Making Global Business Happen

Developer's API Guide

IPS eWallet Web Services

International Payout Systems

2/5/2016

Legal Notice

International Payout Systems, Inc. (IPS) Developers API Guide contains information proprietary to IPS, and is intended only to be used in conjunction with IPS products and services.

Revision History

Date	Version	Description	Author
9/30/13	4.0	Initial version from complete API	Dima Polyakov
10/15/2013	4.1	[add] eWallet_GetCustomerDetails	Julian Garvin
12/10/2013	4.2	[add] eWallet Workflow	Julian Garvin
4/15/2014	4.3	Added JSON adapter	Dharmendar Mothe
7/2/2014	4.4	Performance notes	Dima Polyakov
7/15/2014	4.5	New “hash” parameter in notifications	Dima Polyakov
10/23/2014	4.6	Added new function eWallet_UpdateUserName	Dharmendar Mothe
11/14/2014	4.7	Added new function eWallet_RefundToCustomer	Dharmendar Mothe
2/12/2015	4.9	Updated Self user registration method (FAQ)	Dharmendar Mothe
5/29/2015	4.10	Added some more existing functions eWallet_CheckIfUserNameExists eWallet_DeclineMerchantPayment eWallet_UpdateUserAccountStatus eWallet_RefundPaymentToMerchant eWallet_RefundPartialPaymentToMerchant	Dharmendar Mothe
2/5/2016	4.11	Added numeric values to ENUMs in Appendixes A & F	Danny Gonzalez

Contents

IPS eWallet Web Service

eWallet Web Service is written in C# .Net 4.0

General eWallet Workflow

1. All members of your program should have eWallet accounts and registered using the function **eWallet_RegisterUser**. As users register in your system, you should automatically register an eWallet for them as well.

   Note: Once a user is registered with an eWallet, our system will automatically send a welcome email to the user with a confirmation link the user should click on in order to verify their email.

2. Once users are registered you will be able to accept payments from them. To do so, you will need to call the function **eWallet_AddCheckoutItems**. This will add an invoice to the user's eWallet account and they will need to login to their account to complete this payment.

   *We highly recommend the use of the auto login (**eWallet_RequestUserAutoLogin)** so that as the user requests to make a purchase in your system, you automatically log them into their eWallet so they can complete their payment seamlessly.

   When using eWallet_AddCheckoutItems, there are several things to note:
   a. A pending invoice(s) will be added to the specified eWallet. Once the user is logged into their eWallet, they will have multiple payments options to choose from (cash, bank transfer, wire, etc.)
   b. Once the payment is complete, a notification will be sent out to the notification URL that you specify in our management console.
      *Details on the notification parameters as well as how to setup your notification URL can be found in the eWallet_AddCheckoutItems section.

3. In order to issue commission payouts, do payroll, etc., the function **eWallet_Load** should be called. This will create a batch of eWallet loads in our management system that will need to be approved by a manager to be processed. Auto loads (skipping manual approval) can be setup upon request.

Test web service

URL: https://www.testewallet.com/eWalletWS/ws_eWallet.asmx

not recommended, use HTTP/JSON adapter

You can use test web service to integrate your system with IPS platform and register test users, place test transactions, and "play" with the system using test management console and test user's front end a website. Please contact IPS to obtain your API Merchant ID and Password.

Production web service

URL: https://www.i-payout.net/eWalletWS/ws_eWallet.asmx

not recommended, use HTTP/JSON adapter

Production web service is the same as it is in the test environment, except that all users and transactions are real and there are fees involved per user/transaction. Please contact IPS to obtain your API Merchant ID and Password.

If you use PHP, ASP, Perl, FoxPro, Java or .NET, it is highly recommended to use **Web Service HTTP or JSON Adapter** described in this document (please also see section "Performance notes"). You can also find examples: https://www.i-payout.com/api.

Please DO NOT USE PHP web service components that call WSDL before every function call. It is very inefficient as WSDL file is very big and your servers will suffer a performance and communication speed degradation.

Performance notes

Web service is driven by SOAP protocol that is in essence an XML. As convenient it might be using SOAP protocol in modern IDEs like MS Visual Studio, there is a performance penalty that each side (client and server) will suffer due to overhead of parsing XML and significant increase in data traffic. For example, average SOAP call to eWallet web service will be about 3.5kb, while the same call by using HTTP adapter will be only 900 bytes. On a magnitude of thousands calls, it might be a significant internet traffic difference between SOAP protocol and HTTP/JSON adapter.

It is highly recommended to use HTTP adapter or JSON adapter over traditional SOAP. Please review the examples in www.i-payout.com/api for HTTP/JSON adapters.

Please use production web service only after you have successfully passed all tests by using test web service.

eWallet API Functions

eWallet_RegisterUser

This function creates a user and his or her eWallet in the system.

Response **eWallet_RegisterUser**(Guid MerchantGUID, String MerchantPassword, String UserName, String FirstName, String LastName, String CompanyName, String Address1, String Address2, String City, String State, String ZipCode, String Country2xFormat, String PhoneNumber, String CellPhoneNumber, String EmailAddress, String SSN, String CompanyTaxID, String GovernmentID, String MilitaryID, String PassportNumber, String DriversLicense, DateTime DateOfBirth, String WebsitePassword, String DefaultCurrency, Boolean SkipAutoSVCOrder, String PreferredLanguage, Boolean IsBusinessUser, String BusinessUserName)

Input Fields:

Request Field	Data Type	Required	Comment
MerchantGUID	Guid	Yes	Merchant's ID.
MerchantPassword	String	Yes	Merchant's password.
UserName	String	Yes	Unique user name. It can be in any format.
FirstName	String	Yes	User's first name.
LastName	String	Yes	User's last name.
CompanyName	String	No	User's business name if own any
Address1	String	No	User's home/business address.
Address2	String	No	
City	String	No	
State	String	No	For US users state should be in 2 character format like FL, TX. For non US users, state might be an empty string.
ZipCode	String	No	For US users ZIP code should be in numeric format. For non US users it can be in any format that is used by his or her country.
Country2xFormat	String	No	Country should be in 2 character format, like US, CA. If not specified, default country is US
PhoneNumber	String	No	User's phone number
CellPhoneNumber	String	No	User's cell phone number. This number can be used for SMS notifications.
EmailAddress	String	Yes	Upon registration and after any financial activity, user will receive an email into this address.
SSN	String	No	User's social security number (US only).
CompanyTaxID	String	No	Company's EIN number (US only).
GovernmentID	String	No	Government ID. It can be for example a *cedula* number in some South American countries.
MilitaryID	String	No	Military ID if person does not have SSN.
PassportNumber	String	No	Required if any other id is not applicable.
DriversLicense	String	No	

DateOfBirth	DateTime	Yes	User's date of birth. Should be 18 or older. If date of birth is not known, please set it to: 1/1/1900. It will be asked on a first user login.
WebsitePassword	String	No	Login password to eWallet. If not specified a random password will be generated and email will be sent to the user. Cannot contain spaces or the symbols "<" or ">".
DefaultCurrency	String	No	Default currency of the eWallet. This currency code is used to pay non-transaction based fees, like: monthly maintenance fees, phone support, and so on. Transaction based fees (like eWallet to Bank account) are paid in the currency of that transaction. If not specified, it will be USD
SkipAutoSVCOrder	Boolean	No	If this flag is set to TRUE, auto prepaid card order will be skipped (no card will be ordered). This setting does not have an effect if auto ordering of prepaid cards is not set.
PreferredLanguage	String	No	Preferred language code that will be used for email communications: EN – English, DE – German, and so on. If not specified, it will be EN
IsBusinessUser	Boolean	No	Default value should be FALSE. Set TRUE if this user should be marked as a business user (only if merchant setup allows business users in the program).
BusinessUserName	String	No	Business user name to whom to affiliate this user to (only if merchant setup allows business users in the program).

