Token and the callback URL.

Step 4: Once the User logs in the SP, he is asked to grant access to his account to the Consumer.

Step 5: If the user accepts to grant access, the SP marks the Request Token as User-authorized by that user.

Step 6: The SP redirects the browser back to the Consumer page, together with the Authorized Request Token.

Obtaining an Access Token Phase

Step 7: When the Consumer wants to have access to the User data, it asks the SP to exchange the Authorized Request Token for an Access Token. The SP checks all the data and returns the Access Token.

Accessing Protected Resources Phase

Step 8: Finally, the Consumer uses the Access Token to access the User data, without knowing his identity.

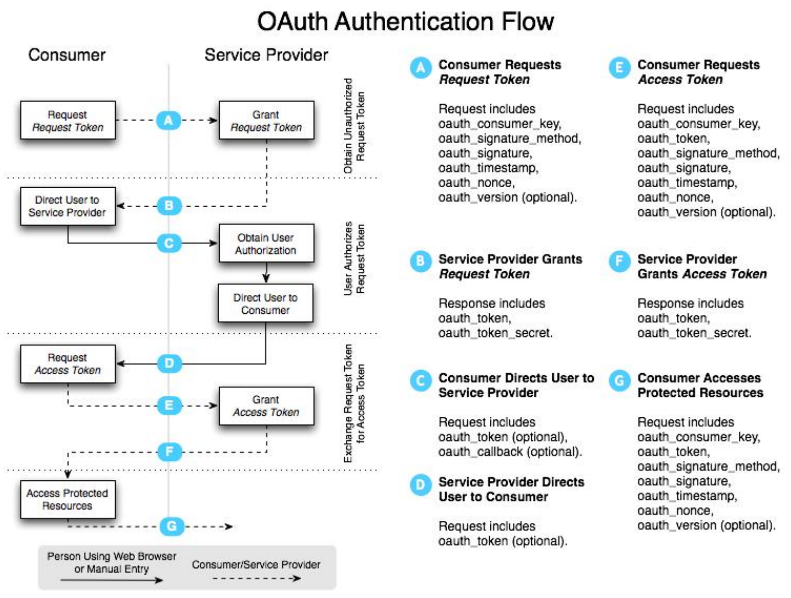

OAuth Authentication Flow
Consumer
Service Provider
Request Request Token
Grant Request Token
Obtain Unauthorized Request Token
Direct User to Service Provider
Obtain User Authorization
Direct User to Consumer
User Authorizes Request Token
Request Access Token
Grant Access Token
Exchange Request Token for Access Token
Access Protected Resources
A
B
C
D
E
F
G
Person Using Web Browser or Manual Entry
Consumer/Service Provider
A Consumer Requests Request Token
Request includes oauth_consumer_key, oauth_signature_method, oauth_signature, oauth_timestamp, oauth_nonce, oauth_version (optional).
B Service Provider Grants Request Token
Response includes oauth_token, oauth_token_secret.
C Consumer Directs User to Service Provider
Request includes oauth_token (optional), oauth_callback (optional).
D Service Provider Directs User to Consumer
Request includes oauth_token (optional).
E Consumer Requests Access Token
Request includes oauth_consumer_key, oauth_token, oauth_signature_method, oauth_signature, oauth_timestamp, oauth_nonce, oauth_version (optional).
F Service Provider Grants Access Token
Response includes oauth_token, oauth_token_secret.
G Consumer Accesses Protected Resources
Request includes oauth_consumer_key, oauth_token, oauth_signature_method, oauth_signature, oauth_timestamp, oauth_nonce, oauth_version (optional).

Figure 5: OAuth authentication flow

5.2.4 MPP OAuth

One of the main characteristics of the OAuth process when applied to the MPP is that as the content to access is always the same -the user account information- the implementation is exactly the same for every merchant, making it really easy to integrate. This procedure is explained with detail in section 7.

The following is a basic explanation on the specific implementation of OAuth in the MPP system. To make it easier to understand, the explanation has been personalized to the use of the web trivia game model.

Step 1: To be able to have access to a user account, www.trivia.com has to send a signed request to the MPP's Request Token server by using the public API (this is automatically done at the beginning of the user session, but can also be manually triggered). The API uses cURL calls to send the information to MicroPayments and waits for a response in JSON[1]format.

e.g.: http://micropayments.com/oauth/requestTokenServer.php?oauth_consumer_key=dpf43f3p2l4k3l03&oauth_signature_method=PLAINTEXT&oauth_signature=kd94hf93k423kf44%26&oauth_timestamp=1191242090&oauth_nonce=hsu94j3884jdopsl&oauth_version=1.0

Step 2: The MicroPayments' OAuth Server receives the request, checks the signature and the data in it, and returns the Unauthorized Request Token to www.php-trivia.com in JSON format.

e.g.: {oauth_token: "hh5s93j4hdidpola", oauth_token_secret: "hdhd0244k9j7ao03"}

The MicroPayments API retrieves the token information for www.php-trivia.com and saves it in the session cookie for further use (because it expires).

Step 3: When www.php-trivia.com wants to access a user account, he redirects the user to the MicroPayments authorization URL including the Unauthorized Request Token in the message.

e.g.: http://micropayments.com/oauth/authorize.php?oauth_token=hh5s93j4hdidpola

Step 4: MicroPayments authorization page checks the token and asks the user for granting access to his account to www.php-trivia.com, giving the possibility to add this merchant to his oneclick list (and so modifying the expiration of its tokens).

Steps 5 & 6: If the user grants access, MicroPayments returns the same Request Token than before, now considered Authorized.

e.g.: http://www.trivia.com/index.php?oauth_token=hh5s93j4hdidpola

Step 7: www.php-trivia.com request MicroPayments to exchange the request token for an access token. MicroPayments checks the request token and responds with the access token if valid.

e.g.: http://micropayments.com/oauth/accessTokenServer.php?oauth_consumer_key=dpf43f3p2l4k3l03&oauth_token=hh5s93j4hdidpola&oauth_signature_method=PLAINTEXT&oauth_signature=kd94hf93k423kf44%26hdhd0244k9j7ao03&oauth_timestamp=1191242092&oauth_nonce=dji430splmx33448&oauth_version=1.0

response: {oauth_token: "nnch734d00sl2jdk", oauth_token_secret: "pfkkdhi9sl3r4s00"}

Step 8: Now php-trivia can use the access token to request user data, by adding the required parameters to that request. In this case, the parameters are included as POST parameters in a cURL call to perform the transaction.

[1]JSON, short for JavaScript Object Notation, is a lightweight computer data interchange format. It is a text-based, human-readable format for representing simple data structures and associative arrays (called objects). Example: {"firstName": "John", "lastName": "Smith", "address": { "streetAddress": "21 2nd Street", "city": "New York", "state": "NY", "postalCode": "10021" }, "phoneNumbers": [{ "type": "home", "number": "212 555-1234" }, { "type": "fax", "number": "646 555-4567" }] }

e.g.:product=question&amount=0.2&oauth_consumer_key=dpf43f3p2l4k3l03&oauth_token= nnch734d00sl2jdk&oauth_signature_method=HMAC-SHA1&oauth_signature=tR3 %2BTy81lMeYAr%2FFid0kMTYa%2FWM%3D&oauth_timestamp=1191242096 &oauth_nonce=kllo9940pd9333jh&oauth_version=1.0

5.2.5 OAuth Oneclick

OAuth allows the merchant to perform transactions in the name of the user. The OAuth specification includes a user authorization step, which grants that permission to the merchant by using the *Access Token*, but most importantly it allows to set a specific validity to that token. Knowing that, the MPP system allows the user to set that validity to a specific time. This feature is called OAuth Oneclick and it is available during the authorization process in the OAuth flow.

If using OAuth Mode, when receiving a OAuth transaction the MPP system will check the OAuth Oneclick options for that user. If they are still valid the transaction will be performed automatically, but if not the system will proceed to the authorization phase of the OAuth flow.

5.3 User Panel Mode

The User Panel is the main result of all the investigation and effort in this project. It is in this mode where the MicroPayments Platform shows its full potential, bringing the typical redirection scheme to a new level of realtime experience and creating a new conception of agile payments.

5.3.1 The iFrame Cross-Domain AJAX (XDA) Technique

OAuth allows users to grant permission to third parties to their content. In the MPP case, it allows merchants to trigger transactions in the name of the users. Depending on the level of security to implement, the access token may have a shorter or longer expiration. As the Authentication Flow explained, every time a merchant needs to perform a transaction it needs that access token, so depending on its expiration the full OAuth process will need to be executed.

OAuth can be, then, a slow process. While implementing the first version of the MPP (which only contemplated the e-shop model) it was a great way to authenticate the merchant, but when implementing the web game model, with much more quantity of transactions, the desirability of real-time transaction triggering emerged.

5.3.2 AJAX

From just some years ago, talking about real-time content access in Internet means one word: AJAX. The core of the so called Web 2.0, AJAX stands for Asynchronous JavaScript and XML, and it is a group of interrelated web development techniques used on the client side to create interactive web applications or rich Internet applications. The main characteristic is that using Ajax web applications can retrieve data from the server asynchronously in the background without interfering with the display and behavior of the existing page. Its capabilities were demonstrated widely when Google implemented it in its mail and maps applications, and since then its use has grown exponentially.

However, Ajax is bound to the *same origin policy*, which prevents scripting languages the access to most methods and properties across pages on different sites, meaning that a web cannot perform Ajax calls to a server in another domain, and that was exactly what I needed for implementing the real-time transaction triggering. To be able to use Ajax functionalities I had to overcome the *same origin policy*, and after some investigation I found the solution in the iFrame Cross-Domain AJAX technique[6].

5.3.3 Bypassing the *Same Origin Policy*

An iFrame is an HTML element that can embed an HTML document inside another one. The source of the embedded page can be on another domain, although because of the *same origin* *policy* the content of the page in an iFrame can be accessed by another one only if the two pages are on the same domain. However, in any case the parent of an iFrame (the page that contains it) has access to the URL of the iFrame, and more particularly to the fragment identifier (the string at the end of the URL starting by #). One of the particularities of the fragment identifier is that when changing it the page does not reload its contents.

Knowing this, the way to bypass the *same origin policy* starts by using an iFrame on the domain of the MPP server. This iFrame is by definition able to do AJAX requests to the MPP server, as it is in the same domain. The result of those requests can then be forwarded to the fragment identifier of a child iFrame on the domain of the merchant. This iFrame will be periodically looking at its fragment identifier to see if changes have been made, and as soon as a new message is received it will forward it to a callback function described in the merchant's page (which is possible because the iFrame is in the same domain as the main page of the merchant).

The following is a schema summarizing this mechanism:

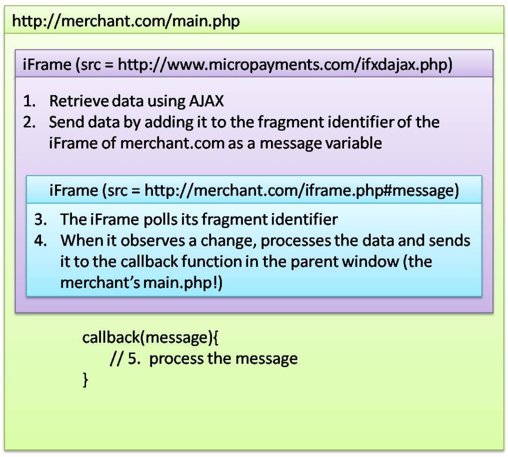

http://merchant.com/main.php
iFrame (src = http://www.micropayments.com/ifxdajax.php)
1. Retrieve data using AJAX
2. Send data by adding it to the fragment identifier of the iFrame of merchant.com as a message variable
iFrame (src = http://merchant.com/iframe.php#message)
3. The iFrame polls its fragment identifier
4. When it observes a change, processes the data and sends it to the callback function in the parent window (the merchant's main.php!)
callback(message){
// 5. process the message
}

Figure 6: XDA Messaging scheme

By transforming the iFrame pointing to the MPP system to a draggable JavaScript window and creating a messaging protocol encapsulated in JavaScript API functions we could create an interactive control panel. This way the most interesting and powerful functionality of this project was born: the real-time User Panel. The nature of the technique makes the real-time interaction possibilities between merchants and the MPP system unlimited, becoming one of the most versatile and usable Internet payment methods. Moreover, it could be easily applicable to other Internet services: social networks, mailing systems, organizers. . . The options are enormous.

But the advantages of using this technique in the MPP system are not limited to the real-time triggering of transactions, or the new User Panel. One really amazing consequence of using this technique is that there is no more need to grant permission to the merchant. As it will be explained in section 8, the iFrame in the figure above pointing to the MPP's "ifdajax.php" will become the User Panel, and it basically load user data from the MPP server. For doing that the user will need to be logged in the MPP system, meaning that the iFrame will contain all the information the MPP system needs to perform transactions without letting the merchant know any of it, as it is only loaded in the web browser of the user. In addition, the MPP system will load the payment service data for the merchant that is embedding the user panel, so there is no way to create fake data.

However, the nature of the XDA messaging together with the public API utilization creates a possible security flaw in the MPP system: a merchant phishing. It was detected in the designing phase and solved with a three-step initialization phase. Its explanation and the way to overcome it are explained in the next subsection.

5.3.4 Merchant Phishing Attack

EXAMPLE:

A malicious person wishes to compromise the reputation of a public figure (e.g. a politician) by making him purchase a doubtful item (e.g. a porno film from his workplace) without him realizing it, and show it to the press.

The malicious person sets up a website that looks like a e-commerce site where people can buy flowers. He manages to "attract" the politician guy to this website during working hours, maybe with a very special promotion. The fake e-commerce site loads the MPP user panel in its iframe. However, the user panel is loaded with the `merchant id` and `payment service id` of a porno website (for more details about how the public API works see section 9). If the politician does not pay attention to this, he may browse the fake e-commerce website and select to buy some flowers. The fake e-commerce website will inform the MPP user panel that the customer wants to buy a porno item. Then the MPP panel will ask the politician if he agrees with the purchase. If the politician does so without paying attention, he will have bought the doubtful item. The malicious person will then be able to show to the press that the politician guy bought a doubtful item from his workplace.

PROTECTION:

The problem is that a fake website can load in its iframe the MPP user panel of any merchant. To protect the system from the attack, we need to enforce that a MPP user panel can be loaded only in it's vendor website.

Because of the web browser cross-domain protection, in theory the MPP user panel cannot know the URL of the parent frame where it is loaded. However, a simple way to achieve this, and making use of the already framework, is to send a "testVendorIdentity" message with a random code to the parent frame trough the panel's sendMessage function. If the parent frame belongs to the announced vendor, the iframe in the user panel will be allowed to transmit the message and the parent frame will receive it. The parent frame needs to forward this message again to the user panel. If the user panel receives back the message with the same code, it has successfully determined that the parent frame corresponds to its vendor. This test needs to be done only once at the beginning, when the panel is loaded, and it successfully protects from the phishing attack.

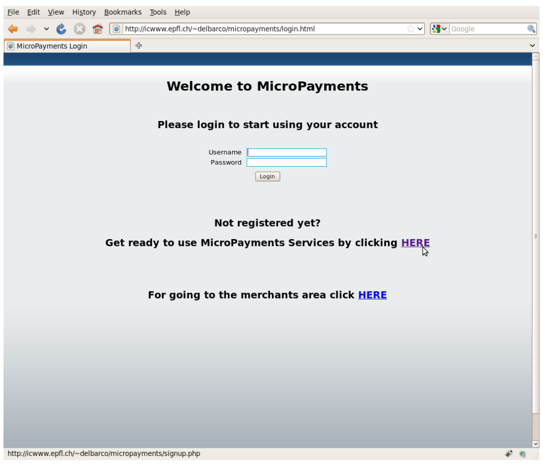
6 Web Interface

Although constructing a professional-looking website was not in the scope of the project, the MPP needed at least a very basic web interface to test all the system capabilities, including a simple signup and a specific area where users and merchants could configure the functionalities. However, it was when the XDA messaging was adopted and the User Panel was born that we spent some more time trying to get at least an approximation on what we thought if should be its interface.

The next subsections will show the idea of what each part of the MPP website should be, as well as the simplified prototypes I created.

6.1 Micropayments Web Page

As a e-commerce business centered in the payment area, the Micropayments webpage should be informative but always concise, clear and usable. The home website should give information on the main functionalities the system offers, have a signup area where new users and merchants can register to use the MPP services, and a login area where registered users can access their personal area.



http://icwww.epfl.ch/~delbarco/micropayments/login.html

Welcome to MicroPayments

Please login to start using your account

Username
Password

Not registered yet?

