API Integration Guide

Confidential - for professional use only

Revision 1.8

1 Preface

1.1 About this document

This document describes the technical integration of the PayToo payment system into sales/transactions systems based on SOAP and SSL.

1.2 Revisions

Changes in revision 1.8

- New functions: WalletSingleTransaction, WalletRecurringTransaction, RequestPayment
- Functions SingleTransaction and RecurringTransaction marked as deprecated

Changes in revision 1.7

- Mandatory fields added (first name, last name, full address)
- New paragraph for the mandatory fields and the API restrictions

Changes in revision 1.6

- Correction of URLs
- Third parties modules

Changes in revision 1.5

- Additional fields in PaytooAccountType definition ('ssn_number' & 'pos_id')
- New function 'EcheckRecurringTransaction'

Changes in revision 1.4

- Additional fields in PaytooBankAccountType definition ('bank_routing')
- Additional fields in PaytooAccountType definition ('level')
- New parameter ('country') in function RequestWireTransfer
- New function 'IsPaytooAccount'
- Java sample

Changes in revision 1.3

- Additional fields in PaytooCreditCardType definition

Changes in revision 1.2

- EcheckSingleTransaction() function
- RequestWireTransfer() function
- Additional fields in PaytooBankAccountType definition
- Additional fields in PaytooRequestSearchCriterias
- GoPaytoo branding

Changes in revision 1.1

- AchSingleTransaction() function
- PaytooBankAccountType definition

Changes in revision 1.0

- $amount parameter in refund function
- new Redeem() function

Changes in revision 0.9

- OTP for PreAuth on wallets is now requested after the PreAuth method and not the Settlement().
- PHP sample rewritten

1.3 System Overview

PayToo is a payment system that allows you to do quick, efficient and secure online transaction processing through an electronic wallet and also a number of third party and bank payments processing systems. PayToo uses real time processing with customer specified payment gateways to verify wallets details. PayToo stores the transactions details in an internal database.
With the PayToo Administration interface you are able to overview user transactions.
It also allows you to manage accounts and to view and analyse declined payments.

1.4 Definitions

merchant – Refers to the merchant of PayToo who is processing transactions for its customers

customer – Refers to the party attempting to process a transaction transaction – Any state or "event" with regards to a processing via the system
(i.e. sale transaction, refund transaction, void transaction, etc.)

OTP – One Time Password, also known as confirmation code: a 6 digits code received by the customer, that he must give to confirm a pending transaction.

request – Every call to the API will generate a request, identified by an unique ID in the Paytoo system.

transaction – A transaction is an act of payment. It is always associated to a request.

1.5 Request VS Transaction

In the Paytoo system, every call to the API will generate a request, identified by a unique ID in the system. This ID must be used as a reference to communicate with the Paytoo API.

In the other side, a transaction is a reference to a payment. So, every transaction is associated to a request, but a request has not necessarily a transaction associated. For example, if a request is rejected because of some invalid parameters, no transaction will be created.
So it is important to always refer to the request id, rather than the transaction id.

Starting from revision 0.7 of the API, the full request will always be returned in the response.

1.6 Documentation Problems

If you discover any errors or have any problems with this documentation, please e-mail us by following the instructions below. Please submit a detailed description of the error or problem you experienced.

Please include your account user name and submit via email to:
rd@paytoo.com

2 API Integration

2.1 General Information

2.1.1 Interface Location

The PayToo payment system uses SOAP to communicate with customer sales systems.

SOAP is a specification designed to allow systems to integrate with each other without knowledge of the language, operating system or engine behind the remote system.
It is based entirely from XML and is widely used in common languages.

The operations of this service and their parameters are detailed in the following document. You can also obtain a WSDL description of this service for integration ease.

It is important that the required parameters passed to the server are formatted with the correct data type, otherwise the transaction will not be accepted.

The SOAP web service is located at:
https://go.paytoo.com/api/merchant/

The WSDL document is located at:
https://go.paytoo.com/api/merchant/?wsdl

2.1.2 Testing / Sandbox

At the beginning of the integration process, a test account will be provided on a “sandbox”.
You will be plugged on the live environment only after several successful tests on the sandbox.