Return:

Response Field	Data Type	Comment
m_Code	StatusConstants	See appendix A.
m_Text	String	Code and error description.
TransactionRefID	Int64	User's ID in IPS platform.

Example:

eWallet_GetCurrencyBalance

This function gets user's eWallet balance in the system.

Response **eWallet_GetCurrencyBalance**(Guid MerchantGUID, String MerchantPassword,
String UserName, String CurrencyCode, out Decimal Balance)

Input Fields:

Request Field	Data Type	Required	Comment
MerchantGUID	Guid	Yes	Merchant's ID.
MerchantPassword	String	Yes	Merchant's password.
UserName	String	Yes	Unique user name. It can be in any format.
CurrencyCode	String	Yes	3 character currency code.

Return:

Response Field	Data Type	Comment
m_Code	StatusConstants	See appendix A.
m_Text	String	Code and error description.
Balance	Decimal	Current eWallet balance.

eWallet_GetCustomerDetails

This function gets various user information.

Response **eWallet_GetCustomerDetails**(Guid MerchantGUID, String MerchantPassword,
String UserName, out CustomerResponse response)

Input Fields:

Request Field	Data Type	Required	Comment
MerchantGUID	Guid	Yes	Merchant's ID.
MerchantPassword	String	Yes	Merchant's password.
UserName	String	Yes	Unique user name. It can be in any format.

Return:

Response Field	Data Type	Comment
m_Code	StatusConstants	See appendix A.
m_Text	String	Code and error description.
response	CustomerResponse	Customer object with user's info. See appendix E.

eWallet_Load

This function makes a request to load eWallets.

Response **eWallet_Load**(Guid MerchantGUID, String MerchantPassword,
String PartnerBatchID, String PoolID, eWalletLoad[] arrAccounts, Boolean AllowDuplicates,
Boolean AutoLoad, String CurrencyCode)

Input Fields:

Request Field	Data Type	Required	Comment
MerchantGUID	Guid	Yes	Merchant's ID.
MerchantPassword	String	Yes	Merchant's password.
PartnerBatchID	String	Yes	It is a parameter that is used to group all loads into one batch. You may call **Wallet_Load** multiple times, and as long as the *PartnerBatchID* is the same. Please note that *PartnerBatchID* is used as a batch name in the management console. The best practice is to use a value that is easy readable like, "payout for commissions on 3/12/2012".
PoolID	String	No	Payout pool/level/tier. If you do not use pools, just leave it blank. This field is just a reference.
arrAccounts	eWalletLoad	Yes	Array of eWallet accounts to load. See appendix E.
AllowDuplicates	Boolean	Yes	Allow multiple same user names in the same *PartnerBatchID*
AutoLoad	Boolean	Yes	If set to TRUE - automatically approve and load payout into eWallets. A batch will be automatically closed, and no other loads can be added into this batch.

			NOTE: This flag must be enabled on IPS' side first. To enable, please send a request via email to compliance@i-payout.com. Once batch is approved and loaded, there is no way to cancel it. Please use this flag only when you double checked on your end that payout transactions you submit are legit and correct.
CurrencyCode	String	Yes	Currency code of the transaction.

Return:

Response Field	Data Type	Comment
m_Code	StatusConstants	See appendix A.
m_Text	String	Code and error description.

- If *AutoLoad = false* and return is NO_ERROR, eWallet loads are ready to be manually approved. To load eWallets a manager with sufficient permissions needs to approve payout by to logging into IPS management console and go to menu EWALLET -> Payout Requests Management. On that page, a manager can find all the payout batches to approve and make changes if necessary. Once a manager approves a batch, depending on your merchant setting, either all eWallet accounts will be loaded automatically or IPS management will need to confirm this load. This flag must be enabled on IPS' side first. To enable, please send a request via email to compliance@i-payout.com
- Once batch is approved by management, it cannot be modified by adding more payouts. If it is needed to make new payouts a new *PartnerBatchID* should be provided.

Example:

```
eWallet.ws_eWallet ew = new eWallet.ws_eWallet();

List<eWallet.eWalletLoad> arrAccounts = new List<eWallet.eWalletLoad>();

eWallet.eWalletLoad acc1 = new eWallet.eWalletLoad();
acc1.UserName = "ip001";
acc1.Amount = 100;
acc1.Comments = "Commission payout";// user will see it in the history
acc1. MerchantReferenceID = "abc123001";// user will not see it in the history
arrAccounts.Add(acc1);

eWallet.eWalletLoad acc2 = new eWallet.eWalletLoad();
acc2.UserName = "ip002";
acc2.Amount = 200;
acc2.Comments = “Reimbursement for order #12345”;
acc2. MerchantReferenceID = "abc123002";
arrAccounts.Add(acc2);

eWallet.Response resp = ew.eWallet_Load(MerchantGUID, MerchantPassword,
        "load 3/12/2012", "level 1", arrAccounts.ToArray(), true, true, “USD”);
```

```
if(resp.m_Code == eWallet.StatusConstants.NO_ERROR){ // SUCCESS
        …
} else {// FAILED
        String Error = resp.m_Text;
}
```

eWallet_GetUserAccountStatus

This function is used to get user's account status.

CustomerAccountStatusResponse **eWallet_GetUserAccountStatus** (Guid MerchantGUID, String MerchantPassword, String UserName)

Input Fields:

Request Field	Data Type	Required	Comment
MerchantGUID	Guid	Yes	Merchant's ID.
MerchantPassword	String	Yes	Merchant's password.
UserName	String	Yes	User name.

Return:

Response Field	Data Type	Comment
m_Code	StatusConstants	See appendix A.
m_Text	String	Code and error description.
AccStatus	CustomerAccountStatus	Status of the account: (see appendix F) • Closed – account is closed. User still can login only if there is balance on the account. System will send weekly emails saying to move funds out and user might be charged with a dormant fee. • Suspended – account is suspended. User cannot login at any case. • Opened – account is alive. User can send/receive funds.

eWallet_UpdateUserAccountStatus

This function is used to manually set a user account status into **Closed, Suspended, or Open** state.
NOTE: Users cannot login into **suspended** accounts, but users can log into **closed** accounts only if they have money on their eWallet to move out.

Response **eWallet_UpdateUserAccountStatus** (Guid MerchantGUID, String MerchantPassword, String UserName, CustomerAccountStatus Status)

Input Fields:

Request Field	Data Type	Required	Comment
MerchantGUID	Guid	Yes	Merchant's ID.
MerchantPassword	String	Yes	Merchant's password.
UserName	String	Yes	User name.

Status	CustomerAccountStatus	Yes	Status of the account: (see appendix F) • Closed – account is closed. User still can login only if there is balance on the account. • Suspended – account is suspended. User cannot login at any case. • Open – account is alive. User can send/receive funds.

Return:

Response Field	Data Type	Comment
m_Code	StatusConstants	See appendix A.
m_Text	String	Code and error description.

eWallet_AddCheckoutItems

This function is used to add items to purchase into eWallet check out page.

Response **eWallet_AddCheckoutItems**(Guid MerchantGUID, String MerchantPassword,
String UserName, CheckoutItem[] arrItems, Boolean AutoChargeAccount,
out Response[] arrItemsResponse)

Input Fields:

Request Field	Data Type	Required	Comment
MerchantGUID	Guid	Yes	Merchant's ID.
MerchantPassword	String	Yes	Merchant's password.
UserName	String	Yes	User name to check.
arrItems	Checkout Item array	Yes	Array of **CheckoutItem** items: (See appendix E) *Decimal* ***Amount*** – cost of the item. *String* ***CurrencyCode*** – currency code, like USD. *String* ***ItemDescription*** – description of the item. *String* ***MerchantReferenceID*** – reference ID in your system. It will be used in call back once item is paid. *String* ***UserReturnURL*** – if payment was done instantly (via credit card or eWallet balance), user can click on this URL after payment. *Boolean* ***MustComplete*** – flag to set if this payment can be canceled manually by user. If it is not mandatory for users to complete a payment, set it to **false** *(recommended value: false)*. *Boolean* ***IsSubscription*** – if **true** this checkout item will generate a monthly invoice. The first invoice will begin 1 month after this invoice has been successfully paid. *Decimal* ***SubscriptionAmount*** – monthly payment amount.
AutoChargeAccount	Boolean	Yes	**False (default)** – do not charge account automatically. User has to go to eWallet and confirm and select a payment method. To avoid unauthorized charges it is recommended to use **FALSE** so users can review an invoice and pay using their Security PIN.