Get ready to use MicroPayments Services by clicking HERE

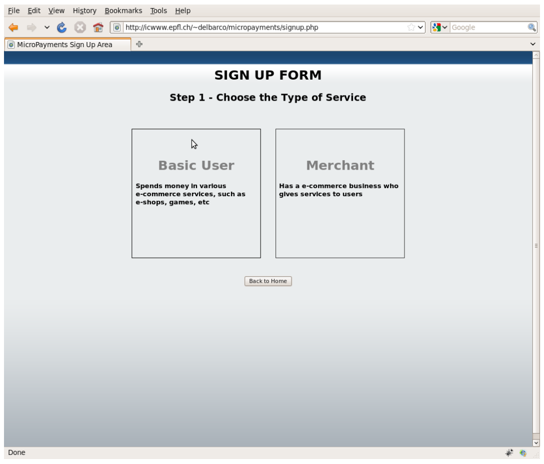
For going to the merchants area click HERE

http://icwww.epfl.ch/~delbarco/micropayments/signup.php

Figure 7: Micropayments main web page

6.1.1 Signup

The MPP signup is a wizard-like registration form that separates into two different processes depending on what kind of user wants to register: a basic user or a merchant (see fig 8).


http://icwww.epfl.ch/~delbarco/micropayments/signup.php

SIGN UP FORM

Step 1 - Choose the Type of Service

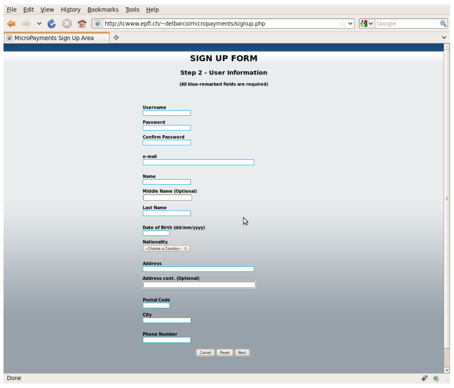
Basic User
Spends money in various e-commerce services, such as e-shops, games, etc

Merchant
Has a e-commerce business who gives services to users

File Edit View History Bookmarks Tools Help
Google
MicroPayments Sign Up Area
Back to Home
Done

Figure 8: Signup - Step 1

The next step will be to collect the user or merchant information, save it and provide the required information to the user.


http://icwww.epfl.ch/~delbarco/micropayments/signup.php

SIGN UP FORM

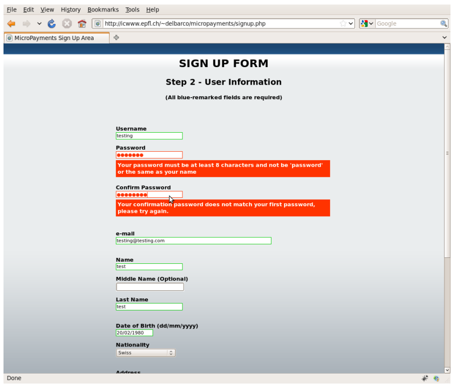
Step 2 - User Information

(All blue-remarked fields are required)

Username
Password
Confirm Password
e-mail
Name
Middle Name (Optional)
Last Name
Date of Birth (dd/mm/yyyy)
Nationality
--Choose a Country--
Address
Address cont. (Optional)
Postal Code
City
Phone Number

File Edit View History Bookmarks Tools Help
Google
MicroPayments Sign Up Area
Cancel
Reset
Next
Done

(a) Step 2 - Personal Information


http://icwww.epfl.ch/~delbarco/micropayments/signup.php

SIGN UP FORM

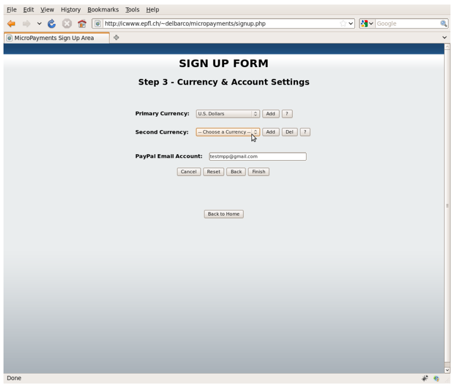
Step 2 - User Information

(All blue-remarked fields are required)

Username
testing
Password
●●●●●●●
Your password must be at least 8 characters and not be 'password' or the same as your name
Confirm Password
●●●●●●●●
Your confirmation password does not match your first password, please try again.
e-mail
testing@testing.com
Name
test
Middle Name (Optional)
Last Name
test
Date of Birth (dd/mm/yyyy)
20/02/1980
Nationality
Swiss
Address

File Edit View History Bookmarks Tools Help
Google
MicroPayments Sign Up Area
Done

(b) Data validation


http://icwww.epfl.ch/~delbarco/micropayments/signup.php

SIGN UP FORM

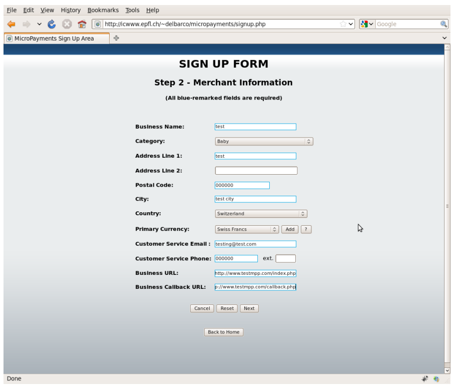
Step 3 - Currency & Account Settings

Primary Currency: U.S. Dollars
Second Currency: -- Choose a Currency --
PayPal Email Account: testmpp@gmail.com

File Edit View History Bookmarks Tools Help
Google
MicroPayments Sign Up Area
Add
?
Add
Del
?
Cancel
Reset
Back
Finish
Back to Home
Done

(c) Step 3 - Account settings

Figure 9: Basic User Signup


File Edit View History Bookmarks Tools Help
http://icwww.epfl.ch/~delbarco/micropayments/signup.php
Google
MicroPayments Sign Up Area
Cancel
Reset
Next
Back to Home
Done

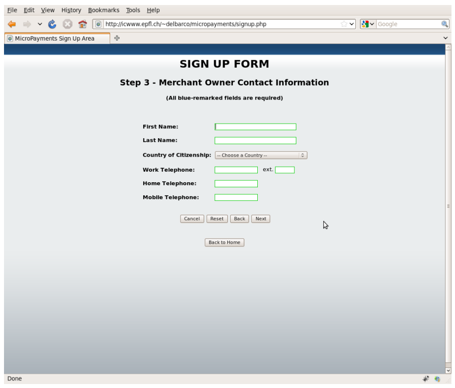
SIGN UP FORM

Step 2 - Merchant Information

(All blue-remarked fields are required)

Business Name:	test
Category:	Baby
Address Line 1:	test
Address Line 2:	
Postal Code:	000000
City:	test city
Country:	Switzerland
Primary Currency:	Swiss Francs
Customer Service Email :	testing@test.com
Customer Service Phone:	000000 ext.
Business URL:	http://www.testmpp.com/index.php
Business Callback URL:	p://www.testmpp.com/callback.php

(a) Step 2 - Merchant Information


File Edit View History Bookmarks Tools Help
http://icwww.epfl.ch/~delbarco/micropayments/signup.php
Google
MicroPayments Sign Up Area
Cancel
Reset
Back
Next
Back to Home
Done

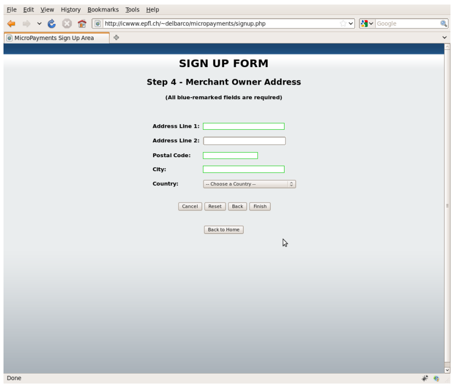
SIGN UP FORM

Step 3 - Merchant Owner Contact Information

(All blue-remarked fields are required)

First Name:	
Last Name:	
Country of Citizenship:	-- Choose a Country --
Work Telephone:	ext.
Home Telephone:	
Mobile Telephone:	

(b) Step 3 - Owner Information


File Edit View History Bookmarks Tools Help
http://icwww.epfl.ch/~delbarco/micropayments/signup.php
Google
MicroPayments Sign Up Area
Cancel
Reset
Back
Finish
Back to Home
Done

SIGN UP FORM

Step 4 - Merchant Owner Address

(All blue-remarked fields are required)

Address Line 1:	
Address Line 2:	
Postal Code:	
City:	
Country:	-- Choose a Country --

(c) Step 4 - Owner Address

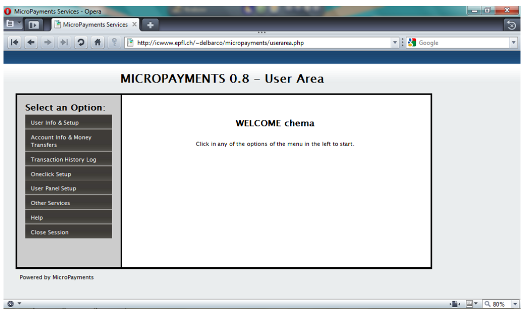
Figure 10: Merchant Signup

The last step implies providing the user or merchant with information about how to use the system. In the case of merchants, it also provides the parameters and files required for the system integration.

6.1.2 User Area

The user area is intended to provide the same functionalities as the User Panel and some additional ones. It has a similar look & feel to ensure the user understands how to use it.

As the following pictures show, it has a left side menu with the different options and configurations that the user can set up:



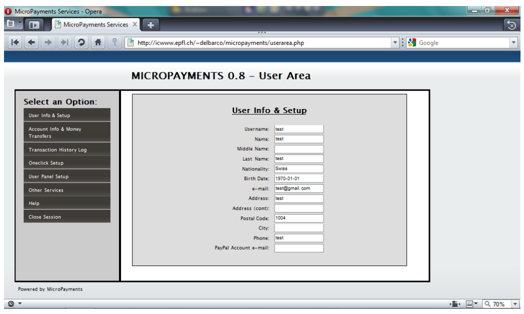
MICROPAYMENTS 0.8 – User Area

WELCOME chema

Click in any of the options of the menu in the left to start.

Powered by MicroPayments

(a) User Area panel



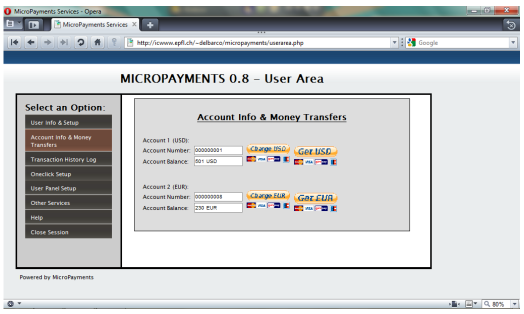
MICROPAYMENTS 0.8 – User Area

User Info & Setup

Username:	test
Name:	test
Middle Name:	
Last Name:	test
Nationality:	Swiss
Birth Date:	1970-01-01
e-mail:	test@gmail. com
Address:	test
Address (cont):	
Postal Code:	1004
City:	
Phone:	test
PayPal Account e-mail:	

Powered by MicroPayments

(b) User Info & Setup



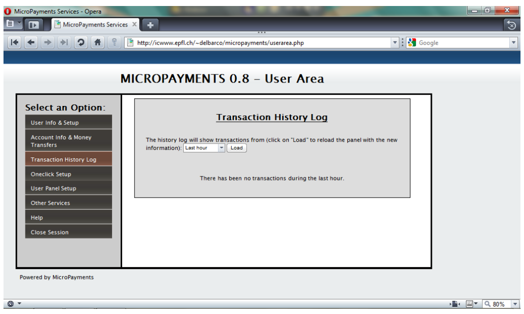
MICROPAYMENTS 0.8 – User Area

Account Info & Money Transfers

Account 1 (USD):
Account Number: 000000001
Account Balance: 501 USD

Account 2 (EUR):
Account Number: 000000008
Account Balance: 230 EUR

(a) Account Info & Money Transfers



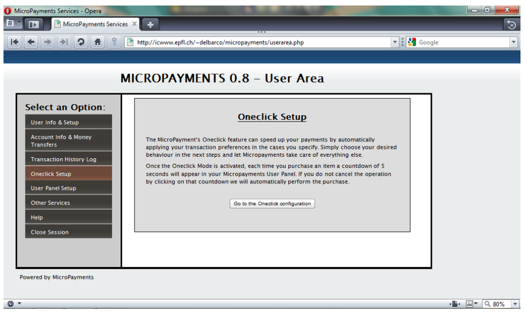
MICROPAYMENTS 0.8 – User Area

Transaction History Log

The history log will show transactions from (click on "Load" to reload the panel with the new information): Last hour

There has been no transactions during the last hour.

(b) Transaction History Log



MICROPAYMENTS 0.8 – User Area

Oneclick Setup

The MicroPayment's Oneclick feature can speed up your payments by automatically applying your transaction preferences in the cases you specify. Simply choose your desired behaviour in the next steps and let Micropayments take care of everything else.

Once the Oneclick Mode is activated, each time you purchase an item a countdown of 5 seconds will appear in your Micropayments User Panel. If you do not cancel the operation by clicking on that countdown we will automatically perform the purchase.

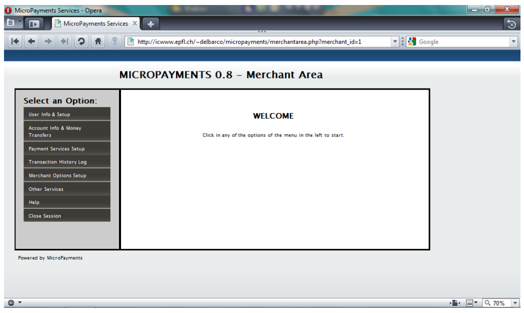
(c) Oneclick Setup

Figure 11: User Area options

The *User Info & Setup* option lets the user view and modify his personal information. The *Account Info & Money Transfers* provides the user his account numbers and balances, as well as a way to transfer money between the PayPal accounts and the MPP Virtual Accounts. The *Account History Log* option shows a history of all the transactions that the user has performed and their state (the user can specify the time window of the history). The *Oneclick Setup* option lets the user configure Normal Oneclicks as explained in section 4.5. The last options, called *User Panel Setup*, *Other Services* and *Help* are meant for future features of the MPP system and thus are not implemented yet. Finally, the *Close Session* option lets the user log out from the MPP website safely.

6.1.3 Merchant Area

As explained before, the merchant area is not fully implemented due to time constraints of the project. However, the basic features it should include are shown in the following capture:



MICROPAYMENTS 0.8 – Merchant Area

WELCOME

Click in any of the options of the menu in the left to start.

Figure 12: Merchant Area Options (prototype)

The option differing from the User Area is the Payment Service Setup, where the merchant is able to configure new Payment Services, edit or delete the existing ones, and configure the payment options for them.

Part II
Technical Implementation

The MPP is intended to be a service to be used by users, but more importantly, by third-parties or merchants. To make merchants attracted to using the MPP it has to be as simple and fast as possible. The way to achieve this is by using public APIs (Application Programming Interface), which encapsulate internal functionalities into simple, argument-based functions, as well as using a given set of files that include all the code needed to make everything work.

A lot of effort has been made in this project to make the MPP integration as simple as possible. A combination of PHP and JavaScript has been chosen as the programming languages, as they are probably the most used technologies in Internet and thus most merchant's developers will have the knowledge required to use them. However, for reaching as many merchants as possible further implementations in other languages are proposed as future enhancements, out of the scope of this project.

The first step in the early implementation of the MPP was to implement the OAuth protocol, which is explained in section 7. However, as explained before, the main functionalities of the project were born after applying the XDA technique, which is explained in section 8. The MPP integration in merchant websites by using the public JS API is explained in section 9, while the internal transaction server is explained in section 10, together with the PayPal integration in section 11. Section 12 shows a brief explanation on the database structure used by the system. Finally, section 13 describes the integration process of the MPP system as well as the e-commerce models used to test it.

7 OAuth Implementation

As explained before, OAuth is a well documented, easy to implement protocol. The specification gives plenty of hints on how to get the best results of it, even about how to add a security layer to the protocol communication. In addition, the website offers a variety of implementation examples in almost every programming language, providing a good starting point for integration.

From those examples I based my implementation on the TwitterOAuth PHP classes [8], as PHP is the language I am more familiar with, and because the core of the MPP server is also written in it. The following is a detailed explanation on the classes involved and the modifications made in the original ones.

7.1 OAuth Package

The OAuth package defines the main classes to request and process OAuth tokens. As the client side (merchant) does not need to process the tokens, only the OAuth package in the MPP server includes those classes, and will be explained in the subsection 7.5. For differentiation purposes the OAuth package for the merchant has been renamed to OAuthMerchant, while the package for the MPP server is still named OAuth. Here is a description of the classes in the OAuthMerchant package:

- **OAuthConsumer**
  Defines a consumer by its key and secret strings.
- **OAuthToken**
  Defines a token by its key and secret strings, and provides a way to format them following the OAuth specifications.
- **OAuthSignatureMethod (OauthSignatureMethod_PLAINTEXT, OAuthSignatureMethod_HMAC_SHA1 and OauthSignatureMethod_RSA_SHA1)**
  Defines the signature method for the OAuth signature parameter. The first one provides a general checker. The other ones implements the different signature methods the specification allows.
- **OAuthRequest**
  Retrieves (from header data or by passed variables) and prepares all the parameters needed for performing an OAuth request to an OAuth server.
- **OAuthException**
  An implementation of a basic Exception interface.
- **OAuthUtil**
  Basically a codec for the rfc3986 coding standard that the OAuth specification uses for transmitting its parameters.

7.2 MPPOAuth Class

The MPPOAuth class uses the OAuthMerchant package to provide the tools to implement the OAuth flow. Its methods are:

__construct ($consumer_key, $consumer_secret, [$oauth_token = NULL], [$oauth_token_secret = NULL])
Constructor. If the consumer data is passed it means the goal is to get an access token, if not means getting a request token.

void requestTokenURL()
URL to the request token server.

void accessTokenURL()
URL to the access token server.

void authorizeURL()
URL to the authorizations server.

string getAuthorizeURL($token, [$callback = NULL])
Accessor for the authorizeURL parameter.

array getRequestToken()
Retrieves a request token from the MPP request token server.

array getAccessToken()
Retrieves an access token from the MPP access token server by exchanging the previously authorized request token for it.

string oAuthRequest($url, [$args = array()], [$method = NULL])
Method used by getRequestToken and getAccessToken to retrieve the tokens from the MPP servers. It uses cURL calls with the oauth parameters passed by GET or POST (preferred). The actual implementation uses POST.

7.3 MPPOAuth API

As it can be observed, the code as found in the *OAuthMerchant* and *MPPOAuth* classes is still too low level to be used easily. For this reason I created the *MPPOAuth* API file, which encapsulates most of the OAuth process into simple functions, making it able to easily implement the whole OAuth request flow. Moreover, with the MPPOAuth API I transformed this request flow into a pseudo-state machine, much easier to handle. By monitoring a self-created variable called *oauth_state*, the whole OAuth process can be performed almost automatically and just using a few lines of code.

The following is a brief explanation of the MPPOAuth API:

<u>CONSTANTS</u>
CONSUMER_KEY
CONSUMER_SECRET

<u>FUNCTIONS</u>

getMpOAuthState()
Gets the oauth_state variable, i.e. the current state of the OAuth process. Can be:

“start”: implies asking for a request token;

”authorized“: implies the user has authorized the operation (the request token has been authorized);

“ready”: implies the access token has been retrieved and it’s prepared to make transactions;

”error“: implies an error in the process.

`mpOAuthRequest()`
Gets the Request Token from MicroPayments. It returns an array with the request token key and authorization link (link that the user has to follow to grant access to his contents). It also saves in the session cookie the token values, as well modify the state of the OAuth process.

`mpOAuthAccess()`
Checks the session cookie for information on the access token. If it doesn't find any it tries to retrieve the access token from MicroPayments. It returns an array with the oauth_access_token key and secret.

`getAuthorization($link)`
Redirects the user to the link `$link`, which is the link generated by `mpOAuthRequest` to ask the user for granting access.