For the sandbox, the SOAP web service is located at:
https://go.paytoo.info/api/merchant/

and the WSDL document is located at:
https://go.paytoo.info/api/merchant/?wsdl

On the sandbox, the OTP is always 888888.

3 API Specifications

The following sections detail the operations that can be performed via the PayToo web service.

Each operation takes a number of fields (listed below) and returns a result indicating the status of each transaction. The returned result has always the same format.

3.1 Method Summary

MerchantApiResponse **auth** (*string* **$merchant_id**, *string* **$api_password**, [*string* ***$sub_account_id***=*null*])
MerchantApiResponse **logout** ()
MerchantApiResponse **AchSingleTransaction** (*PaytooBankAccountType* **$ PaytooBankAccount**, *PaytooAccountType* **$PaytooAccount**, *float* **$amount**, *string* **$currency**, *string* **$ref_id**, *string* **$description**, [*string* **$addinfo** = null])
MerchantApiResponse **ConfirmTransaction** (*integer* **$request_id**, *integer* **$OTP**)
MerchantApiResponse **CreditCardPreAuth** (*PaytooCreditCardType* **$PaytooCreditCard**, *PaytooAccountType* **$PaytooAccount**, *float* **$amount**, *string* **$currency**, *string* **$ref_id**, *string* **$description**, [*string* **$addinfo** = null])
MerchantApiResponse **CreditCardRecurringTransaction** (*PaytooCreditCardType* **$PaytooCreditCard**, *PaytooAccountType* **$PaytooAccount**, *float* **$initial_amount**, *float* **$amount**, *string* **$currency**, *string* **$periodicity**, *int* **$cycles**, *string* **$start_date**, *string* **$ref_id**, *string* **$description**, [*string* **$addinfo** = null])
MerchantApiResponse **CreditCardSingleTransaction** (*PaytooCreditCardType* **$PaytooCreditCard**, *PaytooAccountType* **$PaytooAccount**, *float* **$amount**, *string* **$currency**, *string* **$ref_id**, *string* **$description**, [*string* **$addinfo** = null])
MerchantApiResponse **EcheckRecurringTransaction** (*PaytooBankAccountType* **$ PaytooBankAccount**, *PaytooAccountType* **$PaytooAccount**, *float* **$initial_amount**, *float* **$amount**, *string* **$currency**, *string* **$periodicity**, *int* **$cycles**, *string* **$start_date**, *string* **$ref_id**, *string* **$description**, [*string* **$addinfo** = null])
MerchantApiResponse **EcheckSingleTransaction** (*PaytooBankAccountType* **$ PaytooBankAccount**, *PaytooAccountType* **$PaytooAccount**, *float* **$amount**, *string* **$currency**, *string* **$ref_id**, *string* **$description**, [*string* **$addinfo** = null])
MerchantApiResponse **GetStatus** ([*integer* **$request_id** = null], [*integer* **$tr_id** = null])
PaytooRequestType[] **GetRequests** ([*PaytooRequestSearchCriterias* ***$search_criterias*** = null])
MerchantApiResponse **MicroPayment** (*float* **$amount**, *string* **$currency**, [*string* **$country_or_ip** = null], *string* **$ref_id**, [*string* **$description** = null])
MerchantApiResponse **PreAuth** (*string* **$from**, *integer* **$security_code**, *float* **$amount**, *string* **$currency**, *string* **$ref_id**, *string* **$description**, [*string* **$addinfo** = null])
MerchantApiResponse **RecurringTransaction** (*string* **$from**, *integer* **$security_code**, *float* **$initial_amount**, *float* **$amount**, *string* **$currency**, *string* **$periodicity**, *int* **$cycles**, *string* **$start_date**, *string* **$ref_id**, *string* **$description**, [*string* **$addinfo** = null]) ***deprecated***
MerchantApiResponse **Redeem** (*string* **$voucher**)
MerchantApiResponse **Refund** ([*integer* **$request_id** = null], [*integer* **$tr_id** = null], [*float* **$amount** = null] [*string* **$reason** = null])
MerchantApiResponse **RequestPayment** (*PaytooAccountType* **$PaytooAccount**, *float* **$amount**, *string* **$currency**, *string* **$ref_id**, *string* **$description**, [*string* **$addinfo** = null])
MerchantApiResponse **RequestPayTooPIN** (*string* **$currency**, *string* **$cellphone**, *integer* **$security_code**, [*string* **$firstname** = null], [*string* **$lastname** = null], [*string* **$email** = null])
MerchantApiResponse **RequestWireTransfer** (*PaytooAccountType* **$PaytooAccount**, *float* **$amount**, *string* **$currency**, *string* **$country**, *string* **$ref_id**, *string* **$description**, [*string* **$addinfo** = null])
MerchantApiResponse **Settlement** (*integer* **$request_id,** [*float* **$amount**=null])
MerchantApiResponse **SingleTransaction** (*string* **$from**, *integer* **$security_code**, *float* **$amount**, *string* **$currency**, *string* **$ref_id**, *string* **$description**, [*string* **$addinfo** = null]) ***deprecated***
MerchantApiResponse **ValidatePayTooPIN** (*string* **$cellphone**, *string* **$pin**)