			True – auto charge eWallet account if there is any of: 1) enough balance on eWallet 2) secondary payment account is set: verified bank account (US only) or credit card (subject to amount limitation per merchant setup, and a processor can process credit cards without CVV2). Please contact IPS to allow auto charge account if required

Return:

Response Field	Data Type	Comment
m_Code	StatusConstants	See appendix A.
m_Text	String	Code and error description.
arrItemsResponse	Response []	Array of Response objects for each submitted item in arrItems

Remarks:

This function will try to make:

1. Add a payment to the user
2. If **AutoChargeAccount** =
   True: instantly debit from eWallet. If there are insufficient funds then function will debit from a secondary payment account. System will send an email to a user about this transaction.
   False: System will send an email to a user to complete this transaction.
3. If function returned in **arrItemsResponse** m_Code =
   a. eWallet.StatusConstants.NO_ERROR – transaction was processed and completed successfully.
   b. eWallet.StatusConstants.PROCESSING – transaction was submitted without user interaction and it will take some time to complete this payment (for example, if default account is bank account (ACH) then it will take 2-3 days to complete).
   c. eWallet.StatusConstants.PENDING – transaction could not be submitted at this time and user's interaction is required. In this case the system will send a notification email to the user to complete the transaction.
4. You can call the function: **eWallet_Load** in order to add funds to test payment completions.

Upon a completion or user cancelation of this payment (if ***MustComplete*** flag is False), the eWallet notification system will call *Merchant Notify URL* with the following parameters:

1. act=**PaymentToMerchant**
2. status_id=status id of this transaction
3. status_desc=status description
4. trnx_id= MerchantReferenceID
5. log_id=IPS TransactionID (numeric), same as Response.LogTransactionID
6. hash=SHA1 hash of: trnx_id + log_id + MerchantGUID + MerchantPassword

Hash calculation
For example, you received a notification with the following parameters:
trnx_id = 54321, log_id = 8790
Your MerchantGUID = 6d099995-a2f9-4a01-bef1-258eb99c1a77, MerchantPassword = psd123
You would combine them into one string as: *5432187906d099995-e2f9-4a01-bef1-258eb99c1b77psd123*

and then compute the SHA1 hash of that string, which would generate hash as (in upper case)
F225DA02326628D9444468AB4868A7386D43C912
If your computed hash matches with a received one then this notification is valid and you can process this payment.

status_id can be only *settled (user paid invoice)* or *voided (user canceled invoice)*. "Payment to merchant" transactions cannot have status failed as funds are actually taken from user's eWallet balance.

Merchant Notify URL page will be called as a POST request with all parameters inside. In a response to that POST request, your page should write into a response output 2 characters "OK" meaning that you processed this notification. If "OK" is not received as a response, the eWallet notification system will try to call your page 4 more times with intervals about 1 min apart.

To set up your *Merchant Notify URL* you need to login into eWallet Management Console and set it up on System Overview page in the **eWallet Setup** section.

If ***IsSubscription*** = True **eWallet_AddCheckoutItems** will call internally **eWallet_AddUserRecurringPayin** function. But subscription will not be active until this payment settles. When this payment settles a subscription will become active (initial settlement day) and a next subscription invoice will be posted on user's eWallet a month from initial settlement day. All next subsequent invoices will be posted on eWallet every month relative to initial settlement day. For notifications on subscription payment please take a look at Remarks from **eWallet_AddUserRecurringPayin** function.

You can call a web service function **eWallet_FindTransaction** to get a status of this payment.

If initial payment is not required for the subscription, please use the function "**eWallet_AddRecurringPayIn**"

eWallet_GetCheckoutItems

This function is used to get information about previously submitted checkout items.

Response **eWallet_GetCheckoutItems**(Guid MerchantGUID, String MerchantPassword,
String UserName, String MerchantReferenceID, DateTime DateFrom, DateTime DateTo,
TransactionStatus Status, out CheckoutItem[] arrItems)

Input Fields:

Request Field	Data Type	Required	Comment
MerchantGUID	Guid	Yes	Merchant's ID.
MerchantPassword	String	Yes	Merchant's password.
UserName	String	No	User name of whom the items were posted to
MerchantReferenceID	String	No	Reference ID in your system
DateFrom	DateTime	Yes	Beginning date to search for items
DateTo	DateTime	Yes	End date to seach for items
Status	TransactionStatus (see Appendix C)	No	Transaction status to filter search: **All**: Retrieve all transactions regardless of status

			Pending: Retrieve transactions where no action has been taken to fund the item. **Processing**: Retrieve transactions that was awaiting a funding transaction to complete (bank transfer, wire, etc.) **Settled**: Retrieve transactions that have been completed. **Declined**: Retrieve transactions that have been canceled.

Return:

Response Field	Data Type	Comment
m_Code	StatusConstants	See appendix A.
m_Text	String	Code and error description.
arrItems	CheckoutItem array	Array of CheckoutItems found, (See appendix E)

eWallet_RequestUserAutoLogin

This function is used to auto login a user by using a security parameter.

Response **eWallet_RequestUserAutoLogin**(Guid MerchantGUID, String MerchantPassword,
String UserName)

Input Fields:

Request Field	Data Type	Required	Comment
MerchantGUID	Guid	Yes	Merchant's ID.
MerchantPassword	String	Yes	Merchant's password.
UserName	String	Yes	User name.

Return:

Response Field	Data Type	Comment
m_Code	StatusConstants	See appendix A.
m_Text	String	Code and error description.
ProcessorTransactionRefNumber	String	Security parameter

Remarks:

eWallet_RequestUserAutoLogin returns Response object with a field ProcessorTransactionRefNumber set to security parameter (**GUID**) that needs to be passed to MemberLogin.aspx page on eWallet website.

For example:

Auto login

https://merchant.globalewallet.com/MemberLogin.aspx?secKey=GUID

merchant = Your merchant name, ie. Demo.globalewallet.com

Auto login with page redirect

https://merchant.globalewallet.com/MemberLogin.aspx?secKey=**GUID**&LoginURL=MerchantProducts.aspx

LoginURL – is a page name in eWallet website.

eWallet_GetUserRecurringPayins

This function is used to get all pay-ins assigned to the user.

Response **eWallet_GetUserRecurringPayins**(Guid MerchantGUID, String MerchantPassword,
String UserName, out RecurringPayment[] Payins)

Input Fields:

Request Field	Data Type	Required	Comment
MerchantGUID	Guid	Yes	Merchant's ID.
MerchantPassword	String	Yes	Merchant's password.
UserName	String	Yes	User name.

Return:

Response Field	Data Type	Comment
m_Code	StatusConstants	See appendix A.
m_Text	String	Code and error description.
Payins	RecurringPayment[]	Array of recurring payments. See appendix E.

eWallet_GetUserActiveRecurringPayins

This function is used to get all active recurring pay-ins assigned to the user.

Response **eWallet_GetUserActiveRecurringPayins**(Guid MerchantGUID, String MerchantPassword,
String UserName, out RecurringPayment[] Payins)

Input Fields:

Request Field	Data Type	Required	Comment
MerchantGUID	Guid	Yes	Merchant's ID.
MerchantPassword	String	Yes	Merchant's password.
UserName	String	Yes	User name.

Return:

Response Field	Data Type	Comment
m_Code	StatusConstants	See appendix A.
m_Text	String	Code and error description.
Payins	RecurringPayment[]	Array of recurring payments. See appendix E.

eWallet_ClearUserRecurringPayins

This function is used to disable all recurring pay-ins currently assigned to the user.