`clearMpOAuthSession()`
Clears the OAuth session correctly.

`transaction($type, $product=''Not Specified'', $amount)`
Makes an OAuth authorized call to the user account and performs transaction for the passed amount. Valid values for `$type` are *"deposit"* and *"withdraw"*.

7.4 Merchant OAuth Integration

Once the MPP OAuth process is encapsulated in its own API the only thing left is to implement the OAuth request flow for a given merchant. As explained in the previous subsections, the OAuth process has been implemented using the `oauth_state` parameter as the trigger for the retrieval of the tokens. This way the full schema of the OAuth process can be easily implemented by monitoring that `oauth_state`.

To limit the number of files the MPP integration needs the OAuthMerchant package, the MPPOAuth class and the MPPOAuth API have been included in a single file named `Micropayments.php`.

The following is an example of integration. By loading this file the merchant can start the OAuth request flow, which will finally get in the *'ready'* `oauth_state`. Once the process is finished and the user is returned back to the merchant's page, the merchant can call the specific API function to perform a transaction (see section 10.2), which will call this file and pass the required type, product and `currencyOptions` variables. For the record, this file is known always as `micropayments.php`.

```php
<?php

require_once 'classes/Micropayments.php';

//get request variables
$oauth_state = isset($_REQUEST['oauth_state'])?$_REQUEST['oauth_state']:'start';
$type = isset($_REQUEST['type'])?$_REQUEST['type']:"";
$product = isset($_REQUEST['product'])?$_REQUEST['product']:"";
$currencyOptions =
    isset($_REQUEST['currencyOptions'])?$_REQUEST['currencyOptions']:"";
$callback = isset($_REQUEST['callback'])?$_REQUEST['callback']:"";

switch ($oauth_state) {
        case 'start':
        $request = mpOAuthRequest($callback);
                if(\$request['authorization_link']!=""){
                        getAuthorization($request['authorization_link']);
                }
                break;
        case 'authorized':
                $access_token = mpOAuthAccess();
                if($oauth_state != 'ready'){
                        break;
                }
        case 'ready':
                if($type != "" && $currencyOptions != ""){
                        transaction($type,$product,$currencyOptions);
                }
                break;
        case 'error':
                echo 'error';
                break;
}
?>
```

Here is an explanation following the OAuth flow explained in section 5.2.4:

Obtaining a Request Token
Steps 1 & 2:
The first check to perform when the file loads is the current state of the process. The first state is "start", which means that the merchant server will try to fetch a Request Token from the MPP server.

```
$request = mpOAuthRequest($callback);
```

This creates the `MPPOAuth` object and calls the method `getRequestToken`. This method makes a cURL call to MicroPayment's `requestTokenServer.php` with all the required oauth parameters. The server, after authenticating that the request really comes from the merchant, will generate and return the Unauthorized Request Token in JSON format.

Obtaining User Authorization
Steps 3 & 4:
When the merchant receives a valid request token he can generate the URL for asking user authorization to MicroPayments and redirect the user:

```
if($request['authorization_link']!=""){
        getAuthorization($request['authorization_link']);
}
```

Following the OAuth specifications, the authorization request must include the `oauth_token` and optionally the callback URL. All this data is available at the moment the URL is constructed.

When the link is generated, the merchant is able to redirect automatically the User to that link. As it needs access from the beginning, he does it automatically once the link is available.

The MicroPayments authorization server checks the validity of the request token by:

1. Checking the signature of the request;
2. Checking the nonce parameter for that token in its database.

If the nonce is found it means the request token has already been used, so it is not valid (and probably is a wrong reply or even a "reply attack"). If everything is ok, it asks the user to:

1. Login (if it does not find his information in the session cookie);
2. Grant or Decline access to his account to the merchant.

Steps 5 & 6:
If the user grants access, MicroPayments marks this token as "authorized". This marking is implicit by simply returning to `micropayments.php` in the merchant server, as if the user declines the access the OAuth protocol does not allow to let go back to the merchant. The MicroPayments server also sets the parameter `oauth_state` to "authorized" and redirects the User back to the merchant's `micropayments.php` page.

Obtaining an Access Token
Step 7:
The `micropayments.php` page from the merchant creates another request, this time including the `request_token`:

```
$access_token = mpOAuthAccess();
```

This generates a request call to MicroPayments' `accessTokenServer.php` with all the required parameters (see fig. 5). MicroPayments checks the signature of the token and its timestamp value (to see if it has expired), and if everything is correct it generates an access token and returns it to the merchant, who saves it in the session cookie or in a database for further use.

Step 8:
From this point the merchant can trigger transactions in the name of the user. For doing that he has to call the specific function of the API, that will call the merchant's `micropayments.php` to build a signed request using the access token, the type of transaction to perform and the amount to charge, the product information and the payment options:

```
if($type != "" && $currencyOptions != ""){
        transaction($type,$product,$currencyOptions);
}
```

7.5 OAuth Token Servers

To be able to process tokens, the MPP server has to be able to retrieve the parameters from the calls performed from the `micropayments.php` file (using the API), check they are correct, and create the tokens. This is easily achieved by using the full `OAuth` package, as it includes the classes needed to create a token server. The classes are:

- **MySQLDataStore**
  Is a MySQL connector to retrieve tokens from and save tokens to the MPP database.
- **OAuthServer**
  Has all the methods needed to verify OAuth requests, meaning checking parameters, and comparing signatures. It uses the previous MySQL connector to get and put all the data.

7.6 OAuth Authorizations

The final part of the OAuth process is the authorization. The authorization server from MPP is a basic web interface that uses an `OAuthServer` object to check the authorization request from the merchant's `micropayments.php` (the same way the token server do), check if the user is logged into the MPP system (allowing him to do it if he is not), and show him a message asking to grant the merchant permission to perform transactions with his account. The user can then grant or decline access, being redirected to the same merchant's website he was before if he grants permission, or staying in the MPP website if he declines.

8 XDA & User Panel Implementation

In section 5.3 the XDA technique is presented and explained, including an overview of the implementation, as it is needed to completely understand how it works. Here I will show how the MPP combines the XDA technique with an attractive interactive User Panel, transforming the MPP payment system in a real-time, configurable, fast and reliable platform, unique in its conception.

As explained before, XDA relies in iFrames and hash queries to be able to bypass the *same origin policy*. The following is a technical explanation on how to implement XDA using them.

First of all, for easiness of the explanation I will specify some definitions:

- The specific website of the merchant, in the domain www.merchant.com, that needs to send XDA messages will be known as "XDATop";
- The specific website of the MPP server, in the domain www.micropayments.com, that will receive the XDA messages will be known as "XDAFrame";

MERCHANT DOMAIN SIDE
To implement XDA, the XDATop website needs:

1. An HTML iFrame pointing to the XDAFrame website. ex:

```
<iframe id='micropayments'
    src='http://www.micropayments.com/xdampp.php'></iframe>
```

2. A JavaScript function to send XDA messages to the XDAFrame website and identify them by the message count. ex:

```
sendMessageToMPPUserPanel : function(msg) {
        $('micropayments').src =
            "http://www.micropayments.com/xdampp.php#messageId=" +
            (++this.lastMessageId) + "&message=" + encodeURIComponent(msg);
}
```

To establish a bidirectional communication, the merchant needs a separated HTML file, which I will call XDAMerchantReceiver, with:

1. A JavaScript function that periodically checks the hash of its URL to look for new XDA messages;
2. A JavaScript function to process the XDA messages. ex:

```
<html>
<head></head>
<body>

<script type="text/javascript">
// variable that store the last message id
var previousId = -1;

// regexp to extract the message and message id from the hash
var reg = /^#messageId=(.+)&message=(.*)$/;

window.setInterval(function() {
        // get the message in the hash of the URL
        var idAndMessage = decodeURIComponent(window.location.hash);

        // extract the id and the message
        var id = idAndMessage.replace(reg, '$1');
        var message = idAndMessage.replace(reg, '$2');

        //make sure it is a new message
        if(previousId != id) {
                // change the previous id
                previousId = id;

                // forward the new message to the callback function
                window.top.ps.callback(message);
        }
}, 50); //check for new message every 50ms
</script>
</body>
</html>
```

<u>MPP DOMAIN SIDE</u>
The XDAFrame website needs:

1. A JavaScript function that periodically checks the hash of its URL to look for new XDA messages (same as the merchant side);
2. A JavaScript function to process the XDA messages;

And to establish a bidirectional communication:

1. An HTML iFrame pointing to the XDAMerchantReceiver website. ex:

```
<iframe id="getMessageFromMicropaymentsIframe"
    src="http://www.merchant.com/xdaMerchantReceiver.php" style="height: 0;
    width: 0; position: absolute; top: 0; left: 0;"></iframe>
```

2. A JavaScript function to send XDA messages to the XDAMerchantReceiver website (and identify them by the message count). ex:

```
sendMessageToMPPUserPanel : function(msg) {
        $('micropayments').src =
            "http://www.micropayments.com/xdaMerchantReceiver.php#messageId="
            + (++this.lastMessageId) + "&message=" + encodeURIComponent(msg);
}
```

As it can be observed in the figure 13, this combination of iFrames closes a loop between the merchant side and the MPP side, creating a bidirectional communication.

If we take a deeper look to the figure we can easily realize that both `XDATop.html` and `XDAFrame.html` could be any website inside their respective domains, always as long as they meet the requirements imposed by the XDA technique. Another conclusion is that the interaction between both domains is limited exclusively to XDA messages (which are specified by the MPP system), making it secure and potentially anonymous for the user.

Now, if we create specific XDA messages for triggering actions in the XDAMPP iFrame, like for modifying its content to show specific information, we can establish a pseudo-interactive API for giving merchants the capability of interact with the user that it is using the system without knowing any information about him, while being the MPP system the one controlling the communication between both.

By adding a very usable interface, transforming a normal iFrame in a draggable window, and giving the user some extra configurations, like the ability to configure automatic transactions, the XDAMPP iFrame becomes the best feature of the MPP system: the User Panel. A complete example of utilization of the User Panel will be introduced in sections 10.3 and 13 of this report.

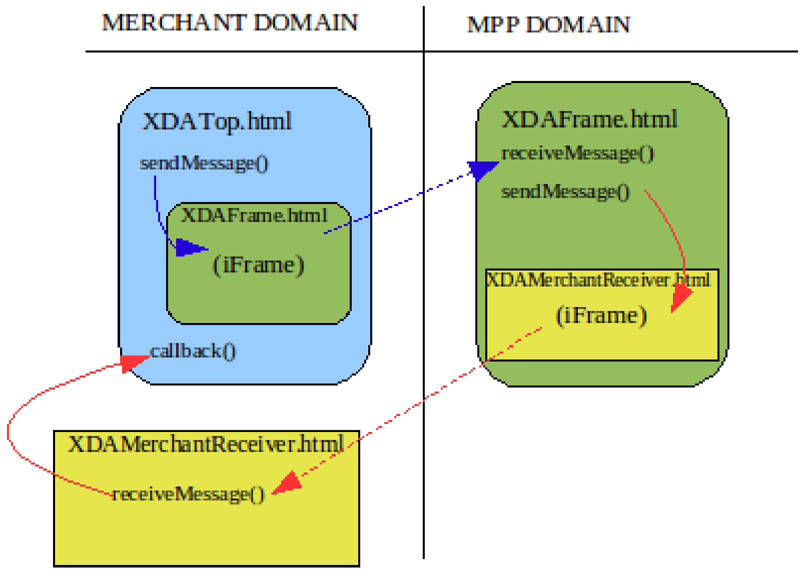


Figure 13: Scheme of the XDA iFrame configuration

9 JavaScript Public API

To understand the main capabilities of the MPP system, in particular the merchant integration using the XDA/User Panel combination, it is first needed to understand the MPP public API.

The use of JavaScript for the public API is explained by the necessity to provide users with a highly usable environment, with pseudo-real-time information retrieval, and without having to reload constantly the webpage they are viewing. All this reasons point to Ajax scripting, and thus to the use of JS.

9.1 Prototype JavaScript Framework

While researching some easy ways to perform Ajax calls I came up with the Prototype JavaScript Framework. This JS package provides not only an easy way to encapsulate Ajax requests and responses, but also a simplified DOM access and a pseudo-class system (JavaScript is not Object Oriented), together with some other useful accessors and functions. By using Prototype the public MPP API became Object Oriented, and thus more intuitive, while the internal programming of the API was simplified.

9.2 PaymentService Pseudo-Class

The PaymentService is a prototype pseudo-class which contains the core of the MPP public API. It encapsulates all the functionalities merchants can integrate in their websites: OAuth, XDA/User Panel, currency payment options, the MPP button and obviously transaction triggering.

The class constructor retrieves from the MPP server the payment service data associated to that merchant and payment service id provided, and transforms it into class attributes.

9.2.1 Attribute List

- Retrieved from the MPP Database:

  `psid`: id of the payment service.

  `merchantId`: id of the merchant associated to that payment service.

  `merchantName`: name of the merchant as it is in the MPP database.

  `typeId`: id of the type of payment service (1 = 'game', 2 = 'e-shop', 3 = 'token-time', 4 = 'token-bw', 5 = 'custom').

  `descriptor`: description of the payment service (by the merchant).

  `acceptedCurrencies`: 'CHF','EUR','USD','GBP' or combination of them.

  `priceCurrencyList`: custom equivalences for currency-dependent prices, in JSON format.

  `baseURL`: root of the URL of the merchant.

  `userPanelContainerName`: name of the HTML element (typically a DIV) that will contain the User Panel.

  `userPanelFloat`: flag to indicates if the User Panel can be moved in-site like a window.

- Generated by the constructor:

  `userPanel`: object containing the user panel.

  `lastMessageId`: message checker for the XDA technique.

  `merchantCallbackName`: name of the merchant-defined callback function associated to receiving an XDA message.

  `callbackURL`: URL from where the API is accessed. Passed by the merchant.

`authorizeURL` **(private):** URL pointing to the MPP authorize server, adding the `merchantid` and the `psid` parameters.

`userPanelURL` **(private):** URL pointing to the MPP's `userpanel.php` file, adding the `merchantid`, the `psid` and the `callbackURL` parameters.

9.2.2 Method List

INITIALIZATION

- `initialize(userPanel, paymentServiceId, merchantCallbackName, callbackURL)`
  Constructor. As the PaymentService object is generated externally to the MPP server, Ajax is bound to the *same origin policy*, and XDA still cannot be used. Then, to retrieve the data the constructor:

  1. Uses Ajax to call the file `micropayments.php` in the merchant server, passing the `psid` that the merchant wants to load;
  2. `micropayments.php` performs a cURL call to `loader.php` in the MPP server, telling it to load the `PaymentService` data associated to the `psid` passed. The server responds with a JSON string containing all the data;
  3. `micropayments.php` echoes the JSON response;
  4. The constructor gets the response from micropayments and processes it, initializing the class attributes fetched from the MPP server and generating the rest.

- `initService(type)`
  Gets the type of payment service and acts accordingly. This function is a helper for the different payment services that the MPP system should be able to handle. For instance, if the payment service is a web game, the cadence of transactions is supposed to be higher. If the user has to confirm every transaction the pace of the game can be broken, so a Oneclick configuration is recommended. Then, in the webgame case this function will check if a oneclick configuration is being used, and suggest the user (by a message in the User Panel) to create it if it is not.

- `showMicropaymentsButton(htmlElementID)`
  Allows the merchant to specify the HTML element which will contain the MPP button to initialize the MPP payment system (explained in section 10), and creates it.

XDA HANDLING

- `sendMessageToMPPUserPanel(message)`
  Internal function used to send XDA messages to the MPP User Panel. The messages are intended to trigger events such as transaction confirmation. They are JSON formatted and API specific.

- `callback(message)`
  Internal function called when the merchant receives a XDA message from the User Panel. Basically passes the message to the merchant-defined callback function.

USER PANEL HANDLING

- `initUserPanel()`
  If the merchant decides to use the User Panel feature, it must be shown when the user comes back from authorization. This function retrieves the GET variable that indicates that and loads the User Panel window.

- `loadUserPanel()`
  It transforms the `userPanelContainerName` HTML container into an interactive window by using the code from Sebastien Gruhier [2], based also on the Prototype framework and a visual effects extension called Script.aculo.us [1]. The same function is used to toggle the visualization of the User Panel when clicking on the MPP button in the merchant website.

- `showUserPanel()`
  Shows the user panel.

- `hideUserPanel()`
  Hides the user panel.

- `forcePanelShow()`
  Checks if the User Panel is visible (not minimized, closed or hidden by other windows). If not if shows it. Useful when triggering transactions, as the user has to know it.

<u>TRANSACTION TRIGGERING</u>

- `setPriceList(priceList)`
  Lets the merchant specify a JSON formatted list of equivalent prices in different currencies. I also allows to specify the preferred currency for the payment.

  Example:

```
var newList = '{
  "list": [
      {"currency": "EUR", "price": "10", "default": "0"},
      {"currency": "USD", "price": "15", "default": "1"},
      {"currency": "CHF", "price": "15", "default": "0"}
  ]
}';
```

- `confirm(product, priceList)`
  This is the main function for triggering purchase-related transactions. If the merchant specified to use OAuth it will trigger the internal function `buy`. If not, it ensures that the User Panel is loaded. If it is, it sends the specific message to the User Panel to make the user confirm a purchase of the product with the price options specified in either the `priceList` parameter passed or the default price list that the merchant specified when creating the payment service.

- `deposit(amount)`
  Internal function that will trigger an OAuth deposit transaction, i.e., that the merchant will transfer money to the user.

- `buy(product, priceList)`
  Internal function that will trigger an OAuth purchase transaction, basically passing the required parameters to the `micropayments.php` file.

<u>OAUTH HANDLING</u>

- `loadOAuth()`
  Starts the OAuth process by redirecting the user to the merchant's `micropayments.php`, which as it has been explained in section 7.4 will start the OAuth request flow.

- `ClearOAuthSession()`
  Clears the session data specific to the OAuth process. Once done, the whole OAuth request flow will have to be run in order to be able to trigger OAuth-based transactions again.

9.3 API Loading

To be able to use the MPP API the merchant needs to include the file `micropayments.js` and `micropayments.css` from the MPP server in the HTML head. Once done, a simple call to create the `PaymentService` pseudo-class with its initial parameters (`OAuthEnabled, paymentServiceId, merchantCallbackName, and callbackURL`) will load all the required data to be able to use all the MPP functionalities, including OAuth, the User Panel handling and the transaction triggering. If he wants, the merchant can also specify a callback function in JavaScript for the XDA messages received from the MPP User Panel. If he does not, the system will use a void function instead (”do nothing“).