MerchantApiResponse **Void** (*integer* **$request_id**)
MerchantApiResponse **WalletSingleTransaction** (*string* **$from**, *float* **$amount**, *string* **$currency**, *string* **$ref_id**, *string* **$description**, [*string* **$addinfo** = null])
MerchantApiResponse **WalletRecurringTransaction** (*string* **$from**, *float* **$initial_amount**, *float* **$amount**, *string* **$currency**, *string* **$periodicity**, *int* **$cycles**, *string* **$start_date**, *string* **$ref_id**, *string* **$description**, [*string* **$addinfo** = null])
boolean **IsPaytooAccount** (*string* **$account_id**)

3.2 Methods

auth

Authentification function: **must be called once, before all other functions.**

MerchantApiResponse auth (string $merchant_id, string $api_password, [*string* ***$sub_account_id****=null*])

- string $merchant_id: Merchant ID
- string $api_password: Merchant password
- string $sub_account_id: Sub Account ID for which all transactions of this session will be associated (max 25 chars)

logout

Destroy all authentification token.

MerchantApiResponse logout ()

AchSingleTransaction

This function is used to create a single ACH. In most cases this will be a purchase in a web shop but it can be any sort of payment.

MerchantApiResponse AchSingleTransaction (*PaytooBankAccountType* **$PaytooBankAccount**, *PaytooAccountType* **$PaytooAccount**, *float* **$amount**, *string* **$currency**, *string* **$ref_id**, *string* **$description**, [*string* **$addinfo** = null])

- PaytooBankAccountType $PaytooBankAccount Bank account informations
- PaytooAccountType $PaytooAccount Customer informations
- float $amount: The requested amount
- string $currency: Currency code, only USD and EUR allowed
- string $ref_id: An id that uniquely refers to this transaction on your side (an order id for example)
- string $description: Short description of the transaction (will appears on the customer end)
- string $addinfo: Additional information you may want to store

ConfirmTransaction

The ConfirmTransaction operation will confirm a sale transaction, or a pre-auth transaction using a One Time Password sent to the customer on his cellphone and/or email address

MerchantApiResponse ConfirmTransaction (integer $request_id, integer $OTP)

- integer $request_id: Request ID to be confirmed
- integer $OTP: One Time Password, the confirmation code the customer has received

CreditCardPreAuth

This function is used to auth a credit card payment without settling it (Preauth). This request will be valid for 24 hours to 72 hours depending of the payment processor.

MerchantApiResponse CreditCardPreAuth (*PaytooCreditCardType* **$PaytooCreditCard**, *PaytooAccountType* **$PaytooAccount**, *float* **$amount**, *string* **$currency**, *string* **$ref_id**, *string* **$description**, [*string* **$addinfo** = null])

- PaytooCreditCardType $PaytooCreditCard Credit Card informations
- PaytooAccountType $PaytooAccount Customer informations
- float $amount: The requested amount
- string $currency: Currency code, only USD and EUR allowed
- string $ref_id: An id that uniquely refers to this transaction on your side (an order id for example)
- string $description: Short description of the transaction (will appears on the customer end)
- string $addinfo: Additional information you may want to store

CreditCardRecurringTransaction

This function is used to process a recurring credit card payment. In most cases this will be a monthly bill. For each recurring payment you will receive a post back notification.