Response **eWallet_ClearUserRecurringPayins**(Guid MerchantGUID, String MerchantPassword, String UserName)

Input Fields:

Request Field	Data Type	Required	Comment
MerchantGUID	Guid	Yes	Merchant's ID.
MerchantPassword	String	Yes	Merchant's password.
UserName	String	Yes	User name.

Return:

Response Field	Data Type	Comment
m_Code	StatusConstants	See appendix A.
m_Text	String	Code and error description.

eWallet_RemoveUserRecurringPayin

This function is used to disable a specific recurring pay-in assigned to the user.

Response **eWallet_RemoveUserRecurringPayin**(Guid MerchantGUID, String MerchantPassword, String UserName, Int64 PayinID, String MerchantReferenceID)

Input Fields:

Request Field	Data Type	Required	Comment
MerchantGUID	Guid	Yes	Merchant's ID.
MerchantPassword	String	Yes	Merchant's password.
UserName	String	Yes	User name.
PayinID	Int64	Yes/No	RecurringPaymentID. Set 0 if unknown.
MerchantReferenceID	String	Yes/No	Reference number for this recurring payment. Set empty if unknown.

Return:

Response Field	Data Type	Comment
m_Code	StatusConstants	See appendix A.
m_Text	String	Code and error description.

eWallet_ActivateUserRecurringPayin

This function is used to re-activate a specific recurring pay-in that was previously disabled.

Response **eWallet_ActivateUserRecurringPayin**(Guid MerchantGUID, String MerchantPassword, String UserName, Int64 PayinID, String MerchantReferenceID)

Input Fields:

Request Field	Data Type	Required	Comment
MerchantGUID	Guid	Yes	Merchant's ID.

MerchantPassword	String	Yes	Merchant's password.
UserName	String	Yes	User name.
PayinID	Int64	Yes/No	RecurringPaymentID. Set 0 if unknown.
MerchantReferenceID	String	Yes/No	Reference number for this recurring payment. Set empty if unknown.

Return:

Response Field	Data Type	Comment
m_Code	StatusConstants	See appendix A.
m_Text	String	Code and error description.

eWallet_AddUserRecurringPayin

This function is used to add a monthly recurring pay-ins to the user.

Response **eWallet_AddUserRecurringPayin**(Guid MerchantGUID, String MerchantPassword,
String UserName, String Description, Decimal Amount, String CurrencyCode,
String RetParams, DateTime StartDate, Boolean ClearActiveRecurringPayins,
String MerchantReferenceID, Int64 MaxStackableCount)

Input Fields:

Request Field	Data Type	Required	Comment
MerchantGUID	Guid	Yes	Merchant's ID.
MerchantPassword	String	Yes	Merchant's password.
UserName	String	Yes	User name.
Description	String	Yes	Description of the recurring payment.
Amount	Decimal	Yes	Amount to charge.
CurrencyCode	String	Yes	Desired currency
RetParams	String	Yes	Parameters to be returned with notification URL
StartDate	DateTime	Yes	Date to begin withdrawing payments
ClearActiveRecurringPayins	Boolean	Yes	If true, user's all current active recurring payins will be cancelled
MerchantReferenceID	String	Yes	Unique Reference number for this recurring payment. This is used for reporting & notification purposes.
MaxStackableCount	Int64	Yes	Recurring will be cancelled, if user didnot paid for this number of months. We recommend maximum of 2 only.

Return:

Response Field	Data Type	Comment
m_Code	StatusConstants	See appendix A.
m_Text	String	Code and error description.
TransactionRefID	Int64	Recurring Payment ID (PayinID) in our system

Remarks:

- Recurring pay-in tries to takes a full amount first from eWallet. If there are not enough funds the function will try to get a full amount from a secondary payment account (if any): bank account or credit card. Once funds are settled a notification will be sent the *Merchant Notification URL*.
- If user's payment failed (by any reason), an invoice will be posted on a user's account and email will be sent to the user saying that there is a pending invoice to pay.
- To set up your *Merchant Notify URL* you need to login into eWallet Management Console and set it up on System Overview page in the **eWallet Setup** section.
- MerchantReferenceID: This field must be unique for each recurring payment. With all the notifications regarding this recurring payment, our system will send this reference number to identify the payment.
- If recurring payment is cancelled by user or due to inactivity, our system will send a notification to your notification URL with the following parameters:
    1) act=eWalletRecurringPayment
    2) isactive=false
    3) trnx_id=MerchantReferenceID
    4) ret_params=recurring return parameters
    5) cancel_by_user=true/false
    6) cancel_by_inactivity=true/false
    7) cancel_reason=reason for the cancellation
- On each month, once user completed the recurring payment, our system will send the notification with the following parameters:
    1) act=PaymentToMerchant
    2) status_id=status id of this transaction
    3) status_desc=status description
    4) trnx_id= MerchantReferenceID
    5) ret_params=recurring return parameters
    6) log_id=IPS TransactionID (numeric), same as Response.LogTransactionID
    7) period=payment period in Month/Year format. Ex: 06/2012

eWallet_UpdateUserRecurringPayin

This function is used to update the user monthly recurring pay-in.

Response **eWallet_UpdateUserRecurringPayin**(Guid MerchantGUID, String MerchantPassword, String UserName, Int64 RecurringPaymentID, Decimal NewAmount, String NewCurrencyCode, DateTime NewStartDate, String NewMerchantReferenceID, String NewDescription, String UserComments)

Input Fields:

Request Field	Data Type	Required	Comment
MerchantGUID	Guid	Yes	Merchant's ID.
MerchantPassword	String	Yes	Merchant's password.
UserName	String	Yes	User name.
RecurringPaymentID	Int64	Yes	Recurring Payment ID of the subscription.
NewAmount	Decimal	Yes	Updated amount to charge.
NewCurrencyCode	String	Yes	Updated currency

NewStartDate	DateTime	Yes	Updated date to begin withdrawing payments
NewMerchantReferenceID	String	Yes	Unique Reference number for this recurring payment. This is used for reporting & notification purposes.
NewDescription	String	Yes	Updated description of the recurring payment.
UserComments	String	Yes	Small description about the reason to update. This text will be sent to the user via email

Return:

Response Field	Data Type	Comment
m_Code	StatusConstants	See appendix A.
m_Text	String	Code and error description.

eWallet_UpdateUserName

Response **eWallet_UpdateUserName** (Guid MerchantGUID, String MerchantPassword,
String CurrentUserName, String NewUserName, String Comments, out Response response)

Input Fields:

Request Field	Data Type	Required	Comment
MerchantGUID	Guid	Yes	Merchant's ID.
MerchantPassword	String	Yes	Merchant's password.
CurrentUserName	String	Yes	Current User name that you want to change
NewUserName	String	Yes	New User name
Comments	String	No	Reason to update

Return:

Response Field	Data Type	Comment
m_Code	StatusConstants	See appendix A.
m_Text	String	Code and error description.

eWallet_CheckIfUserNameExists

This function is used to check if a given user name exists.

Response **eWallet_CheckIfUserNameExists**(Guid MerchantGUID, String MerchantPassword, String UserName)

Input Fields:

Request Field	Data Type	Required	Comment
MerchantGUID	Guid	Yes	Merchant's ID.
MerchantPassword	String	Yes	Merchant's password.
UserName	String	Yes	User name to check.

Return:

Response Field	Data Type	Comment
m_Code	StatusConstants	See appendix A.
m_Text	String	Code and error description.

eWallet_DeclineMerchantPayment

This function will decline any pending merchant payment (invoice). The function will fail if a payment (invoice) is already settled, declined, or if there is any funding payment (i.e. Bank Deposit) that is pending as well.

Response **eWallet_DeclineMerchantPayment**(Guid MerchantGUID, String MerchantPassword,
String MerchantReferenceID)

Input Fields:

Request Field	Data Type	Required	Comment
MerchantGUID	Guid	Yes	Merchant's ID.
MerchantPassword	String	Yes	Merchant's password.
MerchantReferenceID	String	Yes	Merchant's transaction ID.