Example:

```
<script type="text/javascript"
    src="http://www.micropayments.com/js/micropayments.js"></script>
<link href="http://www.micropayments.com/css/micropayments.css" rel="stylesheet"
    type="text/css" />

<script type="text/javascript">

//Merchant defined callback
function callback(message){
  switch(message.action){
    case 'transaction_ok':
      $('confirm').src = 'confirm.php?action=TRANSACTION_OK&amount='+m.amount;
      $('confirm').src += '&currency='+m.currency+'&product='+m.product;
      $('confirm').show();
      setTimeout('$("confirm").hide()',5000);
      break;
    }
}

//Initialice Micropayments System
var callbackURL = 'http://www.merchant.com/show_product.php?product=<?php echo
    $product['id']; ?>';
var ps = new PaymentService(true,"test2",callback,callbackURL);

</script>
```

From this point, the merchant can use the API functions by using the pseudo-object `ps`.

Example:

```
//define currency option list
var priceList = '{  list  : [{
    currency  :  EUR  ,
     price  :  10  ,
     default  :  0
}]}';
ps.setPriceList(priceList);
```

Example 2:

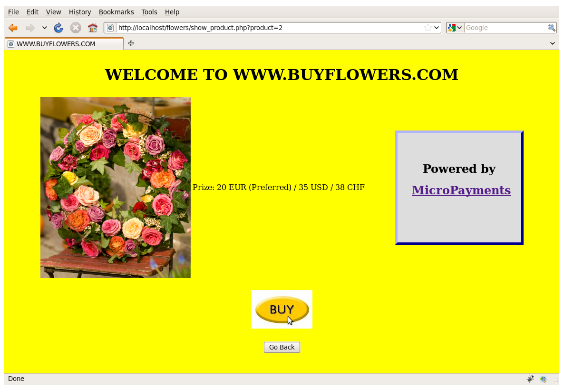
```
//Load the MP button when the DOM is loaded
window.onload = function(){
        ps.showMicropaymentsButton('mpButtonLoad');
}
```

10 Transaction Processing

As explained in the previous sections, the MPP system uses Virtual Accounts to perform transactions between users and merchants. Depending if the system is using OAuth or not, the processing of the transactions implies more or less check filters, although they end in the same transaction server. The following is an explanation of the two different ways of handling transactions that the MPP offers: OAuth and User Panel (XDA). In both, it is assumed that the MPP's `PaymentService` pseudo-object has been properly initialized.

10.1 Redirect Mode

The basic transaction process is similar to the one used by most of the actual payment processors (like PayPal). It redirects the user to the MicroPayments website adding the payment options of the product selected.

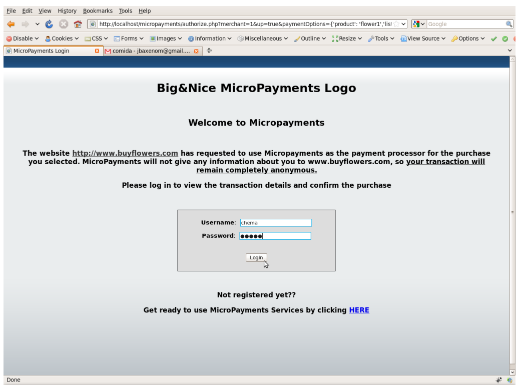

http://localhost/flowers/show_product.php?product=2

WELCOME TO WWW.BUYFLOWERS.COM

Prize: 20 EUR (Preferred) / 35 USD / 38 CHF

Powered by MicroPayments

(a) User clicks on the "Buy" button in the merchant website


http://localhost/micropayments/authorize.php?merchant=1&up=true&paymentOptions={'product': 'flower1','list

Big&Nice MicroPayments Logo

Welcome to Micropayments

The website http://www.buyflowers.com has requested to use Micropayments as the payment processor for the purchase you selected. MicroPayments will not give any information about you to www.buyflowers.com, so your transaction will remain completely anonymous.

Please log in to view the transaction details and confirm the purchase

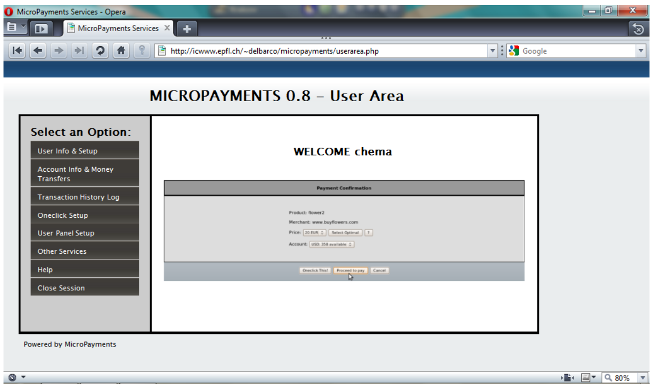
Username: chema

Password: ●●●●●

Not registered yet??

Get ready to use MicroPayments Services by clicking HERE

(b) User is redirected to the MPP website, where is asked to login if he is not


http://icwww.epfl.ch/~delbarco/micropayments/userarea.php

MICROPAYMENTS 0.8 – User Area

Select an Option:

- User Info & Setup
- Account Info & Money Transfers
- Transaction History Log
- Oneclick Setup
- User Panel Setup
- Other Services
- Help
- Close Session

WELCOME chema

Payment Confirmation

Product: flower2

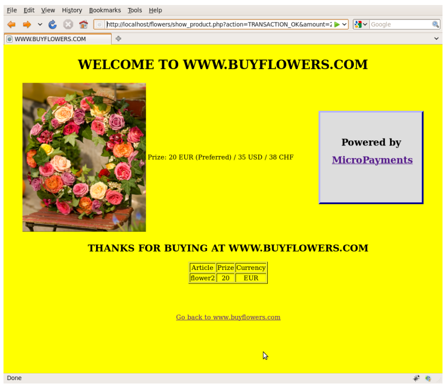
Merchant: www.buyflowers.com

Price: 20 EUR

Account: USD: 358 available

Powered by MicroPayments

(c) Once logged in, the user area is shown, with a new panel showing the transaction details and the confirmation button


http://localhost/flowers/show_product.php?action=TRANSACTION_OK&amount=2

WELCOME TO WWW.BUYFLOWERS.COM

Prize: 20 EUR (Preferred) / 35 USD / 38 CHF

Powered by MicroPayments

THANKS FOR BUYING AT WWW.BUYFLOWERS.COM

Article	Prize	Currency
flower2	20	EUR

Go back to www.buyflowers.com

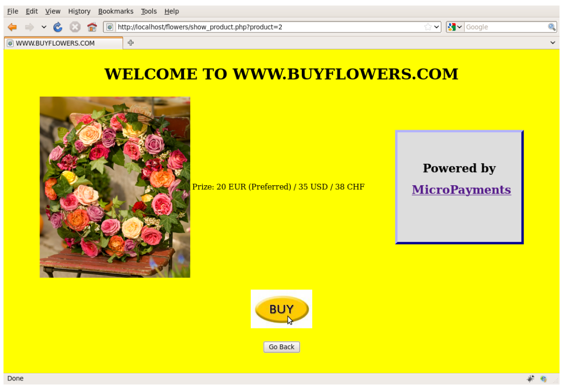
(d) The MPP performs the transaction and redirects the user back to the merchant website, where it is confirmed

Figure 14: Redirect Mode transaction flow

10.2 OAuth Mode

While using OAuth, transactions are triggered by the `micropayments.php` file in the merchant server. This file redirects the user to the website `process.php` in the MPP server, while adding the OAuth parameters required as `GET` variables. Process.php creates an OAuthServer object to validate the request, and if valid it presents the information of the transaction to the user, who has to accept or deny the transaction. At this point the user is able to configure an automatic granting of transactions, which will allow the full potential of the OAuth as the system will not need to redirect the user to the MPP website, but only check the validity of the OAuth data and perform the transaction without asking the user.

The following pictures show the OAuth transaction handling in the e-shop model:



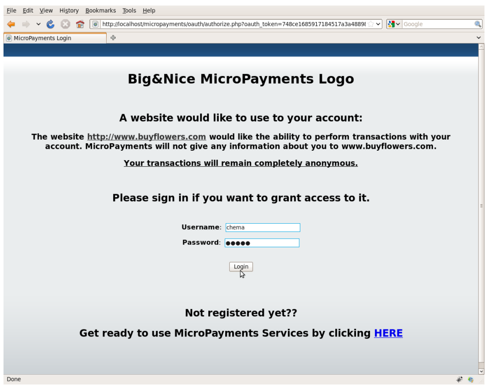
http://localhost/flowers/show_product.php?product=2

WELCOME TO WWW.BUYFLOWERS.COM

Prize: 20 EUR (Preferred) / 35 USD / 38 CHF

Powered by MicroPayments

(a) User clicks on the "Buy" button in the merchant website



http://localhost/micropayments/oauth/authorize.php?oauth_token=748ce1685917184517a3a4889

Big&Nice MicroPayments Logo

A website would like to use to your account:

The website http://www.buyflowers.com would like the ability to perform transactions with your account. MicroPayments will not give any information about you to www.buyflowers.com.

Your transactions will remain completely anonymous.

Please sign in if you want to grant access to it.

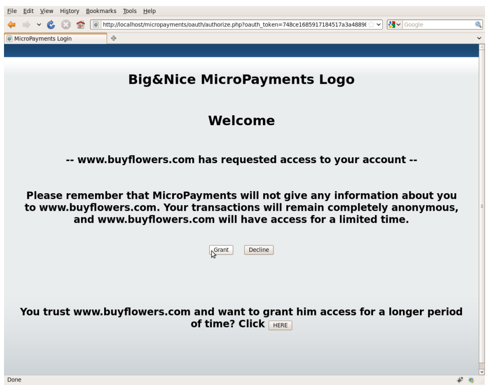
Username: chema

Password: ●●●●●

Not registered yet??

Get ready to use MicroPayments Services by clicking HERE

(b) User is redirected to the MPP website, where is asked to login if he is not



http://localhost/micropayments/oauth/authorize.php?oauth_token=748ce1685917184517a3a4889

Big&Nice MicroPayments Logo

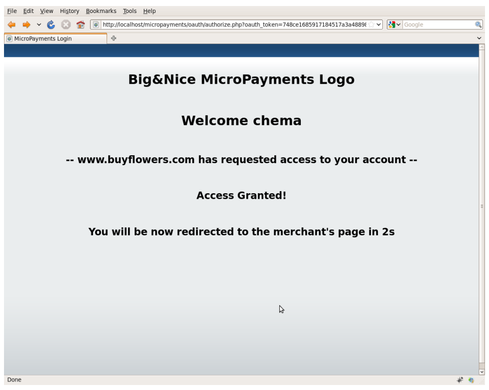
Welcome

-- www.buyflowers.com has requested access to your account --

Please remember that MicroPayments will not give any information about you to www.buyflowers.com. Your transactions will remain completely anonymous, and www.buyflowers.com will have access for a limited time.

You trust www.buyflowers.com and want to grant him access for a longer period of time? Click HERE

(c) Once logged in, the user is asked to grant access to www.buyflowers.com. He can also set up an OAuth Oneclick.



http://localhost/micropayments/oauth/authorize.php?oauth_token=748ce1685917184517a3a4889

Big&Nice MicroPayments Logo

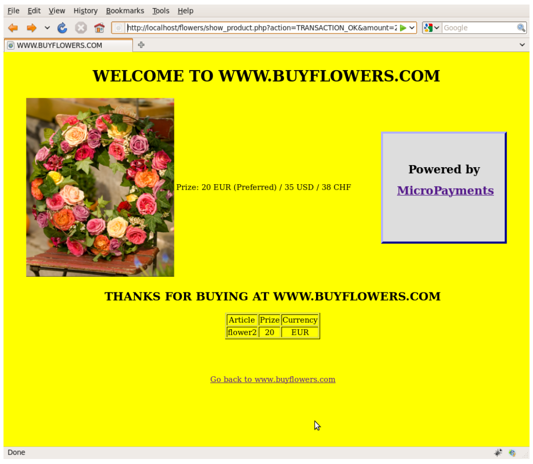
Welcome chema

-- www.buyflowers.com has requested access to your account --

Access Granted!

You will be now redirected to the merchant's page in 2s

(d) The user clicks in the "Grant" button and is redirected back to the merchant's website after 3 seconds


File Edit View History Bookmarks Tools Help
http://localhost/flowers/show_product.php?action=TRANSACTION_OK&amount=2
Google
WWW.BUYFLOWERS.COM
Powered by
MicroPayments
Go back to www.buyflowers.com
Done

WELCOME TO WWW.BUYFLOWERS.COM

Prize: 20 EUR (Preferred) / 35 USD / 38 CHF

THANKS FOR BUYING AT WWW.BUYFLOWERS.COM

Article	Prize	Currency
flower2	20	EUR

Figure 15: (e) The transaction is performed. From that moment every time the user clicks on ”Buy” again the process repeats from step (c). If the user sets up an OAuth Oneclick the process goes directly to step (e) (no redirection).

10.3 User Panel Mode

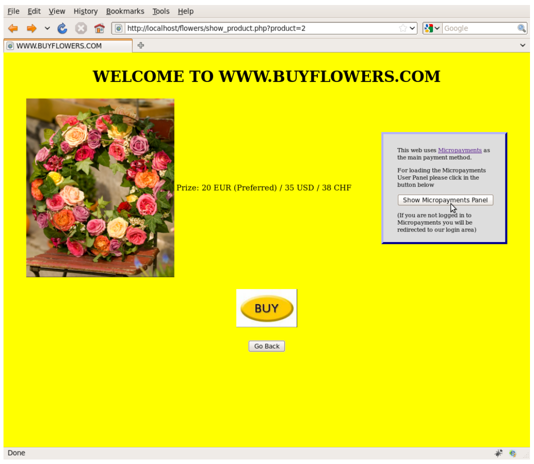
If the system uses XDA messages as transaction triggers the flow is completely different, as it is completely handled by the MPP API.

Obviously, for the XDA to work the User Panel has to be loaded in the user web browser. To do that a specific MicroPayments button is shown in the merchant website. Clicking on it triggers an MPP API call to check if the user is logged in the MPP server. If not, the user is automatically redirected to a specific MPP login area where he is alerted about the steps to follow and, once logged in, he is redirected back to the same merchant page he was before. The User Panel is then automatically loaded and shown, and the user can start using the MPP payment system. Note that if the user was already logged in the MPP server, clicking on the MPP button in the merchant website will directly load and show the User Panel, without the necessity to redirect him to the MPP website.

Whenever the merchant wants to trigger a transaction (usually when the user clicks on an specific button or link) he has to call the `confirm` API function and pass to it the `product` parameter and optionally the `priceList` parameter. This function creates the XDA message to alert the User Panel of a new transaction. From that point the User Panel takes control of everything, showing the user the information related to the new transaction, letting him choose whether to accept or cancel it and, if accepting it, which payment option should apply from the ones that the merchant has specified. The user has unlimited time to choose those options and finish the transaction procedure, although for security reasons the User Panel only handles one transaction at a time, alerting the user in case other transactions are trying to be performed before finishing the current one. All the rest of functionalities of the User Panel are available while the transaction is still pending, but any new Oneclick configuration will apply after the current transaction has been finished, and not on it. Also while confirming the transaction, the User Panel gives the user the chance to configure an Express Oneclick, as explained in section 4.5.

Once the user has confirmed the transaction, the User Panel sends an XDA confirmation message to the merchant. As those messages are standard, the merchant can set up his own callback functions to act accordingly to the transaction confirmation message. If so, he must have included the name of the callback function when initializing the MPP Payment Service (view section 9.3). If no callback function was specified then, the MPP system will not do anything with the transaction confirmation.

The following pictures show the XDA transaction handling in the same e-shop model than the previous section:


File Edit View History Bookmarks Tools Help
http://localhost/flowers/show_product.php?product=2
WWW.BUYFLOWERS.COM
BUY
Go Back
Done

WELCOME TO WWW.BUYFLOWERS.COM

Prize: 20 EUR (Preferred) / 35 USD / 38 CHF

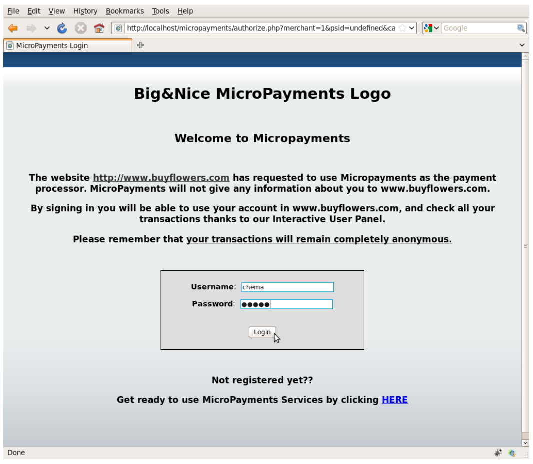
This web uses Micropayments as the main payment method.

For loading the Micropayments User Panel please click in the button below

Show Micropayments Panel

(If you are not logged in to Micropayments you will be redirected to our login area)

(a) User clicks on the MPP button


File Edit View History Bookmarks Tools Help
http://localhost/micropayments/authorize.php?merchant=1&psid=undefined&ca
MicroPayments Login
Done

Big&Nice MicroPayments Logo

Welcome to Micropayments

The website http://www.buyflowers.com has requested to use Micropayments as the payment processor. MicroPayments will not give any information about you to www.buyflowers.com.

By signing in you will be able to use your account in www.buyflowers.com, and check all your transactions thanks to our Interactive User Panel.

Please remember that your transactions will remain completely anonymous.

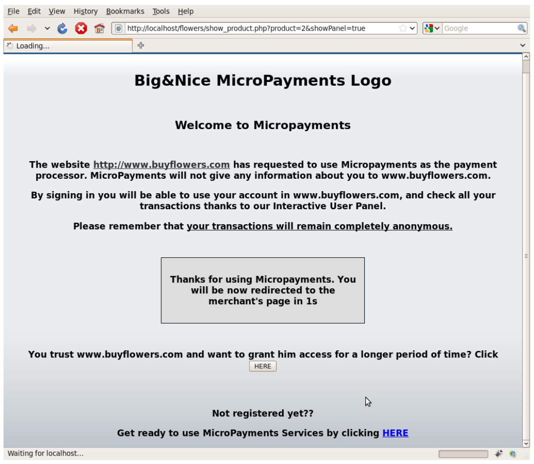
Username: chema
Password: ●●●●●

Login

Not registered yet??