MerchantApiResponse CreditCardRecurringTransaction (*PaytooCreditCardType* **$PaytooCreditCard**, *PaytooAccountType* **$PaytooAccount**, *float* **$initial_amount**, *float* **$amount**, *string* **$currency**, *string* **$periodicity**, *int* **$cycles**, *string* **$start_date**, *string* **$ref_id**, *string* **$description**, [*string* **$addinfo** = null])

- PaytooCreditCardType $PaytooCreditCard Credit Card informations
- PaytooAccountType $PaytooAccount Customer informations
- float $initial_amount: The requested amount
- float $amount: The requested amount
- string $currency: Currency code, only USD and EUR allowed
- string $periodicity Recurring period (days, weeks, months, years)
- int $cycles Number of cycles/periods (>1 or 0 for unlimited)
- string $start_date Date of the first transaction (format is YYYY-MM-DD)
- string $ref_id: An id that uniquely refers to this transaction on your side (an order id for example)
- string $description: Short description of the transaction (will appears on the customer end)
- string $addinfo: Additional information you may want to store

CreditCardSingleTransaction

This function is used to process a single payment using a credit card. In most cases this will be a purchase in a web shop but it can be any sort of payment.

MerchantApiResponse CreditCardSingleTransaction (*PaytooCreditCardType* **$PaytooCreditCard**,

PaytooAccountType **$PaytooAccount**, *float* **$amount**, *string* **$currency**, *string* **$ref_id**, *string* **$description**, [*string* **$addinfo** = null])

- PaytooCreditCardType $PaytooCreditCard Credit Card informations
- PaytooAccountType $PaytooAccount Customer informations
- float $amount: The requested amount
- string $currency: Currency code, only USD and EUR allowed
- string $ref_id: An id that uniquely refers to this transaction on your side (an order id for example)
- string $description: Short description of the transaction (will appears on the customer end)
- string $addinfo: Additional information you may want to store

EcheckRecurringTransaction

This function is used to process a recurring echeck payment. In most cases this will be a monthly bill. The first echeck has to be confirmed/signed by the customer, but not the next ones and you will receive a post back notification for each recurring payment.

MerchantApiResponse EcheckRecurringTransaction (*PaytooBankAccountType* **$PaytooBankAccount**, *PaytooAccountType* **$PaytooAccount**, *float* **$initial_amount**, *float* **$amount**, *string* **$currency**, *string* **$periodicity**, *int* **$cycles**, *string* **$start_date**, *string* **$ref_id**, *string* **$description**, [*string* **$addinfo** = null])

- PaytooBankAccountType $ PaytooBankAccount Bank account informations
- PaytooAccountType $PaytooAccount Customer informations
- float $initial_amount: The requested amount
- float $amount: The requested amount
- string $currency: Currency code, only USD and EUR allowed
- string $periodicity Recurring period (days, weeks, months, years)
- int $cycles Number of cycles/periods (>1 or 0 for unlimited)
- string $start_date Date of the first transaction (format is YYYY-MM-DD)
- string $ref_id: An id that uniquely refers to this transaction on your side (an order id for example)
- string $description: Short description of the transaction (will appears on the customer end)
- string $addinfo: Additional information you may want to store

EcheckSingleTransaction

This function is used to create a single eCheck. In most cases this will be a purchase in a web shop but it can be any sort of payment.

MerchantApiResponse EcheckSingleTransaction (*PaytooBankAccountType* **$PaytooBankAccount**, *PaytooAccountType* **$PaytooAccount**, *float* **$amount**, *string* **$currency**, *string* **$ref_id**, *string* **$description**, [*string* **$addinfo** = null])

- PaytooBankAccountType $PaytooBankAccount Bank account informations
- PaytooAccountType $PaytooAccount Customer informations
- float $amount: The requested amount
- string $currency: Currency code, only USD and EUR allowed

- string $ref_id: An id that uniquely refers to this transaction on your side (an order id for example)
- string $description: Short description of the transaction (will appears on the customer end)
- string $addinfo: Additional information you may want to store

GetStatus

The GetStatus operation will retrieve the current status of a request / transaction

MerchantApiResponse GetStatus ([integer $request_id = null], [integer $tr_id = null])

- integer $request_id: Request ID
- integer $tr_id: Transaction ID

GetRequests

This function will return a list of requests corresponding with the provided search criteria. If no criteria are provided, all requests will be returned.