Return:

Response Field	Data Type	Comment
m_Code	StatusConstants	See appendix A.
m_Text	String	Code and error description.
LogTransactionID	Int64	Transaction ID in IPS system.

eWallet_RefundToCustomer

This function is used to refund the customer.

Response **eWallet_RefundToCustomer** (Guid MerchantGUID, String MerchantPassword,
String UserName, Decimal Amount, String Comments, String MerchantReferenceID,
String CurrencyCode)

Input Fields:

Request Field	Data Type	Required	Comment
MerchantGUID	Guid	Yes	Merchant's ID.
MerchantPassword	String	Yes	Merchant's password.
UserName	String	Yes	User Name of the customer
Amount	Decimal	Yes	Amount to credit
Comments	String	No	Reason to credit
MerchantReferenceID	String	Yes	Unique Merchant Transaction ID
CurrencyCode	String	Yes	3 character ISO currency code

Return:

Response Field	Data Type	Comment
m_Code	StatusConstants	See appendix A.
m_Text	String	Code and error description.
LogTransactionID	Int64	Transaction ID in IPS system.

eWallet_RefundPaymentToMerchant

This function is used to refund "eWallet to Merchant" transactions. These transactions can be created by using functions like **eWallet_AddCheckoutItems**.

Response **eWallet_RefundPaymentToMerchant**(Guid MerchantGUID, String MerchantPassword, String UserName, Int64 LogTransactionID, Boolean Refund_FundingTransactionFee, Boolean Refund_eWalletToMerchantFee, String Comments, String MerchantReferenceID)

Input Fields:

Request Field	Data Type	Required	Comment
MerchantGUID	Guid	Yes	Merchant's ID.
MerchantPassword	String	Yes	Merchant's password.
UserName	String	Yes	User name.
LogTransactionID	Int64	Yes	Log transaction ID that was received from **eWallet_AddCheckoutItems** or similar functions that create invoices.
Refund_FundingTransactionFee	Boolean	Yes	**True** – return a funding transaction fee from deposit to eWallet transaction like "Bank to eWallet" if applicable.
Refund_eWalletToMerchantFee	Boolean	Yes	**True** – return "eWallet to Merchant" fee.
Comments	String	No	Manager's comments.
MerchantReferenceID	String	Yes	A unique merchant's reference id to refund transaction in merchant's database.

eWallet_RefundPartialPaymentToMerchant

This function is used to refund partial amount of "eWallet to Merchant" transactions. These transactions can be created by using functions like **eWallet_AddCheckoutItems**.

Response **eWallet_RefundPartialPaymentToMerchant** (Guid MerchantGUID, String MerchantPassword, String UserName, Int64 LogTransactionID, Decimal Amount, String Comments, String MerchantReferenceID)

Input Fields:

Request Field	Data Type	Required	Comment
MerchantGUID	Guid	Yes	Merchant's ID.
MerchantPassword	String	Yes	Merchant's password.
UserName	String	Yes	User name.
LogTransactionID	Int64	Yes	Log transaction ID that was received from **eWallet_AddCheckoutItems** or similar functions that create invoices.
Amount	Decimal	Yes	Amount that your want to refund to the customer. This amount cannot be more than original invoice amount
Comments	String	No	Manager's comments.
MerchantReferenceID	String	Yes	A unique merchant's reference id to refund transaction in merchant's database.

Return:

Response Field	Data Type	Comment
m_Code	StatusConstants	See appendix A.
m_Text	String	Code and error description.

Remarks:

eWallet_ RefundPartialPaymentToMerchant will refund invoice for the specified amount to customer eWallet. The way how this function works is it credits back eWallet as “Merchant to eWallet” transactions. Then, if payment (invoice) was paid by another user (invoice forwarding), that amount will be transferred to that eWallet. In the case if there was a funding transaction from a credit card, this function will attempt to refund a used credit card automatically. In all other cases amount will stay on eWallet and user will have to move it manually. User will also receive an email about this refund and also email will be sent to user who paid an invoice in the case of forwarding.

eWallet Login/Password verification

In order for IPS to verify user's credentials for eWallet login, IPS can perform a web call to merchant's verification page to verify login name and password for this user. This option is valid only for merchants who want to have unified passwords for users on merchant's back end office and eWallet sites.

Via page call (GET)

Verify login/password of the user on merchant's site. Merchants are required to implement a verification web page if user names and passwords are stored on the merchant's server. Please contact IPS to register such page.

Example:
https://www.merchant.com/backoffice/validate_user.php?**UserName**=username&**Password**=userpassword

The response of this page should be only one character:
"1" – login info was successful
"0" – login failed

Notifications

Besides notifications outlined in API functions there are more notifications that eWallet platform can send. All notification will be sent to *Merchant Notify URL*. To set up your *Merchant Notify URL* you need to login into eWallet Management Console and set it up on System Overview page in **eWallet Setup** section.

Payment completion

Please see eWallet_AddCheckoutItems function remarks.

Recurring Payment completion

Please see eWallet_AddCheckoutItems AND eWallet_AddUserRecurringPayin functions remarks.

New Account Created

When a new eWallet account is created IPS services will call *Merchant Notify URL* with the following parameters:
act=**AccCREATED**&user=*UserName*

Example: https*://www.merchant.com/ips_notifications.php?*act=**AccCREATED**&user=JohnDow

Account Status Notification of Change

If a user account has changed its status IPS services will call *Merchant Notify URL* with the following parameters:
act=**AccNOC**&user=*UserName*&status=[CLOSED|OPENED]

Example: https*://www.merchant.com/ips_notifications.php?*act=**AccNOC**&user=JohnSmith&status=CLOSED

Manual Merchant Deposit

If requested, an eWallet transfer option can be added to the eWallet menu to allow users to transfer funds directly to the merchant, without the need for a specific product. If you would like this option available for your users, please contact the IPS technical department. When a user successfully completes this transfer, IPS services will call *Merchant Notify URL* with the following parameters:
act=**ManualMerchantDeposit**&user=UserName&amount=Amount&currencyCode=Currency

Example:
https*://www.merchant.com/ips_notifications.php?*act=**ManualMerchantDeposit**&user=JohnSmither&amount=11.67&currencyCode=USD

Web Service HTTP Adapter

Web Service HTTP Adapter is http request processor that can execute web service functions using regular form style POST/GET requests. It could be beneficial for languages that do not directly support web services like ASP, PHP, and Perl.

Test HTTP Adapter URL:

https://testewallet.com/eWalletWS/ws_Adapter.aspx

Production HTTP Adapter URL:

https://www.i-payout.net/eWalletWS/ws_Adapter.aspx

Here is a format how to call HTTP adapter: *URL*?fn=*functionName*&*parameters*
Where,
URL: HTTP Adapter URL
functionName: name of web service function, like: eWallet_RegisterUser
parameters: function parameters – same as **input fields** in this documentation per function. Please refer to the web service WSDL file for correct names of parameters. All required parameters should be passed, all missed parameters will be treated as: null, false or 0.

Format to make an array: [field1=value1**%26**field2=value2][field1=value3**%26**field2=value4]. If value contains character '=', you will need to double URL encrypt it.

Response format is similar to regular POST/GET format. It consists of Response structure with each member values are being URL-encoded. In order to parse a response correctly, you have to get a "response" value of a HTTP response and URL-decode it.