Get ready to use MicroPayments Services by clicking HERE

(b) If not logged in, user is redirected to the MPP login area


File Edit View History Bookmarks Tools Help
http://localhost/flowers/show_product.php?product=2&showPanel=true
Loading...
Waiting for localhost...

Big&Nice MicroPayments Logo

Welcome to Micropayments

The website http://www.buyflowers.com has requested to use Micropayments as the payment processor. MicroPayments will not give any information about you to www.buyflowers.com.

By signing in you will be able to use your account in www.buyflowers.com, and check all your transactions thanks to our Interactive User Panel.

Please remember that your transactions will remain completely anonymous.

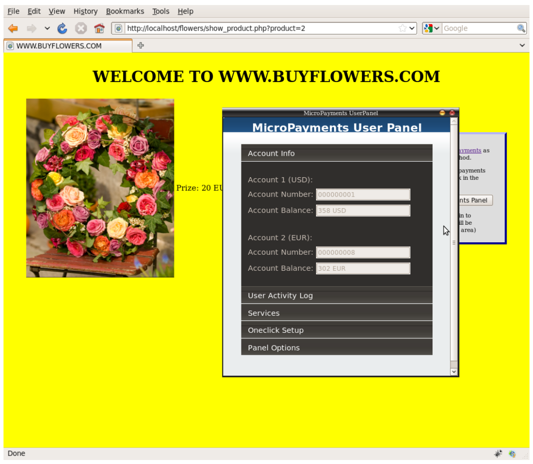
Thanks for using Micropayments. You will be now redirected to the merchant's page in 1s

You trust www.buyflowers.com and want to grant him access for a longer period of time? Click HERE

Not registered yet??

Get ready to use MicroPayments Services by clicking HERE

(c) When logged in, user is redirected back to the Merchant website


File Edit View History Bookmarks Tools Help
http://localhost/flowers/show_product.php?product=2
WWW.BUYFLOWERS.COM
WELCOME TO WWW.BUYFLOWERS.COM
MicroPayments UserPanel
Done

MicroPayments User Panel

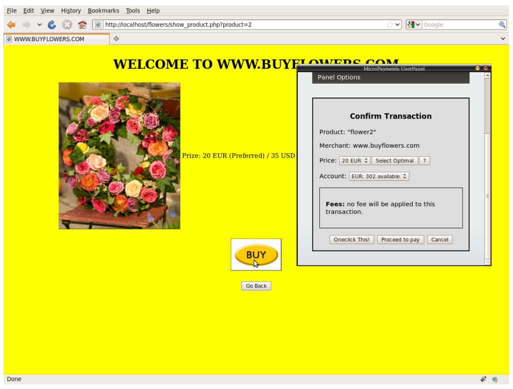
Account Info

Account 1 (USD):
Account Number: 000000001
Account Balance: 358 USD

Account 2 (EUR):
Account Number: 000000008
Account Balance: 302 EUR

User Activity Log
Services
Oneclick Setup
Panel Options

(d) The User Panel loads with all the user data


File Edit View History Bookmarks Tools Help
http://localhost/flowers/show_product.php?product=2
WWW.BUYFLOWERS.COM
WELCOME TO WWW.BUYFLOWERS.COM
Prize: 20 EUR (Preferred) / 35 USD
BUY
Go Back
MicroPayments UserPanel
Panel Options
Done

Confirm Transaction

Product: "flower2"

Merchant: www.buyflowers.com

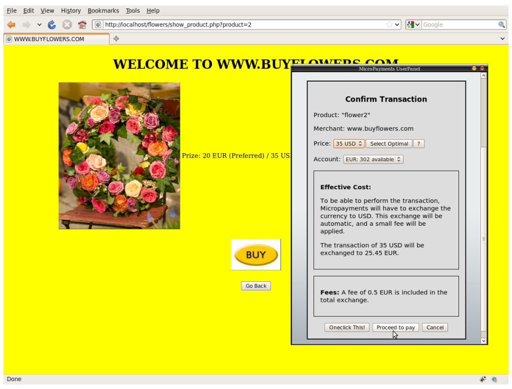
Price: 20 EUR | Select Optimal | ?

Account: EUR: 302 available

Fees: no fee will be applied to this transaction.

Oneclick This! | Proceed to pay | Cancel

(e) If user clicks on "Buy" the User Panel shows the confirmation panel


File Edit View History Bookmarks Tools Help
http://localhost/flowers/show_product.php?product=2
WWW.BUYFLOWERS.COM
WELCOME TO WWW.BUYFLOWERS.COM
Prize: 20 EUR (Preferred) / 35 US
BUY
Go Back
MicroPayments UserPanel
Done

Confirm Transaction

Product: "flower2"

Merchant: www.buyflowers.com

Price: 35 USD | Select Optimal | ?

Account: EUR: 302 available

Effective Cost:

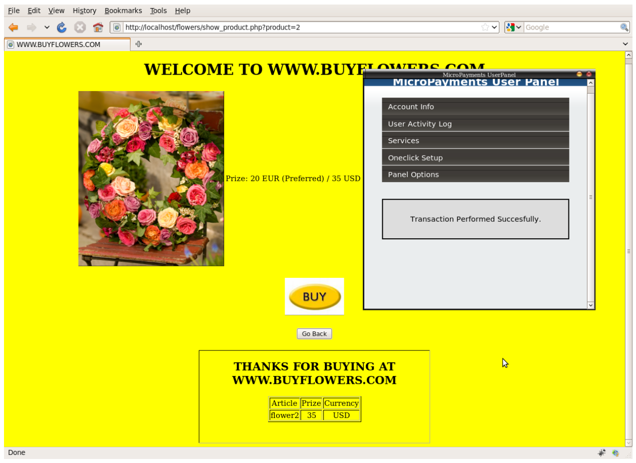
To be able to perform the transaction, Micropayments will have to exchange the currency to USD. This exchange will be automatic, and a small fee will be applied.

The transaction of 35 USD will be exchanged to 25.45 EUR.

Fees: A fee of 0.5 EUR is included in the total exchange.

Oneclick This! | Proceed to pay | Cancel

(f) User can choose from the different payment options the Merchant offers



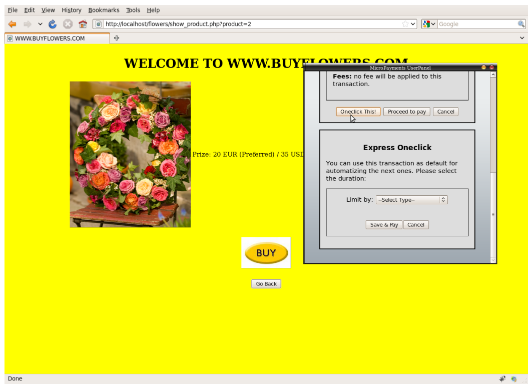
Transaction Performed Succesfully.

THANKS FOR BUYING AT WWW.BUYFLOWERS.COM

Article	Prize	Currency
flower2	35	USD

Figure 16: (g) User clicks on ”Proceed to pay“, the User Panel performs the transaction and confirms the state to both user and merchant

XDA ONECLICK TRANSACTION FLOW:



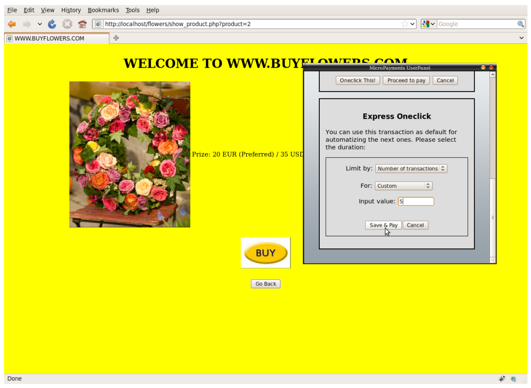
Fees: no fee will be applied to this transaction.

Express Oneclick

You can use this transaction as default for automatizing the next ones. Please select the duration:

Limit by: --Select Type--

(a) User clicks on ”Buy” button and selects the ”Oneclick This!” option



Express Oneclick

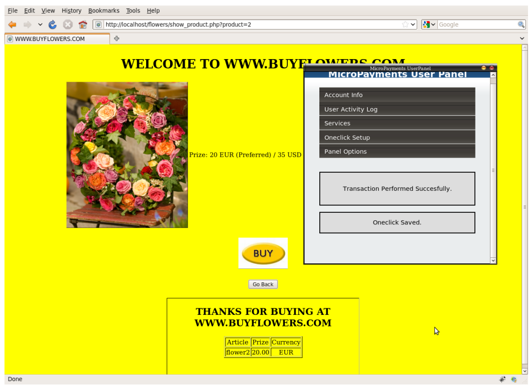
You can use this transaction as default for automatizing the next ones. Please select the duration:

Limit by: Number of transactions

For: Custom

Input value: 5

(b) User selects the validity of the Oneclick



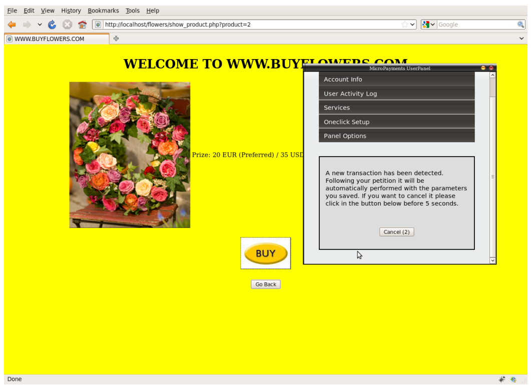
Transaction Performed Succesfully.

Oneclick Saved.

THANKS FOR BUYING AT WWW.BUYFLOWERS.COM

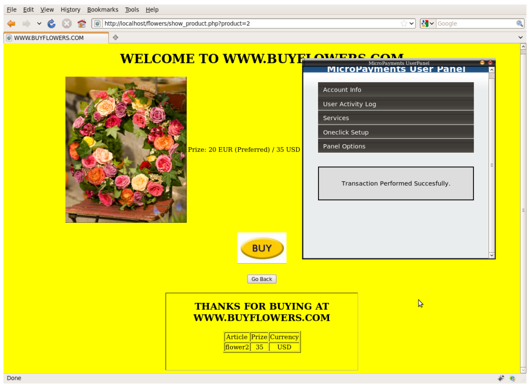
Article	Prize	Currency
flower2	20.00	EUR

(c) When he clicks on ”Save and Pay” he saves the Oneclick option and performs the transaction.



A new transaction has been detected. Following your petition it will be automatically performed with the parameters you saved. If you want to cancel it please click in the button below before 5 seconds.

(d) From that point, and for the validity selected in the step (b), any other transaction performed will show a countdown of 5 seconds in the User Panel



WELCOME TO WWW.BUYFLOWERS.COM

Prize: 20 EUR (Preferred) / 35 USD

Transaction Performed Succesfully.

THANKS FOR BUYING AT WWW.BUYFLOWERS.COM

Article	Prize	Currency
flower2	35	USD

Figure 17: (e) If the user does not cancel the transaction in those 5 seconds, it is performed automatically

10.4 MPP Transaction Server

As the MPP payment system relies on virtual accounts for performing transactions between users and merchants, the transaction server only has to:

1. Add or subtract the amount of the transaction to the existent balance of the user and/or merchant in the MPP database;
2. Save all the data of that transaction in the MPP database for future logging, and
3. Update the user data in the User Panel.

Points 1 and 2 are handled in the MPP server by the file `transaction.php`. It receives the parameters needed (`psid` associated, if it is a cancellation or not, the type of transaction, the amount, the user and merchant id's and accounts and a flag that indicates if the user wants to create an Express Oneclick with it), instances a `Transaction` class and uses its perform method to do all the database procedures.

The `Transaction` class is prepared to perform any of the transactions the MPP system is able to handle, and respond with a single string indicating the final state of the transaction. Depending on the parameters passed to its constructor the type of transaction that it can perform varies. For example, for performing a deposit in a user account the constructor needs the user id, the user account id or the currency of the deposit, and the amount, while for performing a purchase it needs all the parameters but the currency. If a parameter is missing, typically the currency of the transaction, the constructor will try to get it from the parameters it has. If the constructor lacks a parameter and it is not able to automatically guess it, it will report an error as the response of the transaction and save it in the MPP database.

However, the `Transaction` class is obviously an internal tool. It can be instanced only in the MPP server. Actually only the `transaction.php` file access it, so for security reasons it asks for authentication credentials only the MPP server knows.

Point 3 is handled by the User Panel once it has retrieved the response from the transaction server. It basically uses two JavaScript functions called `confirmToUser` and `confirmToMerchant`. The first one changes the visualization of the User Panel to show a confirm message with the result of the transaction, while the second one sends a XDA message to the merchant with the result of the transaction, so the merchant can capture it with its callback function and act accordingly.

11 PayPal Integration

As explained in section 4.1. PayPal offers the MPP the external layer to transform virtual money into real one. To make this process as transparent to the MPP users (merchants and their users) as possible, the MPP system uses the public PayPal API, which is able to trigger transfers between PayPal accounts and initiate checkout processes (which in the MPP case will be transformed into MPP Virtual Account transfers).

In this section I will review all the PayPal tools needed for its implementation in the MPP system: the services they provide, the testing environment, its API and callback system and the currency handling. Then I will explain the integration process, and I will comment briefly their fee policies in production environments.

11.1 PayPal Services

PayPal offers a variety of services for their customers to perform online transactions. Those services are usually related to a checkout process triggered by a preset button, although they provide businesses with more advanced functionalities, bundled in different packages.

The *Website Payments Standard* package provides the basic functionalities, which include the creation of personalized buttons and the use of the basic API functions to perform transfers between PayPal accounts.

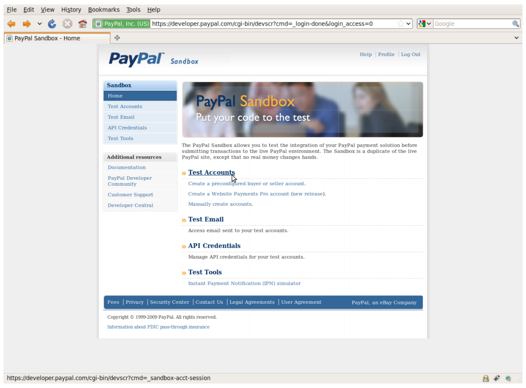
The *Website Payments Pro* package provides advanced functionalities to businesses with a PCI compliance certificate, meaning they already had implemented other payment options. The Direct Payment allows the use of the API to perform payments using a credit card introduced in the merchant website, so the user does not have to be redirected to the PayPal website. The Express Checkout provides a fast checkout service for users with a PayPal account, independently of where they are performing the purchase transaction (the merchant or the PayPal website). It requires to pay a monthly fee of around 30 USD.

Finally, PayPal offers a gateway service called *Payflow*, which allows the merchant to communicate directly to the bank of its user. It uses a completely separated API and provides advanced security options such as fraud protection and recurring billings. Right now the gateway is limited to United States, New Zealand, Australia, Canada and Singapore, and its basic setup has an initial fee of 250 USD with a monthly fee of around 50 USD.

As it can be observed, the PCI constraints for the Website Payments Pro and the country for the Payflow, together with the fact that they are more expensive solutions, limited the choice of the services to apply in the MPP system to the Website Payments Standard.

11.2 The SANDBOX Testing Environment

PayPal offers its users a testing environment called Sandbox. It is basically a mock-up of the real PayPal for business to test their API implementations. For using the environment you must signup in its webpage without using a real PayPal-assigned e-mail account. In sandbox, a user can create any number of fake accounts (basic, premium or business), which will be associated to him, and recreate its payment service, for example creating buttons or allowing the fake premium account to use the API as it was a real account. It provides the credentials for every account, and even a control panel from which the tester can trigger the login of the fake users. It also allows to specify any balance for any fake accounts, as well as any number of currencies.


File Edit View History Bookmarks Tools Help
PayPal, Inc. (US) https://developer.paypal.com/cgi-bin/devscr?cmd=_login-done&login_access=0
Google
PayPal Sandbox - Home
PayPal Sandbox
Help | Profile | Log Out
Sandbox
Home
Test Accounts
Test Email
API Credentials
Test Tools
Additional resources
Documentation
PayPal Developer Community
Customer Support
Developer Central
PayPal Sandbox
Put your code to the test
Fees | Privacy | Security Center | Contact Us | Legal Agreements | User Agreement
PayPal, an eBay Company
https://developer.paypal.com/cgi-bin/devscr?cmd=_sandbox-acct-session

The PayPal Sandbox allows you to test the integration of your PayPal payment solution before submitting transactions to the live PayPal environment. The Sandbox is a duplicate of the live PayPal site, except that no real money changes hands.

» **Test Accounts**

Create a preconfigured buyer or seller account.

Create a Website Payments Pro account (new release).

Manually create accounts.

» **Test Email**

Access email sent to your test accounts.

» **API Credentials**

Manage API credentials for your test accounts.

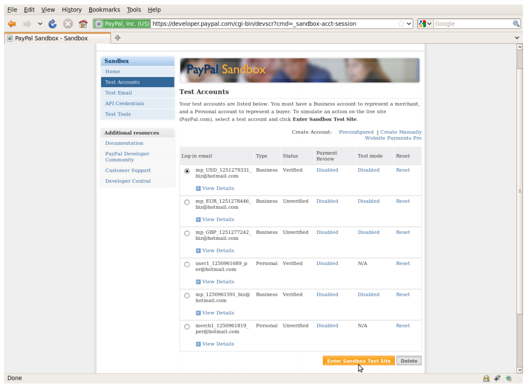
» **Test Tools**

Instant Payment Notification (IPN) simulator

Copyright © 1999-2009 PayPal. All rights reserved.

Information about FDIC pass-through insurance

Figure 18: Sandbox control panel



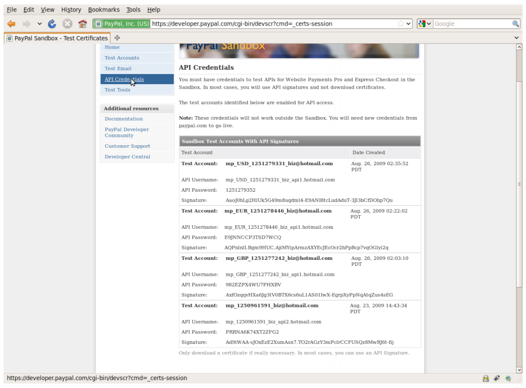
Test Accounts

Your test accounts are listed below. You must have a Business account to represent a merchant, and a Personal account to represent a buyer. To simulate an action on the live site (PayPal.com), select a test account and click **Enter Sandbox Test Site**.

Create Account: Preconfigured | Create Manually Website Payments Pro

	Log-in email	Type	Status	Payment Review	Test mode	Reset
●	mp_USD_1251279331_biz@hotmail.com View Details	Business	Verified	Disabled	Disabled	Reset
○	mp_EUR_1251278446_biz@hotmail.com View Details	Business	Unverified	Disabled	Disabled	Reset
○	mp_GBP_1251277242_biz@hotmail.com View Details	Business	Unverified	Disabled	Disabled	Reset
○	user1_1250961689_per@hotmail.com View Details	Personal	Verified	Disabled	N/A	Reset
○	mp_1250961591_biz@hotmail.com View Details	Business	Verified	Disabled	Disabled	Reset
○	merch1_1250961819_per@hotmail.com View Details	Personal	Unverified	Disabled	N/A	Reset

(a) PayPal User mockup


File Edit View History Bookmarks Tools Help
PayPal, Inc. (US) https://developer.paypal.com/cgi-bin/devscr?cmd=_certs-session
Google
PayPal Sandbox - Test Certificates
PayPal Sandbox
Home
Test Accounts
Test Email
API Credentials
Test Tools
Additional resources
Documentation
PayPal Developer Community
Customer Support
Developer Central
https://developer.paypal.com/cgi-bin/devscr?cmd=_certs-session

API Credentials

You must have credentials to test APIs for Website Payments Pro and Express Checkout in the Sandbox. In most cases, you will use API signatures and not download certificates.

The test accounts identified below are enabled for API access.

Note: These credentials will not work outside the Sandbox. You will need new credentials from paypal.com to go live.

Sandbox Test Accounts With API Signatures

Test Account		Date Created
Test Account:	**mp_USD_1251279331_biz@hotmail.com**	Aug. 26, 2009 02:35:52 PDT
API Username:	mp_USD_1251279331_biz_api1.hotmail.com	
API Password:	1251279352	
Signature:	AaoJ0hLg2H1Uk5G49m8uqdml4-E9ANI8tcLudAduT-3Jl3bCfDOhp7Qu	
Test Account:	**mp_EUR_1251278446_biz@hotmail.com**	Aug. 26, 2009 02:22:02 PDT
API Username:	mp_EUR_1251278446_biz_api1.hotmail.com	
API Password:	E9JNNCCP3TSD7WCQ	
Signature:	AQPnlnlI.Bgm9HUC.AjiMYipArmzAXYEcJEcOcr2hPpBcp7vqOGlyi2q	
Test Account:	**mp_GBP_1251277242_biz@hotmail.com**	Aug. 26, 2009 02:03:10 PDT
API Username:	mp_GBP_1251277242_biz_api1.hotmail.com	
API Password:	982EZPX4WU7PHXBV	
Signature:	AxfOzqqvHXa6Jg3tV0BTX6cs6uL1AS01IwX-EgrpXyPpNqAtqZus4sEG	
Test Account:	**mp_1250961591_biz@hotmail.com**	Aug. 23, 2009 14:43:34 PDT
API Username:	mp_1250961591_biz_api2.hotmail.com	
API Password:	PRRNA6K74XT2ZFG2	
Signature:	Ad9iWAA-sJOxEzE2XumAsn7.TO2rAGzY3mPclrCCFUSQz8MwRJ6t-fij	

Only download a certificate if really necessary. In most cases, you can use an API Signature.

(b) PayPal API credentials

Figure 19: Sandbox user control panel

Obviously this environment is perfect for the scope of the MicroPayments project, as it allows to recreate all possible situations without using real accounts with real money.

11.3 The PayPal API

For using the PayPal API a Premium or Business account has to be created. Any person can have a Premium account, but the Business account has to be bound to the business information (the Sandbox environment allows to create them by using fake data).

Once the account is activated, you have to ask for API credentials in the PayPal web. You have to choose between ”signature“ or ”certificate“ credentials. The first one is a set of username, password and encrypted signature parameters that have to be sent with the API call; the second one is using an SSL certificate to send the API calls, without needing the signature parameters. For easiness I choose the signature method, although in production environments the SSL certificate is recommended.

PayPal has two different API technologies: Name-Value Pairs (NVP) and SOAP. The simplest implementation is the NVP, based on HTML, scripting and cURL calls. The SOAP implementation has a much more complicated, Object Oriented programming overhead and it’s designed specifically for securing the Direct Payments and Express Checkout calls (which can only be used in the non-free Website Payments Pro package). Anyway, as the Express Checkout and Direct Payments functionalities are out of the scope of the MicroPayments project, the only API functions than can be used are:

`GetBalance`
Shows the balance of the account associated to the API credentials.

`GetTransctionDetail`
Shows all the data related to an already performed transaction.

`MassPay`
Transfers money between PayPal accounts.

`RefundTransaction`
Forces the payback of a previously performed transaction.

`TransactionSearch`
Uses multiple filters for searching already performed transactions.

11.4 Callback Handling

If you have a Premium or Business account PayPal lets you enable a functionality called "Auto Return", which basically returns the user to the URL you set in the PayPal configuration or using an API call. Then, to establish a real callback method PayPal supports different options:

Payment Data Transfer (PDT): provides merchants with the information on the transaction performed (identified by a token), so he can use it to confirm the user. It uses a token system and GET/POST variables to authorize the merchant to retrieve the information.

Instant Payment Notification (IPN): a messaging protocol that enables more advance functionalities, such as triggering other API actions. It requires a listener script running in the merchant server.

In the case of MicroPayments, PDT will provide all the information needed to confirm the user of the payment, so there is no necessity of implementing the more complicated IPN messaging. Note that this is only used when the user is redirected to the PayPal website. The rest is done by cURL responses.

11.5 Currency Handling

A PayPal user can have only one account, but he can specify the currency or currencies to use in it. If more than one, any transaction will automatically be charged to the account with the same currency as the transaction. If the user has not that currency the transaction will be stopped (pending) until the user manually decides what to do: exchange it or cancel it.

From the PayPal documentation: *"If your transaction involves a currency conversion, it will be completed at a foreign exchange rate PayPal obtains from a financial institution, which is adjusted regularly based on market conditions. This exchange rate includes a 2.5% processing fee above the exchange rate, and the processing fee is retained by PayPal. The specific exchange rate that applies to your multiple currency transaction will be displayed at the time of the transaction"*.

There is no way of exchanging currencies between accounts from the same user using the API, although it can be done manually using the PayPal web.

For all this reasons, the currency handling of the MPP system will not use the PayPal system, but a self implementation of the forex exchange rate conversions. Concretely, the MPP prototype uses the FoXRate XML-RPC API [citation], a free and simple forex table updated frequently:

RPC endpoint: http://foxrate.org/rpc/

Method name: foxrate.currencyConvert

Parameters:
- from currency (eg: USD) = string

- to currency (eg: GBP) = string
- amount to convert (eg:100.0) = float

Response: foxrate xml-rpc interface will response with a Struct of 3 members:

- `flerror` (Boolean): 1 if there is an error , 0 if the call is successful
- `amount` (double) : converted amount
- `message` (string) : date of latest exchange rate. contains error message if flerror is 1

Example of Request:

```
POST /rpc/ HTTP/1.0
Host: foxrate.org
User-Agent: xmlrpclib.py/1.0  1 (by www.pythonware.com)
Content-Type: text/xml
Content-Length: 270

<?xml version='1.0'?>
<methodCall>
        <methodName> foxrate.currencyConvert </methodName>
        <params>
                <param><value>  <string>USD</string></value></param>
                <param><value><string>GBP</string></value></param>
                        <param><value><double>100</double></value></param>
        </params>
</methodCall>
```

Example of Response:

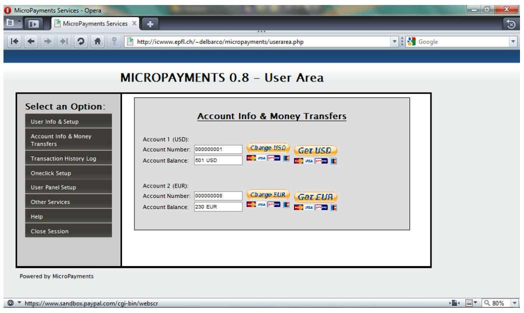
```
HTTP/1.1 200 OK
Date: Sun, 11 Feb 2007 14:47:14 GMT
Server: Apache/2.0.54
X-Powered-By: PHP/5.1.2
Connection: close
Content-Length: 396
Vary: Accept-Encoding
Content-Type: text/xml

<?xml version="1.0"?>
<methodResponse>
  <params>
  <param>
    <value><struct>
      <member>
        <name>flerror</name>
        <value><int>0</int></value>
      </member>
      <member>
        <name>amount</name>
        <value><double>50.36</double></value>
      </member>
      <member>
        <name>message</name>
        <value><string>"2/9/2007"</string></value>
      </member>
    </struct></value>
  </param>
  </params>
</methodResponse>
```

11.6 MPP Virtual Account Handling

The MPP system uses personalized buttons to deposit and withdraw money from the PayPal accounts to the MPP Virtual Accounts, making it really easy to use its users. Those buttons can be found in the option ”Account Info & Money Transfers” of the User Area, in the MicroPayments website.

The steps for depositing money in the MPP Virtual Accounts are shown in the following pictures:



http://icwww.epfl.ch/~delbarco/micropayments/userarea.php

MICROPAYMENTS 0.8 – User Area

Account Info & Money Transfers

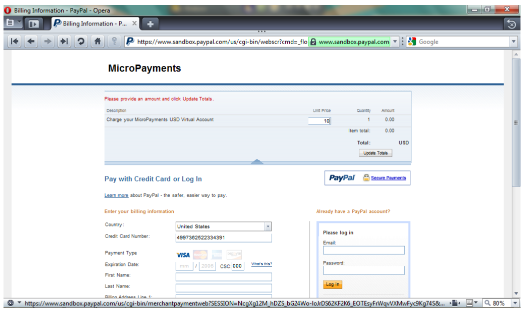
Account 1 (USD):
Account Number: 000000001
Account Balance: 501 USD

Account 2 (EUR):
Account Number: 000000008
Account Balance: 230 EUR

Powered by MicroPayments

https://www.sandbox.paypal.com/cgi-bin/webscr

(a) User clicks on the ”Charge USD” button in the User Area


Billing Information - PayPal - Opera
Billing Information - P...
Google
PayPal
Secure Payments
Update Totals
What's this?
Log In
80%

https://www.sandbox.paypal.com/us/cgi-bin/webscr?cmd=_flo www.sandbox.paypal.com

MicroPayments

Please provide an amount and click Update Totals.

Description	Unit Price	Quantity	Amount
Charge your MicroPayments USD Virtual Account	10	1	0.00
		Item total:	0.00
		Total:	USD

Pay with Credit Card or Log In

Learn more about PayPal - the safer, easier way to pay.

Enter your billing information

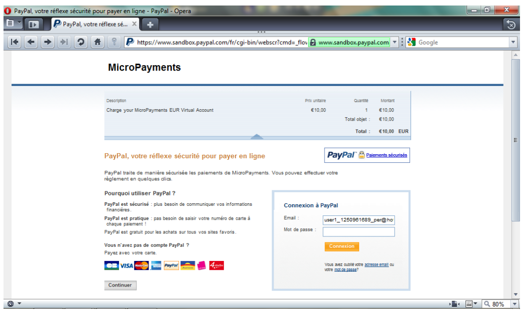
Country: United States
Credit Card Number: 4997362522334391
Payment Type
Expiration Date: mm / 2006 CSC: 000
First Name:
Last Name:
Billing Address Line 1:

Already have a PayPal account?

Please log in
Email:
Password:

https://www.sandbox.paypal.com/us/cgi-bin/merchantpaymentweb?SESSION=NcgXg12M_hDZS_bG24Wo-IoJrDS62KF2K6_EOTEsyFrWqvVXMwFyc9Kg74S&...

(b) User is redirected to PayPal website, where he writes the amount to deposit in the MPP Virtual Account


PayPal, votre réflexe sécurité pour payer en ligne - PayPal - Opera
PayPal, votre réflexe sé...
Google
PayPal
Paiements sécurisés
Connexion
Continuer
80%

https://www.sandbox.paypal.com/fr/cgi-bin/webscr?cmd=_flov www.sandbox.paypal.com

MicroPayments

Description	Prix unitaire	Quantité	Montant
Charge your MicroPayments EUR Virtual Account	€10,00	1	€10,00
		Total objet :	€10,00
		Total :	€10,00 EUR

PayPal, votre réflexe sécurité pour payer en ligne

PayPal traite de manière sécurisée les paiements de MicroPayments. Vous pouvez effectuer votre règlement en quelques clics.

Pourquoi utiliser PayPal ?

PayPal est sécurisé : plus besoin de communiquer vos informations financières.

PayPal est pratique : pas besoin de saisir votre numéro de carte à chaque paiement !

PayPal est gratuit pour les achats sur tous vos sites favoris.

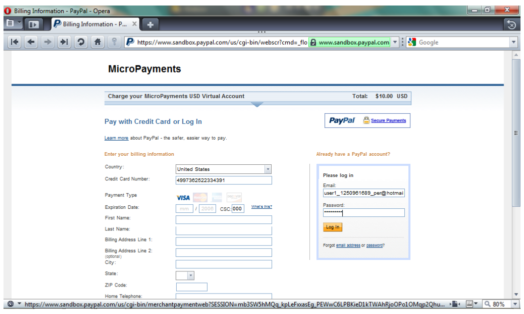
Vous n'avez pas de compte PayPal ?

Payez avec votre carte.

Connexion à PayPal
Email : user1_1250961689_per@ho
Mot de passe :

Vous avez oublié votre adresse email ou votre mot de passe?

(c) The same would happen by clicking on ”Charge EUR” button in the second user account. Note that now the PayPal currency is EUR.


Billing Information - PayPal - Opera
Billing Information - P...
Google
PayPal
Secure Payments
What's this?
Log In
80%

https://www.sandbox.paypal.com/us/cgi-bin/webscr?cmd=_flo www.sandbox.paypal.com

MicroPayments

Charge your MicroPayments USD Virtual Account Total: $10.00 USD

Pay with Credit Card or Log In

Learn more about PayPal - the safer, easier way to pay.

Enter your billing information

Country: United States
Credit Card Number: 4997362522334391
Payment Type
Expiration Date: mm / 2006 CSC: 000
First Name:
Last Name:
Billing Address Line 1:
Billing Address Line 2: (optional)
City:
State:
ZIP Code:
Home Telephone:

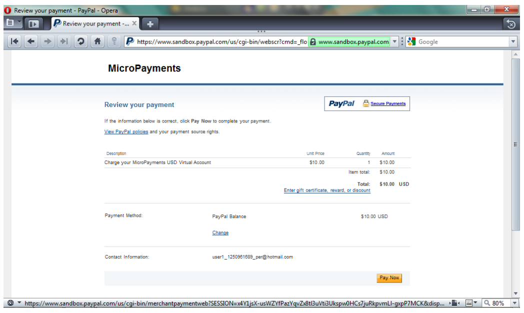
Already have a PayPal account?

Please log in
Email: user1_1250961689_per@hotmai
Password: ••••••••••

Forgot email address or password?

https://www.sandbox.paypal.com/us/cgi-bin/merchantpaymentweb?SESSION=mb3SW5hMQq_kpLeFxxasEg_PEWwC6LPBKieD1kTWAhRjoOPo1OMqp2Qhu...

(d) User clicks on ”Update Totals” and enters his PayPal account e-mail and password



https://www.sandbox.paypal.com/us/cgi-bin/webscr?cmd=_flo www.sandbox.paypal.com

MicroPayments

Review your payment

If the information below is correct, click Pay Now to complete your payment.

View PayPal policies and your payment source rights.

Description	Unit Price	Quantity	Amount
Charge your MicroPayments USD Virtual Account	$10.00	1	$10.00
		Item total:	$10.00
		Total:	$10.00 USD

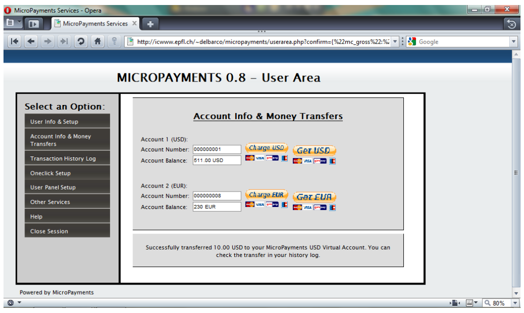
Enter gift certificate, reward, or discount

Payment Method: PayPal Balance $10.00 USD

Contact Information: user1_1250961689_per@hotmail.com

https://www.sandbox.paypal.com/us/cgi-bin/merchantpaymentweb?SESSION=x4Y1jsX-usWZYfPazYqvZx8tl3uVti3Ukspw0HCs7juRkpvmLI-gxpP7MCK&disp...

(e) Information on the payment is shown and user clicks on ”Pay Now”



http://icwww.epfl.ch/~delbarco/micropayments/userarea.php?confirm={%22mc_gross%22:%

MICROPAYMENTS 0.8 – User Area

Account Info & Money Transfers

Account 1 (USD):
Account Number: 000000001
Account Balance: 511.00 USD

Account 2 (EUR):
Account Number: 000000008
Account Balance: 230 EUR

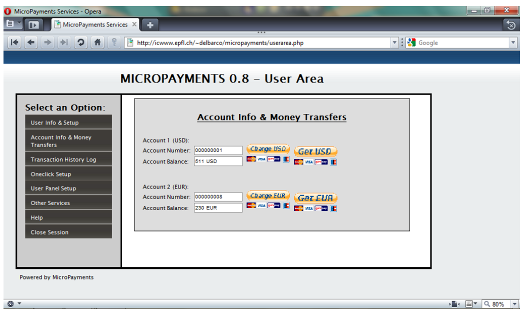
Successfully transferred 10.00 USD to your MicroPayments USD Virtual Account. You can check the transfer in your history log.

Powered by MicroPayments

(f) User is redirected back to the MPP website and is informed of the successful transaction. His balance is automatically updated.