Examples:

1. Register user (eWallet_RegisterUser)

Request:

fn=eWallet_RegisterUser&MerchantGUID=12345678-e2f9-4a01-bef1-258eb99c1b77&MerchantPassword=12345&UserName=IP001&FirstName=WSTest&LastName=Test&CompanyName=&Address1=2500 E Hallandale Beach Blvd&Address2=Suite 800&City=Hallandale Beach&State=FL&ZipCode=33009&Country2xFormat=US&PhoneNumber=954-123-4567&CellPhoneNumber=&EmailAddress=IP001@aol.com&SSN=123-45-6789&CompanyTaxID=&GovernmentID=&MilitaryID=&PassportNumber=&DriversLicense=&DateOfBirth=9/18/1945&WebsitePassword=1111&DefaultCurrency=USD

Successful response:

response=m_Code%3dNO_ERROR%26m_Text%3dOK%26LogTransactionID%3d0%26TransactionRefID%3d1302%26ACHTransactionID%3d0%26ProcessorTransactionRefNumber%3d%26CustomerFeeAmount%3d0%26CurrencyCode%3d

Error response:

response=m_Code%3dINTERNAL_ERROR%26m_Text%3dSSN+is+Required+for+US+Customers%26LogTransactionID%3d0%26TransactionRefID%3d0%26ACHTransactionID%3d0%26ProcessorTransactionRefNumber%3d%26CustomerFeeAmount%3d0%26CurrencyCode%3d

2. Get Customer Balance (eWallet_GetCurrencyBalance)

Request:

fn=eWallet_GetCurrencyBalance&MerchantGUID=12345678-e2f9-4a01-bef1-258eb99c1b77&MerchantPassword=12345&UserName=IP001&CurrencyCode=USD

Response:

response=m_Code%3dNO_ERROR%26m_Text%3dOK%26LogTransactionID%3d0%26TransactionRefID%3d0%26ACHTransactionID%3d0%26ProcessorTransactionRefNumber%3d%26CustomerFeeAmount%3d0%26CurrencyCode%3dUSD&Balance=868.1700

3. Payout to User (eWallet_Load)

Request:

fn=eWallet_Load&MerchantGUID=12345678-e2f9-4a01-bef1-258eb99c1b77
&MerchantPassword=12345&PartnerBatchID=test_load2&PoolID=test&arrAccounts=[UserName=IP001**%26**Amount=100.00**%26**Comments=commissions**%26**MerchantReferenceID=REFID001][UserName=IP587**%26**Amount=200.00**%26**Comments=eWallet+Load**%26**MerchantReferenceID=REFID002]&CurrencyCode=USD

Response:

response=m_Code%3dNO_ERROR%26m_Text%3dOK%26LogTransactionID%3d0%26TransactionRefID%3d0%26ACHTransactionID%3d0%26ProcessorTransactionRefNumber%3d%26CustomerFeeAmount%3d0%26CurrencyCode%3d

4. Add Invoice to User (eWallet_AddCheckoutItems)

Request:

fn=eWallet_AddCheckoutItems&MerchantGUID=12345678-e2f9-4a01-bef1-258eb99c1b77&MerchantPassword12345&UserName=IP001&arrItems=[Amount=10.79**%26**CurrencyCode=USD**%26**ItemDescription=Payment 1**%26**MerchantReferenceID=MyDatabaseTransactionRef1**%26**UserReturnURL=http%3a%2f%2fwww.mysite.com%2fThankyou.aspx**%26**MustComplete=true][Amount=15.79%26CurrencyCode=USD%26ItemDescription=Payment 2**%26**MerchantReferenceID=MyDatabaseTransactionRef2**%26**UserReturnURL= http%3a%2f%2fwww.mysite.com%2fThankyou.aspx**%26**MustComplete=false]

Response:

response=m_Code%3dNO_ERROR%26m_Text%3d%26LogTransactionID%3d0%26TransactionRefID%3d0%26ACHTransactionID%3d0%26ProcessorTransactionRefNumber%3d%26CustomerFeeAmount%3d0%26CurrencyCode%3d

Web Service JSON Adapter

Web Service JSON Adapter can be used to execute web service functions by posting parameters in JSON format. To send JSON request please use HTTP **POST** method to test or production JSON adapter URL.

Test JSON Adapter URL:

https://testewallet.com/eWalletWS/ws_JsonAdapter.aspx

Production JSON Adapter URL:

https://www.i-payout.net/eWalletWS/ws_JsonAdapter.aspx

JSON Request Format:

{"fn":"web service function name", “parameter_name”:”parameter_value”}

- fn *(function Name)*: name of web service function, like: eWallet_RegisterUser (name of the function must match any function outlined in this document)
- *parameter_name & parameter_value*: function parameters – same as **input fields** in this documentation per function. Please refer to the web service WSDL file for correct names of parameters. All required parameters should be passed, all missed parameters will be treated as: null, false or 0.

JSON Response Format:

Response format will also be in JSON format. It consists of Response structure with each member values.

Examples:

1. Register user (eWallet_RegisterUser)

Request JSON:

{"fn":"eWallet_RegisterUser","MerchantGUID":"12345678-e2f9-4a01-bef1-258eb99c1b77","MerchantPassword":"1a#@Sdr","UserName":"JsonTestUser1","FirstName":"John","LastName":"Doe","CompanyName":"","Address1":"2500 E Hallandale Beach","Address2":"Suite 800","City":"Hallandale beach","State":"FL","ZipCode":"33009","Country2xFormat":"US","PhoneNumber":"9545133150","CellPhoneNumber":"","EmailAddress":"dharmendar@i-payout.com","SSN":"","CompanyTaxID":"","GovernmentID":"","MilitaryID":"","PassportNumber":"","DriversLicense":"","DateOfBirth":"1980-01-01T00:00:00","WebsitePassword":"","DefaultCurrency":"","SkipAutoSVCOrder":true,"PreferredLanguage":"","IsBusinessUser":false,"BusinessUserName":""}

Successful Response JSON:

{"response":{"m_Code":0,"m_Text":"OK","LogTransactionID":0,"TransactionRefID":230004,"ACHTransactionID":0,"ProcessorTransactionRefNumber":"","CustomerFeeAmount":0.0,"CurrencyCode":null}}

Error Response JSON:

{"response":{"m_Code":-1,"m_Text":"Country should be in 2 letter format, like US","LogTransactionID":0,"TransactionRefID":0,"ACHTransactionID":0,"ProcessorTransactionRefNumber":"","CustomerFeeAmount":0.0,"CurrencyCode":null}}

2. Payout to User (eWallet_Load)

Request JSON:

{"fn":"eWallet_Load","MerchantGUID":"12345678-e2f9-4a01-bef1-258eb99c1b77","MerchantPassword":"12#@Sdr","PartnerBatchID":"635330851376218624","PoolID":"635330851376218624","arrAccounts":[{"UserName":"ip123","Amount":10.50,"Comments":"Test Json","MerchantReferenceID":"15645121"},{"UserName":"ip001","Amount":20.50,"Comments":"Test Json","MerchantReferenceID":"1455645121"}],"AllowDuplicates":true,"AutoLoad":false,"CurrencyCode":"USD"}

Successful Response JSON:

{"response":{"m_Code":0,"m_Text":"OK","LogTransactionID":0,"TransactionRefID":32493,"ACHTransactionID":0,"ProcessorTransactionRefNumber":"","CustomerFeeAmount":0.0,"CurrencyCode":null}}

Error Response JSON:

{"response":{"m_Code":-2,"m_Text":"Customer with user name nousername is not found","LogTransactionID":0,"TransactionRefID":0,"ACHTransactionID":0,"ProcessorTransactionRefNumber":"","CustomerFeeAmount":0.0,"CurrencyCode":null}}

3. Add Invoice to User (eWallet_AddCheckoutItems)

Request JSON:

{"fn":"eWallet_AddCheckoutItems","MerchantGUID":"12345678-e2f9-4a01-bef1-258eb99c1b77","MerchantPassword":"12#@Sdr","UserName":"ip123","arrItems":[{"Amount":10.50,"CurrencyCode":"USD","ItemDescription":"Test Payment using Json","MerchantReferenceID":"5487adfdsf","UserReturnURL":"","MustComplete":false,"Status":0,"IsSubscription":false,"SubscriptionAmount":0.0,"SubscriptionDaysToStart":0}],"AutoChargeAccount":false}

Successful Response JSON:

{"response":{"m_Code":0,"m_Text":"","LogTransactionID":0,"TransactionRefID":0,"ACHTransactionID":0,"ProcessorTransactionRefNumber":"","CustomerFeeAmount":0.0,"CurrencyCode":null},"arrItemsResponse":[{"m_Code":-57,"m_Text":"","LogTransactionID":183078,"TransactionRefID":91506,"ACHTransactionID":0,"ProcessorTransactionRefNumber":"","CustomerFeeAmount":0.0,"CurrencyCode":"USD"}]}

Error Response JSON 1:

{"response":{"m_Code":-2,"m_Text":"User not found","LogTransactionID":0,"TransactionRefID":0,"ACHTransactionID":0,"ProcessorTransactionRefNumber":"","CustomerFeeAmount":0.0,"CurrencyCode":null},"arrItemsResponse":[]}

Error Response JSON 2:

{"response":{"m_Code":0,"m_Text":"","LogTransactionID":0,"TransactionRefID":0,"ACHTransactionID":0,"ProcessorTransactionRefNumber":"","CustomerFeeAmount":0.0,"CurrencyCode":null},"arrItemsResponse":[{"m_Code":-3,"m_Text":"Such Reference ID: adsf898 already exists in the system","LogTransactionID":0,"TransactionRefID":0,"ACHTransactionID":0,"ProcessorTransactionRefNumber":"","CustomerFeeAmount":0.0,"CurrencyCode":null}]}

4. Auto login User (eWallet_RequestUserAutoLogin)

Request JSON:

{"fn":"eWallet_RequestUserAutoLogin","MerchantGUID":"12345678-e2f9-4a01-bef1-258eb99c1b77","MerchantPassword":"12#@Sdr","UserName":"ip123"}

Successful Response JSON:

{"response":{"m_Code":0,"m_Text":"OK","LogTransactionID":0,"TransactionRefID":0,"ACHTransactionID":0,"ProcessorTransactionRefNumber":"b0eb897c-4fb2-40ee-bbc6-5a71bcf46fcb","CustomerFeeAmount":0.0,"CurrencyCode":null}}

Error Response JSON:

{"response":{"m_Code":-2,"m_Text":"User not found","LogTransactionID":0,"TransactionRefID":0,"ACHTransactionID":0,"ProcessorTransactionRefNumber":"","CustomerFeeAmount":0.0,"CurrencyCode":null}}

5. Get Customer Balance (eWallet_GetCurrencyBalance)

Request JSON:

{"fn":"eWallet_GetCurrencyBalance","MerchantGUID":"12345678-e2f9-4a01-bef1-258eb99c1b77","MerchantPassword":"12#@Sdr","UserName":"ip123","CurrencyCode":"USD"}

Successful Response JSON:

{"response":{"m_Code":0,"m_Text":"OK","LogTransactionID":0,"TransactionRefID":0,"ACHTransactionID":0,"ProcessorTransactionRefNumber":"","CustomerFeeAmount":0.0,"CurrencyCode":"USD"},"Balance":1710.5600}

Error Response JSON:

{"response":{"m_Code":-2,"m_Text":"User not found","LogTransactionID":0,"TransactionRefID":0,"ACHTransactionID":0,"ProcessorTransactionRefNumber":"","CustomerFeeAmount":0.0,"CurrencyCode":null},"Balance":0.0}

PHP Examples

eWallet_RegisterUser

```
$url = "https://testewallet.com/eWalletWS/ws_Adapter.aspx";
$post = "fn=eWallet_RegisterUser&MerchantGUID=MERCHANT_ID&MerchantPassword= PASSWORD
&UserName=USERNAME&FirstName=WSTest&LastName=Test&CompanyName=&Address1=2500 E Hallandale
Beach Blvd&Address2=Suite 800&City=Hallandale Beach&State=FL&ZipCode=33009&Country2xFormat=US
&PhoneNumber=954-123-4567&CellPhoneNumber=&EmailAddress=user@aol.com&SSN=123-45-
6789&CompanyTaxID=&GovernmentID=&MilitaryID=&PassportNumber=&DriversLicense=&DateOfBirth=9/18/1
945&WebsitePassword=password&DefaultCurrency=USD";

$ch = curl_init($url);
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);
curl_setopt($ch, CURLOPT_POST, 1);
curl_setopt($ch, CURLOPT_POSTFIELDS, $post);
curl_setopt($ch, CURLOPT_FOLLOWLOCATION, 1);
curl_setopt($ch, CURLOPT_HEADER, 0);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
$ips_response = curl_exec($ch);
curl_close($ch);

parse_str($ips_response);
//echo $response;

parse_str($response);
echo $m_Code; // if success it should be NO_ERROR
```

eWallet_GetCurrencyBalance

```
$url = "https://testewallet.com/eWalletWS/ws_Adapter.aspx";
$post = "fn=eWallet_GetCurrencyBalance&MerchantGUID=MERCHANT_ID&MerchantPassword=PASSWORD
&UserName=USERNAME&CurrencyCode=USD";

$ch = curl_init($url);
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);
curl_setopt($ch, CURLOPT_POST, 1);
curl_setopt($ch, CURLOPT_POSTFIELDS, $post);
curl_setopt($ch, CURLOPT_FOLLOWLOCATION, 1);
curl_setopt($ch, CURLOPT_HEADER, 0);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
$ips_response = curl_exec($ch);
curl_close($ch);
```

```
parse_str($ips_response);
//echo $response;

parse_str($response);
echo $m_Code; // if success it should be NO_ERROR
echo $Balance; // user's eWallet balance
```

How to reply with "OK" on invoice notifications:

```
<?php  echo "OK"; exit; ?>
```

Appendix A – Status Constants

Name	Value	Description
NO_ERROR	0	Success.
INTERNAL_ERROR	-1	Internal error. Can happen if something is not set up right or database error.
CUSTOMER_NOT_FOUND	-2	User not found.
GENERAL_ERROR	-3	Any other error, look at m_Text field for an error description.
INVALID_PASSWORD	-4	Invalid password.
ACCOUNT_NOT_FOUND	-5	Account not found.
NOT_AVAILABLE_FOR_THIS_TYPE	-6	Functionality is not available for this type of account.
INVALID_CLIENT_CREDENTIALS	-7	Invalid client credentials.
CUSTOMER_ALREADY_LOGGED_IN	-8	User already logged in.
METHOD_IS_AVAILABLE_BUT_OBSOLETE	-9	Method is declared unnecessary/obsolete. Please contact IPS.
USER_NOT_LOGGED_IN	-10	User is not logged in or timeout occurred.
PROCESSORID_NOT_FOUND	-11	Processor is not found. It means that there is a problem with configuration in merchant's profile. Please contact IPS.
USERNAME_EXISTS	-12	A user with this User Name already exists. It may happen during the registration.
EMAIL_EXISTS	-13	A user with this email already exists.
SSN_EXISTS	-14	A user with this SSN already exists.
INSUFFICIENT_FUNDS	-15	Insufficient funds.
NO_DEFAULT_CARD	-16	There is no default prepaid card set up for this user.
CARD_NOT_EXISTS	-17	Card does not exist.
CARD_ALREADY_ASSIGNED	-18	Card is already assigned to another user.
INVALID_PIN	-19	Invalid PIN.
PROCESSORID_FOR_CARD_NOT_FOUND	-20	Cannot find Processor ID for the card number.
ALIAS_ALREADY_EXISTS	-21	This alias/nick name already exists for this user. All aliases must be unique per user.
ERROR_SAVING_CUSTOMER_DETAILS	-22	Error saving customer details.
DRIVERLICENSE_EXISTS	-23	Driver's license already exists. All DL must be unique per merchant.
PASSPORT_EXISTS	-24	Passport number already exists.
CARD_ALREADY_ASSIGNED_TO_YOU	-25	Prepaid card is already assigned to the user.
INVALID_CARD_NUMBER	-26	Invalid card number.
CARD_ALREADY_ACTIVATED	-27	Card is already activated.
CARD_STATUS_NOT_ISSUED_NOR_ACTIVE	-28	Card status is not issued or not active.
CARD_NOT_ACTIVE	-29	Card is not Active.
PROCESSOR_ERROR	-30	Specific processor's error.
NETWORK_FAILURE	-31	Network failure.
INVALID_ALIAS	-32	Invalid alias.