Figure 20: Using PayPal for depositing money in a MPP Virtual Account

The steps for withdrawing money from the MPP Virtual Accounts to the PayPal account are shown in the following pictures:



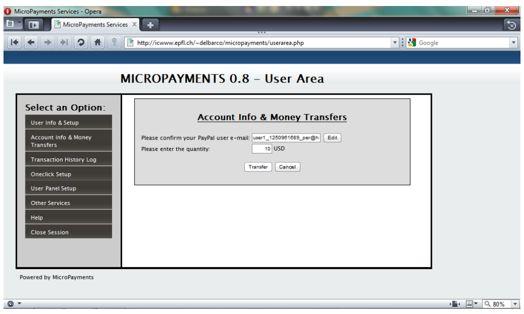
MICROPAYMENTS 0.8 – User Area

Account Info & Money Transfers

Account 1 (USD):
Account Number: 000000001
Account Balance: 511 USD

Account 2 (EUR):
Account Number: 000000008
Account Balance: 230 EUR

(a) User clicks on the "Get USD" button in the User Area

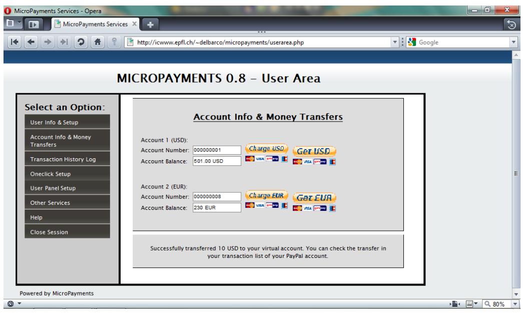


MICROPAYMENTS 0.8 – User Area

Account Info & Money Transfers

Please confirm your PayPal user e-mail: user1_1250961689_per@h
Please enter the quantity: 10 USD

(b) User confirms the PayPal account where to transfer the money and inserts the amount



MICROPAYMENTS 0.8 – User Area

Account Info & Money Transfers

Account 1 (USD):
Account Number: 000000001
Account Balance: 501.00 USD

Account 2 (EUR):
Account Number: 000000008
Account Balance: 230 EUR

Successfully transferred 10 USD to your virtual account. You can check the transfer in your transaction list of your PayPal account.

(c) The MPP system uses the PayPal API to transfer the money without redirecting the user. A confirmation message is shown and the balance is updated.

Figure 21: Using PayPal for withdrawing money from a MPP Virtual Account

11.7 Fees in Production Environment

PayPal charges its users with a fair amount of fees for using their services. For information about them, please refer to the following links:

- General Fee overview:
  https://www.paypal.com/ch/cgi-bin/webscr?cmd=p/gen/ua/policy_fees-outside
- Normal Transaction Fees:
  https://www.paypal.com/ch/cgi-bin/webscr?cmd=_display-receiving-fees-outside
- Cross-border Transaction Fees (for receiving transactions from buyers outside Switzerland):
  https://www.paypal.com/ch/cgi-bin/webscr?cmd=_display-xborder-fees-outside&countries=
- Withdrawal Fees (transfer from paypal to a real account):
  https://www.paypal.com/ch/cgi-bin/webscr?cmd=_display-withdrawal-fees-outside

Other characteristics of the PayPal fee system are:

- Receiving fees are assessed in the currency in which the funds were sent. Payments converted to the user's primary currency are converted at a frequently updated exchange rate with an applied fee of 2.5%.
- The Mass Pay feature (needed for sending money to the merchants and user's PayPal accounts) charges a fee of 2% of the payment amount with the following cap will be assessed on each payment:

Payment Currency	**Payment Cap**
USD	\$1.00 USD
CAD	\$1.25 CAD
EUR	0.85 EUR
GBP	0.65 GBP
JPY	120 JPY
AUD	\$1.25 AUD
CHF	1.30 CHF
NOK	6.75 NOK
SEK	9.00 SEK
DKK	6.00 DKK
PLN	3.00 PLN
HUF	210 HUF
CZK	24.00 CZK
SGD	\$1.60 SGD
HKD	\$7.00 HKD
NZD	\$1.50 NZD
ILS	4.00 ILS
MXN	\$11.00 MXN

It seems obvious to think that to be able to maximize the benefits of the MPP system in a real business environment, the amount of PayPal based transactions has to be reduced to a minimum, as in the worst case they charge a fee for performing the transaction, another if it is a cross-border, and even more if a currency exchange has to be done.

One of the scopes of future implementations should be, then, to make the necessary steps to become a PCI compliance business, so to be able to use or even implement a direct gateway system.

12 Database Management

The MPP server uses a MySQL database for saving the data. The database structure is the following:

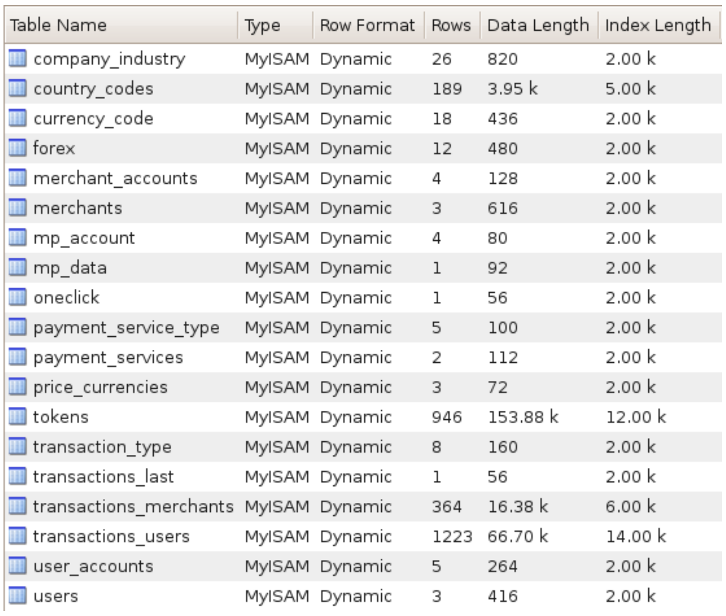


Table Name	Type	Row Format	Rows	Data Length	Index Length
company_industry	MyISAM	Dynamic	26	820	2.00 k
country_codes	MyISAM	Dynamic	189	3.95 k	5.00 k
currency_code	MyISAM	Dynamic	18	436	2.00 k
forex	MyISAM	Dynamic	12	480	2.00 k
merchant_accounts	MyISAM	Dynamic	4	128	2.00 k
merchants	MyISAM	Dynamic	3	616	2.00 k
mp_account	MyISAM	Dynamic	4	80	2.00 k
mp_data	MyISAM	Dynamic	1	92	2.00 k
oneclick	MyISAM	Dynamic	1	56	2.00 k
payment_service_type	MyISAM	Dynamic	5	100	2.00 k
payment_services	MyISAM	Dynamic	2	112	2.00 k
price_currencies	MyISAM	Dynamic	3	72	2.00 k
tokens	MyISAM	Dynamic	946	153.88 k	12.00 k
transaction_type	MyISAM	Dynamic	8	160	2.00 k
transactions_last	MyISAM	Dynamic	1	56	2.00 k
transactions_merchants	MyISAM	Dynamic	364	16.38 k	6.00 k
transactions_users	MyISAM	Dynamic	1223	66.70 k	14.00 k
user_accounts	MyISAM	Dynamic	5	264	2.00 k
users	MyISAM	Dynamic	3	416	2.00 k

Figure 22: MPP Database structure

`company_industry`
The industry of the merchant. Saved in the signup process.

`country_codes`
The standard three capital letter codes of the main countries of the world.

`currency_code`
The standard three capital letter codes of the main currencies of the world.

`forex`
A table with the exchange rates of all the currencies supported by the system, and the fee that the MPP system will apply to an exchange between them. The information is retrieved from the FoXRate public API. In production environments should be updated at least daily.

`merchant_accounts`
The data related to the virtual accounts of the merchants with all the subsequent data (balance, currency, etc.).

`merchants`
A list of the merchants that use the MPP system. It has all the data introduced by them during the signup process.

`mp_account`
The data related to the MPP virtual account with all the subsequent data (balance, currency, etc.).

`mp_data`
General data about the MPP (version, transaction server URL, base URL, etc.)

`oneclick`
All the information related to the Oneclick configurations of the users.

`payment_service_type`
Types of payment services that the MPP system handles.

`payment_services`
All the data related to the payment services created by the merchants.

`price_currencies`
List of default price/currency rates that the merchant can specify in the signup process or in his control panel. If no price/currency information parameter is passed when triggering a transaction, the system will retrieve and uses this one.

`tokens`
Placeholder for the tokens used by the OAuth protocol.

`transaction_type`
All the transaction types the MPP system handles.

`transactions_last`
It contains the last transaction performed by all the users of the system, in case they want to create an Express Oneclick configuration.

`transactions_merchants`
List of transactions where merchants are involved. This includes purchases from users as well as PayPal transactions.

`transactions_users`
List of transactions where users are involved. This includes purchases as well as PayPal transactions.

`user_accounts`
The data related to the virtual accounts of the users with all the subsequent data (balance, currency, etc.).

`users`
A list of the users that use the MPP system. It has all the data introduced by them during the signup process.

13 Merchant Integration

Now that the MicroPayments Platform has been explained in detail, the only step left is to be able to integrate it in merchant websites. As the reader can see, one of the main concerns of this project has been minimizing and simplifying the steps involving the integration and using the MPP system. Thanks to this effort, the MPP integration is very simple no matter how the merchant website is constructed and programmed. Moreover, it is compatible with all the main web browsers. The MPP API can be loaded from any site in the merchant server, even a plain HTML file, as it only needs JavaScript to work. The technical knowledge required to integrate it, then, is very low. However, please note that the whole MPP system does require PHP to work, as the current MPP library is only implemented in PHP.

In subsection 13.1 an integration guide is explained, while subsection 13.2 shows the test examples that have been programmed along the MPP to test all its features.

13.1 Integration Steps

The MPP payment system needs only 2 files and a few lines of code to be fully functional, no matter if the system is using OAuth or the XDA messaging options.

The steps for a basic integration (the one in the e-shop model explained in section 13.2.1) are:

1. If not done yet, create a merchant account by signing up in the MicroPayments webpage. During the process, the merchant will be asked to create a Payment Service associated to him. After finishing the signup process, the system will provide the merchant his consumer_key, consumer_secret and all the account information. It also will give him a link to download the required files for the MPP integration, which have to be downloaded and used as described in the next steps.

2. Copy the file `micropayments.php` and `Micropayments.php` in the server where the website is. The `micropayments.php` file is the central monitor of the XDA messaging and the OAuth processor and it requires the file `Micropayments.php`, as it includes all its classes and functions. The merchant has to ensure then to include those files in the php code of the web page.

3. If the merchant wants to use the OAuth system he has to insert the `CONSUMER_KEY` and `CONSUMER_SECRET`, that were provided to him when signing in the MPP website, in the file `Micropayments.php`. For doing that, he must use the `define("CONSUMER_KEY","")` and `define("CONSUMER_SECRET","")` PHP statements. For security reasons a hard-coding of those variables is not recommended. For example:

```
$link = mysql_connect('host','username','password');
$sql = 'select consumer_key,consumer_secret from private_data';
$result = mysql_query($sql,$link);
$consumer_data = mysql_fetch_assoc($result);

// Consumer key and secret from MicroPayments (NOT MODIFY THIS VALUES)
define("CONSUMER_KEY", $consumer_data['consumer_key']);
define ("CONSUMER_SECRET",$consumer_data['consumer_secret']);
```

4. Insert the following lines in the `<head>` section of every merchant web page that needs to use the MPP functionality.

```
<script type="text/javascript"
    src="http://www.micropayments.com/js/micropayments.js"></script>
<link href="http://www.micropayments.com/css/micropayments.css"
    rel="stylesheet" type="text/css" />
```

5. To initialize the Payment Service API, insert the following line in the `<head>` tag, after the previous ones:

```
<script type="text/javascript">

var ps = new PaymentService(true,"test2",callback,
'http://www.merchant.com/whateverurl.php');

</script>
```

If the merchant has already other javascript code, the previous lines can be inserted anywhere inside the `<script>` tag, but obviously before the using of any API function.

The first parameter in the PaymentService initialization indicates if the User Panel is activated or not. The second is the unique id of the payment service to load. This id is given to the merchant when he creates that Payment Service in the MicroPayments website. The third parameter is the name of the callback function the merchant wants to use (must be JavaScript and loaded before the initialization of the PaymentService API). Finally the last parameter is the URL to go back in case the user needs to authenticate to use the MPP User Panel.

6. If the OAuth mode is not activated the MPP button must be loaded by inserting the following code:

```
//Load the MP button when the DOM is loaded
window.onload = function(){
  ps.showMicropaymentsButton('mpButtonLoad');
}
```

Where `mpButtonLoad` is the name of the HTML container of the button. If it is activated, this step is not required.

And that's it. From this point and with only 6 basic steps at much (4 if OAuth is not used, which is the normal scenario) the MPP system is integrated and completely functional. Now the merchant can use the MPP API as explained in section 9, and the user will be able to load the User Panel and perform transactions in the MPP server.

13.2 Test Examples

As said before in this report, two different e-commerce models where created to test the implementation system and the different payment authorization modes. The idea was to recreate in a basic way real e-commerce sites and integrate the MPP platform in them. The following is a description of those test sites.

13.2.1 e-Shop Example

The first example of integration is a basic and typical e-shop in the flower market. The scope is to test the platform in situations where small amounts of more expensive transactions are performed, as a substitutive of the normal checkout process.

The site provides a small catalog of flowers to sell, with their prices, and a "Buy" button. The web interface has been fully explained as the integration example in the Transaction Processing section (10). The files involved in it are:

index.php: The home of the website. It shows the catalog of flowers to purchase as images. Clicking on any of them will load the respective data in the `show_product.php` web.

show_product.php: It displays the price of the specific flower selected in the `index.php`, and provides a "Buy" button to start the purchase process. This website is where the MPP system is loaded, so it is the subject of the merchant integration explained in section 13.1.

confirm.php This webpage, embedded as an iFrame in the `show_product.php`, shows the results of the transactions after processing the callback messages sent by the MPP platform (independently of the payment mode used).

css/flowers.css A basic Cascade Style Sheet file for the style of the HTML elements inside the www.buyflowers.com website.

js/confirm.js The JavaScript file containing the callback function for processing the MPP callback messages.

The site also contains the files required by MPP as explained in section 13.1: `micropayments.php` and `classes/Micropayments.php`.

13.2.2 Web Game Example

This second example of integration goes directly into the micropayment problematic by presenting a web-based trivia game. For testing the integration capabilities of the MPP we used a predefined, open-source package called "PHP Dynamic Trivia", by Robin van de Vusse (r.vandevusse@rogers.com). The base game includes a MySQL database with the system data and trivia questions, a login area and the main game area, where the logged user can choose between various topics to start the game. Once he chooses a topic, the user has to answer a randomized series of questions, with the only hint of knowing how many words and letters the answer has.

To integrate the MPP system and test all its features the code of the game has been changed to:

- Allow an anonymous play (no login required) in addition to the account-based one;
- Charge the user in real-time for every formulated question;
- Reward the user in case the answer is correct;
- Show a panel (iFrame) with statistics about number of correct and wrong answers and the maximum number of correct answers in a row;
- Forced OAuth or User Panel Mode (specialy the last one), to show the real capabilities of the MPP system when applied to fast payments;
- For testing purposes, the system can show the correct answer in the "hint" area.

As the rate of the transactions is much higher (around 5 per minute), a Oneclick configuration (see 4.5) is suggested to the user once he performs the first transaction (or loads the User Panel). If activated, the game becomes much more playable and with fully automated payments.

The most relevant files involved in the website are:

index.php: The home of the website. It shows the login area and a new "Anonymous Play" button, as well as a register link in case the user wants to create an account to save his data. The login form calls the file `login.php` to process the database query. It is also the first webpage with MPP integration, so the MPP button to load the system is here.

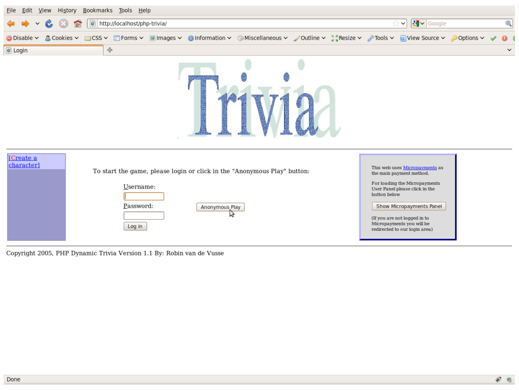
main.php: This webpage lets the user select the topic of the trivia to play. It also loads the MPP User Panel. If the user does not have a Oneclick configuration activated for that site, the User Panel suggests him to do so. This page also contains a score panel with real-time data (not present in the original code).

question.php: This webpage shows the question for the selected topic. It also loads the User Panel, and updates the amount if the question is correct.

questionproc.php: This file processes the answers introduced by the user and checks if they are correct or not

As in the e-shop example, the files required for the MPP integration are also present (`micropayments.php` and `classes/Micropayments.php`).

The following are a set of screenshots showing the web game and the MPP implementation:



http://localhost/php-trivia/

[Create a character]

To start the game, please login or click in the "Anonymous Play" button:

Username:

Password:

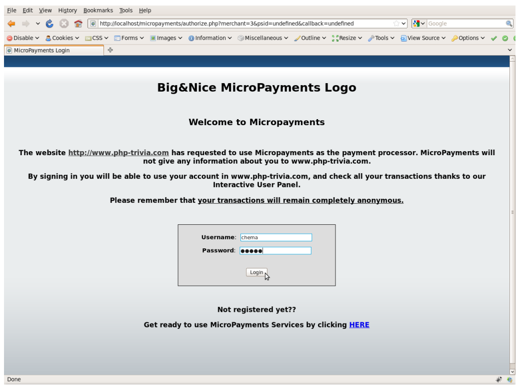
This web uses Micropayments as the main payment method.

For loading the Micropayments User Panel please click in the button below

(If you are not logged in to Micropayments you will be redirected to our login area)

Copyright 2005, PHP Dynamic Trivia Version 1.1 By: Robin van de Vusse

(a) Main page. The MPP button is displayed.



http://localhost/micropayments/authorize.php?merchant=3&psid=undefined&callback=undefined

Big&Nice MicroPayments Logo

Welcome to Micropayments

The website http://www.php-trivia.com has requested to use Micropayments as the payment processor. MicroPayments will not give any information about you to www.php-trivia.com.

By signing in you will be able to use your account in www.php-trivia.com, and check all your transactions thanks to our Interactive User Panel.

Please remember that your transactions will remain completely anonymous.

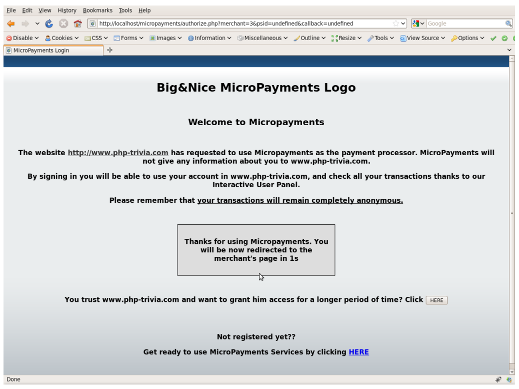
Username: chema

Password: ●●●●●

Not registered yet??

Get ready to use MicroPayments Services by clicking HERE

(b) When user clicks in "Anonymous Play" and is not logged in the MPP website, he is redirected to it to do so



http://localhost/micropayments/authorize.php?merchant=3&psid=undefined&callback=undefined

Big&Nice MicroPayments Logo

Welcome to Micropayments

The website http://www.php-trivia.com has requested to use Micropayments as the payment processor. MicroPayments will not give any information about you to www.php-trivia.com.

By signing in you will be able to use your account in www.php-trivia.com, and check all your transactions thanks to our Interactive User Panel.

Please remember that your transactions will remain completely anonymous.

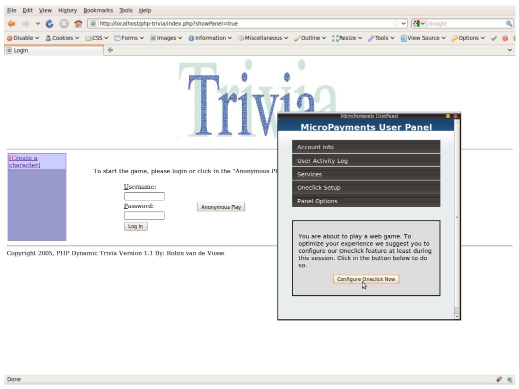
Thanks for using Micropayments. You will be now redirected to the merchant's page in 1s

You trust www.php-trivia.com and want to grant him access for a longer period of time? Click

Not registered yet??

Get ready to use MicroPayments Services by clicking HERE

(c) User logs in and is redirected back to the trivia website



http://localhost/php-trivia/index.php?showPanel=true

[Create a character]

To start the game, please login or click in the "Anonymous Pl

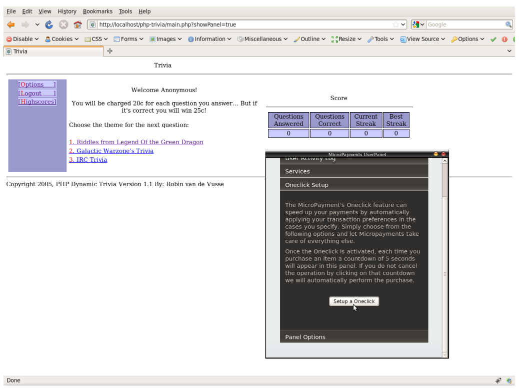
Username:

Password:

Copyright 2005, PHP Dynamic Trivia Version 1.1 By: Robin van de Vusse

You are about to play a web game. To optimize your experience we suggest you to configure our Oneclick feature at least during this session. Click in the button below to do so.

(d) The User Panel detects the user does not have a Oneclick configuration, and suggests to do so



http://localhost/php-trivia/main.php?showPanel=true

Trivia

Welcome Anonymous!

You will be charged 20c for each question you answer... But if it's correct you will win 25c!

Choose the theme for the next question:

1. Riddles from Legend Of the Green Dragon
2. Galactic Warzone's Trivia
3. IRC Trivia

Score

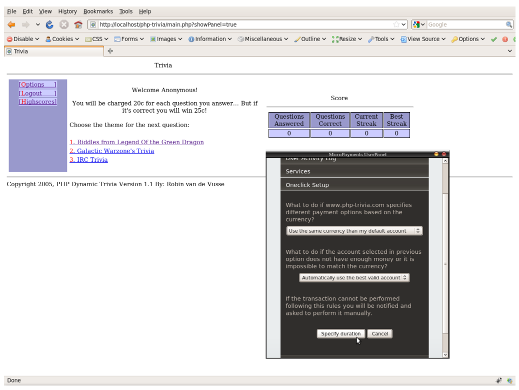
Questions Answered	Questions Correct	Current Streak	Best Streak
0	0	0	0

Copyright 2005, PHP Dynamic Trivia Version 1.1 By: Robin van de Vusse

The MicroPayment's Oneclick feature can speed up your payments by automatically applying your transaction preferences in the cases you specify. Simply choose from the following options and let Micropayments take care of everything else.

Once the Oneclick is activated, each time you purchase an item a countdown of 5 seconds will appear in this panel. If you do not cancel the operation by clicking on that countdown we will automatically perform the purchase.

(e) The user starts the Oneclick setup



http://localhost/php-trivia/main.php?showPanel=true

Trivia

Welcome Anonymous!

You will be charged 20c for each question you answer... But if it's correct you will win 25c!

Choose the theme for the next question:

1. Riddles from Legend Of the Green Dragon
2. Galactic Warzone's Trivia
3. IRC Trivia

Score

Questions Answered	Questions Correct	Current Streak	Best Streak
0	0	0	0

Copyright 2005, PHP Dynamic Trivia Version 1.1 By: Robin van de Vusse

What to do if www.php-trivia.com specifies different payment options based on the currency?

Use the same currency than my default account

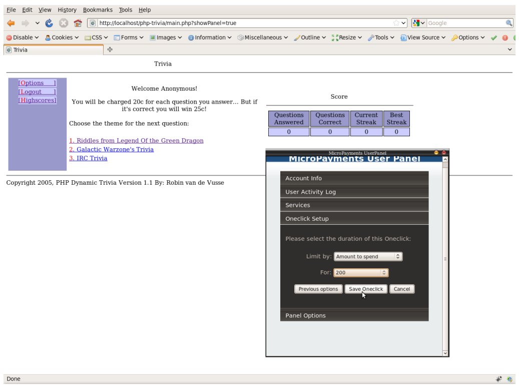
What to do if the account selected in previous option does not have enough money or it is impossible to match the currency?

Automatically use the best valid account

If the transaction cannot be performed following this rules you will be notified and asked to perform it manually.

(f) The user selects the currency behavior

Figure 23: PHP-Trivia Web Game



http://localhost/php-trivia/main.php?showPanel=true

Trivia

[Options] [Logout] [Highscores]

Welcome Anonymous!

You will be charged 20c for each question you answer... But if it's correct you will win 25c!

Choose the theme for the next question:

1. Riddles from Legend Of the Green Dragon
2. Galactic Warzone's Trivia
3. IRC Trivia

Score

Questions Answered	Questions Correct	Current Streak	Best Streak
0	0	0	0

Copyright 2005, PHP Dynamic Trivia Version 1.1 By: Robin van de Vusse

MicroPayments User Panel

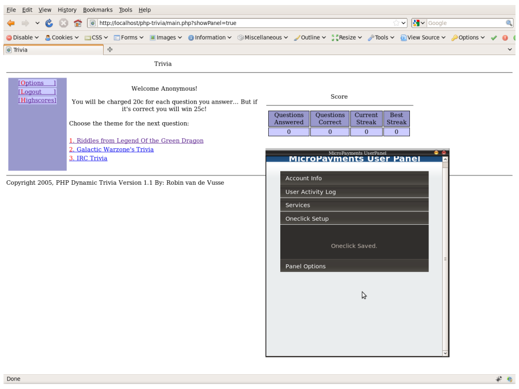
Please select the duration of this Oneclick:

Limit by: Amount to spend

For: 200

File Edit View History Bookmarks Tools Help
Disable Cookies CSS Forms Images Information Miscellaneous Outline Resize Tools View Source Options
MicroPayments UserPanel
Account Info
User Activity Log
Services
Oneclick Setup
Previous options
Save Oneclick
Cancel
Panel Options
Done

(a) The user specifies that wants to automate all transactions until 200$ are spent



http://localhost/php-trivia/main.php?showPanel=true

Trivia

Welcome Anonymous!

You will be charged 20c for each question you answer... But if it's correct you will win 25c!

Choose the theme for the next question:

1. Riddles from Legend Of the Green Dragon
2. Galactic Warzone's Trivia
3. IRC Trivia

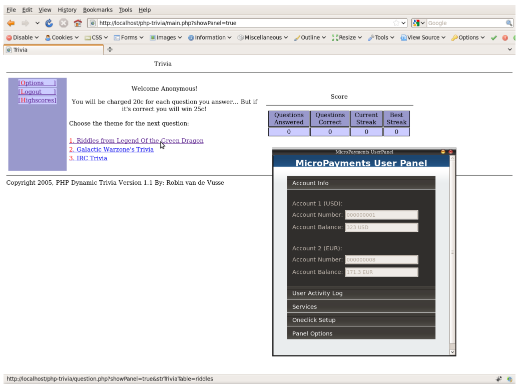
Questions Answered	Questions Correct	Current Streak	Best Streak
0	0	0	0

Copyright 2005, PHP Dynamic Trivia Version 1.1 By: Robin van de Vusse

MicroPayments User Panel

Oneclick Saved.

(b) The Oneclick configuration is saved



http://localhost/php-trivia/main.php?showPanel=true

Trivia

Welcome Anonymous!

You will be charged 20c for each question you answer... But if it's correct you will win 25c!

Choose the theme for the next question:

1. Riddles from Legend Of the Green Dragon
2. Galactic Warzone's Trivia
3. IRC Trivia

Questions Answered	Questions Correct	Current Streak	Best Streak
0	0	0	0

Copyright 2005, PHP Dynamic Trivia Version 1.1 By: Robin van de Vusse

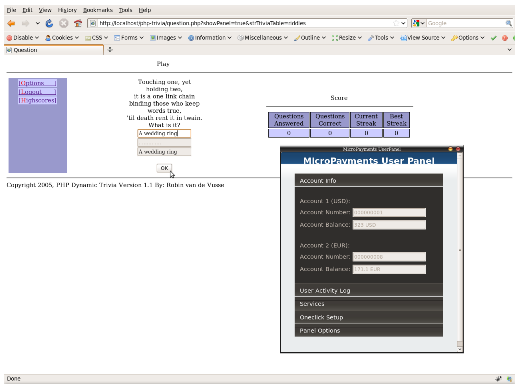
MicroPayments User Panel

Account 1 (USD):
Account Number: 000000001
Account Balance: 323 USD

Account 2 (EUR):
Account Number: 000000008
Account Balance: 171.3 EUR

http://localhost/php-trivia/question.php?showPanel=true&strTriviaTable=riddles

(c) The user clicks on the first topic for playing the quiz



http://localhost/php-trivia/question.php?showPanel=true&strTriviaTable=riddles

Play

Touching one, yet
holding two,
it is a one link chain
binding those who keep
words true,
'til death rent it in twain.
What is it?

A wedding ring

A wedding ring

Questions Answered	Questions Correct	Current Streak	Best Streak
0	0	0	0

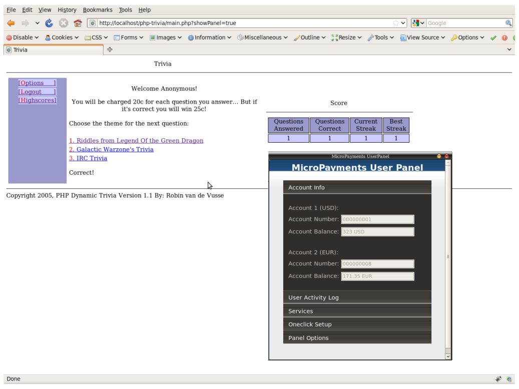
Copyright 2005, PHP Dynamic Trivia Version 1.1 By: Robin van de Vusse

MicroPayments User Panel

Account 1 (USD):
Account Number: 000000001
Account Balance: 323 USD

Account 2 (EUR):
Account Number: 000000008
Account Balance: 171.1 EUR

(d) The user is charged 0.20, and answers the question



http://localhost/php-trivia/main.php?showPanel=true

Trivia

Welcome Anonymous!

You will be charged 20c for each question you answer... But if it's correct you will win 25c!

Choose the theme for the next question:

1. Riddles from Legend Of the Green Dragon
2. Galactic Warzone's Trivia
3. IRC Trivia

Correct!

Questions Answered	Questions Correct	Current Streak	Best Streak
1	1	1	1

Copyright 2005, PHP Dynamic Trivia Version 1.1 By: Robin van de Vusse

MicroPayments User Panel

Account 1 (USD):
Account Number: 000000001
Account Balance: 323 USD

Account 2 (EUR):
Account Number: 000000008
Account Balance: 171.35 EUR

(e) The answer is correct, so the user is paid 0.25 and is asked to select another question

Figure 24: PHP-Trivia Web Game (Part 2)

Part III

Conclusion

The MicroPayments Platform has become a prototype with the potential to become a successful payment system in production environments: it provides plenty of benefits -including some never seen before in a payment platform- with almost no integration effort from merchants, and no effort is always a great marketing tool for itself. It is, at the end, a great conjunction of ideas from the people involved in it, also web users, a response to the necessities we have found while trying to purchase over Internet.

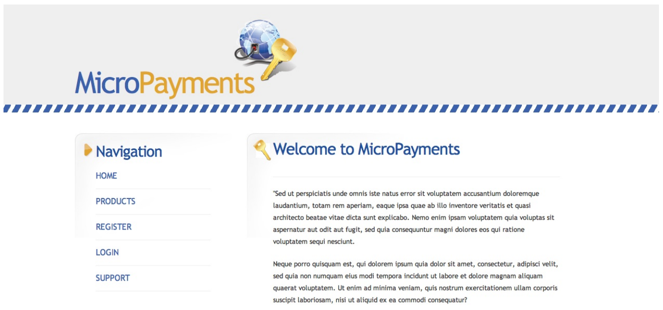
The Micropayments Platform, though fully functional, is still a prototype. A lot of new functionalities and enhancements can be done with more time. Its possibilities to be integrated in different payment services around Internet are unlimited.

Some of the enhancements that, in my opinion, should be done to transform the MPP project into a real money maker are:

- Implement the EmbededOneclick payment method;
- Implement its own payment gateway system, or use and existent one (meaning getting the PCI compliance);
- Add SSL encryption to the MPP server;
- Add fraud detection techniques;
- Implement the merchant MPP library in programming languages other than PHP;
- Add a Web browser plugin as an extended MPP User Panel;
- Add a financial strategy to manage currency conversions off-line;
- Integrate it in ”pay-per-x“ services: pay-per-time, pay-per-token, etc.;
- Add refund and chargeback features;
- Add User Panel interface personalization;
- Integrate it in Social Networks;
- Optimize the database usage;
- Optimize the code for faster execution and less requirements;

The XDA messaging system is a powerful tool to bypass the *same origin policy*. It can be used (and it is starting to be) in plenty of applications. However, special attention has to be made to the new proposition of the W3C called Cross-Origin Resource Sharing[7]. This will allow, in the close future, a better and easier implementation of the Cross Domain messaging.

Finally, at the moment this report was written the MPP website was being redesigned to have a more professional look & feel, able to meet real user expectations. The next capture shows just as a taste of it.



"Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo. Nemo enim ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur magni dolores eos qui ratione voluptatem sequi nesciunt.

Neque porro quisquam est, qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit, sed quia non numquam eius modi tempora incidunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim ad minima veniam, quis nostrum exercitationem ullam corporis suscipit laboriosam, nisi ut aliquid ex ea commodi consequatur?

Figure 25: New design for the MicroPayments Platform Home Website

About the Project

This project has been about challenges and learning. From the very beginning —coming from the communications engineering branch and thus with no programming expertise— I knew that designing and implementing a whole payment platform by myself would be a really hard work. The path the MPP took presented technological challenges (Ajax, OAuth and finally the XDA messaging) that forced me to learn a lot of web related technologies. Now, at the end of the project, after giving birth to the prototype of the Micropayments Platform, I can recall that all I though I knew when I started were mere basic concepts, the point of the iceberg in a world based on user satisfaction, business and commerce, and thus I centered all my attention to satisfy users and merchants, to make this project not a theoretical concept, but a real approximation to a professional product that can be used in real websites.

In this project I learned to be wrong. I learned that when coding there is always, ALWAYS a bug, the real problem is when you will find it. I learned all the steps involved in the creation of a fully functional software, its design, the look for the technology needed for implementing it and the long hours of coding. Despite the hard work, and in fact thanks to it, I really enjoyed this project.

I must thank David Portabella for giving me the opportunity to do this project at EPFL; for guiding me through this learning process and supporting me in all the technological doubts; for giving so many ideas about how to solve the main technological problems of the project, such as OAuth and the XDA technique, and spending hours with me trying to optimize them for the project... And most importantly, for asking me the questions I needed to be asked to understand by myself all the implications of what I was doing.

I must thank Pierre Tholence for his unlimited resource in coding techniques. Thanks to his prior investigation I could implement the XDA messaging technique, which opened all the doors I needed for the functionalities I wanted to give to the project. He also aided me in all the doubts I had during the implementation phase.

I must thank my family (the farther and the closer) for their infinite support, not only in this project but in the whole career.

Finally, but not least, I must thank my friends, specially the ones that helped me when I needed it, and made this time something else.

References

[1] Thomas Fuchs. http://script.aculo.us/.

[2] Sbastien Gruhier. http://prototype-window.xilinus.com/.

[3] IEEE. *A Web Service Based Micro-payment System*, 2006.

[4] Mahesh Tripunitara; Tom Messergers. Resolving the micropayment problem. 2007.

[5] Rbert Prhonyi; Lambert J.M. Nieuwenhuis; Aiko Pras. Second generation micropayment systems:lessons learned, 2005.

[6] Pierre Tholence. On demand proxy server and immortal sockets, 2009.

[7] W3C. Cross-origin resource sharing (w3c working draft 17 march 2009). *W3C*. http://www.w3.org/TR/access-control/.

[8] Abraham Williams. https://docs.google.com/view?docid=dcf2dzzs_2339fzbfsf4.