CANT_APPEND_TO_DESCRIPTION	-33	Cannot append text to transaction description.
CARD_ALREADY_ACTIVATED_NEED_PIN	-34	Card is already activated. PIN is required.
PAL_ALIAS_ALREADY_EXISTS	-35	Such pal's alias already exists.
PAL_IS_ALREADY_ADDED	-36	You already added this pal.
PAL_INVALID_EMAIL	-37	Cannot find pal by such email.
TOTAL_AMOUNT_EXCEEDS_MAX_BALANCE	-38	Resulting total balance exceeds the max balance allowed on the Card.
TOTAL_MONTH_TRANSFER_EXCEEDS _MAX_MONTHLY_TRANSFER	-39	Total monthly transfer exceeds the max monthly transfer allowed.
TOTAL_DAY_TRANSFER_EXCEEDS _MAX_DAY_TRANSFER	-40	Total daily transfer exceeds the max daily transfer allowed.
ALL_ACHS_FAILED	-41	All ACHs failed.
CANNOT_RETRIEVE_PIN	-42	Cannot retrieve PIN.
CANNOT_INSERT_BANK_ACCOUNT	-43	Cannot insert bank account.
CANNOT_TRANSFER_TO_BANK_ACCOUNT	-44	Cannot transfer to bank account.
INVALID_BANKID	-45	Invalid Bank ID.
INVALID_VIRTUAL_ACCOUNT_ID	-46	Invalid eWallet ID.
INVALID_CARDID	-47	Invalid Card ID.
INVALID_ACH_PROCESSOR	-48	There is no ACH Processor defined in the database.
MILITARY_ID_ALREADY_EXISTS	-49	Military id already exists.
INCOMPATIBLE_SOURCE_AND _DESTINATION_ACCOUNTS	-50	Incompatible source and destination accounts.
INVALID_PSEUDODDA_NUMBER	-51	The Pseudo DDA number and/or the routing number of the card are/is invalid.
MOBILE_PASSWORD_IS_NOT_SET	-52	Mobile password is not setup.
INVALID_MERCHANT	-53	Invalid Merchant.
NOT_ACTIVE	-54	Not active.
NOT_PAID	-55	Not paid.
NO_RECORDS_FOUND	-56	No records found.
PENDING	-57	Transaction is Pending.
PROCESSING	-58	Transaction is Processing.
EMAIL_REQUIRED	-59	Email is required
INVALID_COUNTRY_CODE	-60	Invalid Country Code
ACCESS_DENIED	-61	Access permission is required to use this function

Appendix B – Currency Codes

Supported Currency Codes: *(if you do not see your currency code, please contact IPS to include it)*

USD	United States Dollars
EUR	Euro
GBP	Great Britain Pound
AUD	Australian Dollars
JPY	Japanese Yen
CAD	Canadian Dollars
CNY	Chinese Yuan
HKD	Hong Kong Dollar
CHF	Swiss Franc
NZD	New Zealand Dollar
THB	Thai Baht
SGD	Singapore Dollar

Appendix C – Transaction Status Codes

- **Declined** – Transaction has been declined.
- **Pending** – Transaction is pending. Usually ACH transactions have such status when they are sent to the ACH processor but have not settled yet.
- **Settled** – Transaction is settled.
- **Unknown** – Transaction status is unknown or cannot be retrieved.
- **Voided** – Transaction has been voided (usually by manager).
- **Unsent** – Transaction has not been sent yet.
- **Sent** – Transaction has been sent to the processor.
- **SentButNoReply** – Transaction has been sent to the processor but there was no reply.
- **Processing** – Transaction is in a processing state.
- **All** – All transactions

Appendix D – Custom Objects

This section will describe the custom objects which are used in the above API functions.

eWalletLoad

Field	Data Type	Comment
UserName	String	eWallet user name of the customer
Amount	Decimal	Commission payout amount
Comments	String	Short description for the commission payout (user will see it in a history page)
MerchantReferenceID	String	Reference id from merchant's platform. MerchantReferenceID can be used in **eWallet_FindTransaction** method.

CustomerResponse

Field	Data Type	Comment
UserName	String	eWallet user name
CustomerGuid	String	eWallet user guid
IsActivated	Boolean	True: Customer is activated to login
Email	String	Customer email
FirstName	String	Customer first name
LastName	String	Customer last name
CompanyName	String	Customer company name
Phone	String	Customer phone number
CellPhoneNumber	String	Customer cell number
eWalletID	String	Customer eWallet ID
Address1	String	Customer address
Address2	String	Customer address2
State	String	Customer state
City	String	Customer city
ZipCode	String	Customer zip code
Country	String	2 character Customer country code ie. 'US'
DateOfBirth	DateTime	Customer date of birth
IsSuspended	Boolean	True: Customer is suspended
IsInfoVerified	Boolean	True: Customer profile is verified on the first login
IsClosed	Boolean	True: Customer eWallet is closed
CreatedDate	DateTime	Customer registered date
IsAgreedToFees	Boolean	True: Customer has agreed to eWallet fees
IsBusiness	Boolean	True: Customer eWallet is of type "Business"
IsInvalidEmail	Boolean	True: Customer email address is invalid
PreferredLanguage	String	Language code for the emails from eWallet
SVCShippingAddress	String	Shipping address for the ordered stored value card

CheckoutItem

Field	Data Type	Comment
Amount	Decimal	Cost of the item
CurrencyCode	String	Currency code, like USD
ItemDescription	String	Description of the item
MerchantReferenceID	String	Reference ID in your system. It will be used in call back once item is paid
UserReturnURL	String	If payment was done instantly (via credit card), user will have an option to click on this URL to redirect back to merchant.
MustComplete	Boolean	Flag to see if payment must be complete. For example: if user has to pay for initial subscription, this flag should be true. If it is not mandatory for users to complete a payment, set it to false.
IsSubscription	Boolean	True is this invoice is a monthly subscription.
SubscriptionAmount	Decimal	monthly subscription amount
SubscriptionDaysToStart	int	Start day of a subscription. If 0 – next payment will be in a month from today. If > 0 then next time will be: today + SubscriptionDaysToStart. It could be used in scenario: 14 days free, then 19.99/m

UserBalance

Field	Data Type	Comment
UserName	String	eWallet user name of the customer
Amount	Decimal	Balance amount
CurrencyCode	String	3 character currency code like ‘USD’

RecurringPayment

Field	Data Type	Comment
RecurringPaymentID	Int64	IPS reference id for this recurring payment
Description	String	Small description about the recurring payment
Period	String	Recurring payment period like ‘monthly’
Amount	Decimal	Amount for the recurring payment
CurrencyCode	String	Currency code for the recurring payment like ‘USD’
RetParams	String	Custom return parameters that will be passed back (encoded) to the notification_url
StartDate	DateTime	(MM/dd/yyyy) Start Date of the recurring payment
MerchantReferenceID	String	Reference number for this recurring payment from your system
IsActive	Boolean	False: Recurring payment is cancelled
CancelDueToInactivity	Boolean	True: Recurring payment is cancelled by system because of not paid by user
CancelByUser	Boolean	True: Recurring payment is cancelled by user
CancelReason	String	Reason for the recurring payment cancellation

Appendix E – Custom Data Types (Enums)

This section will describe the custom variables (Enums) which are used in the above API functions.

CustomerAccountStatus

Value	Value	Description
Closed	-1	eWallet is closed. User still can login only if there is balance on the account. System will send weekly emails saying to move funds out and user might be charged with a dormant fee.
Suspended	0	eWallet is suspended. User cannot login at any case.
Open	1	eWallet is alive. User can send/receive funds.