PaytooRequestType GetRequests ([PaytooRequestSearchCriterias $search_criterias = null])

- PaytooRequestSearchCriterias $search_criterias: Search criteria

MicroPayment

Reserve a phone number for a micro payment. The phone number will be reserved for 15 mins.

MerchantApiResponse MicroPayment (float $amount, string $currency, [string $country_or_ip = null], string $ref_id, [string $description = null])

- float $amount: Requested amount (max. 10)
- string $currency: Currency code, only USD and EUR allowed
- string $country_or_ip Country code (2 chars) of the desired phone number or customer IP to autodetect the country and thus provide a local phone number
- string $ref_id: An id that uniquely refers to this transaction on your side (a session id for example).
- string $description: Short description of the transaction (will appears on the customer end)

PreAuth

This function is used to auth a payment without settling it. This request will be valid for 24h.

MerchantApiResponse PreAuth (string $from, integer $security_code, float $amount, string $currency, string $ref_id, string $description, [string $addinfo = null])

- string $from: PayToo account Identifier: it can be a wallet number, a registered phone number, an email address or a paytoo mobile phone number
- integer $security_code: The 6 digits security code of the PayToo account
- float $amount: The requested amount

- string $currency: Currency code, only USD and EUR allowed
- string $ref_id: An id that uniquely refers to this transaction on your side (an order id for example)
- string $description: Short description of the transaction (will appears on the customer end)
- string $addinfo: Additional information you may want to store

RecurringTransaction

Deprecated, replaced by function WalletRecurringTransaction.

This function is used to process recurring payments. In most cases this will be a monthly bill. For each recurring payment you will receive a post back notification.

MerchantApiResponse RecurringTransaction (string $from, integer $security_code, *float* **$initial_amount**, *float* **$amount**, *string* **$currency**, *string* **$periodicity**, *int* **$cycles**, *string* **$start_date**, *string* **$ref_id**, *string* **$description**, [*string* **$addinfo** = null])

- string $from: PayToo account Identifier: it can be a wallet number, a registered phone number, an email address or a paytoo mobile phone number
- integer $security_code: The 6 digits security code of the PayToo account
- float $initial_amount: The requested amount
- float $amount: The requested amount
- string $currency: Currency code, only USD and EUR allowed
- string $periodicity Recurring period (days, weeks, months, years)
- int $cycles Number of cycles/periods (>1 or 0 for unlimited)
- string $start_date Date of the first transaction (format is YYYY-MM-DD)
- string $ref_id: An id that uniquely refers to this transaction on your side (eg: an order id)
- string $description: Short description of the transaction (will appears on the customer end)
- string $addinfo: Additional information you may want to store

Refund

The ProcessRefund operation will perform a refund on a sale transaction, or a settled pre-auth transaction. If the transaction is NOT yet settled, it will attempt to perform a void. Refund a transaction means the merchant is debited the FULL amount of the transaction and a refund penalty may be incurred. The customer's account (or bank account) will be credited the amounts necessary.

MerchantApiResponse Refund ([integer $request_id = null], [integer $tr_id = null], [float $amount = null], [string $reason = null])

- integer $request_id: Request ID to be refunded
- integer $tr_id: Transaction ID to be refunded
- float $amount: Amount to refund, default is original request amount
- string $reason: The reason of this refund

RequestPayTooPIN

Request a PayToo PIN to create a new account The PIN will be valid for 24 hours

MerchantApiResponse RequestPayTooPIN (string $currency, string $cellphone, integer $security_code, [string $firstname = null], [string $lastname = null], [string $email = null])

- string $currency: Main currency code, only USD and EUR allowed
- string $cellphone: Cell phone on which you want to send the PayToo PIN / you want to register
- integer $security_code: A 6 digits security code chosen by the user
- string $firstname: User first name
- string $lastname: User last name
- string $email: User email address

RequestWireTransfer

This function is used to request a wire transfer to a customer. Instruction will be sent to the customer on his email address and cellphone. Once the wire transfer will be received, you will be notified by IPN.

MerchantApiResponse RequestWireTransfer (*PaytooAccountType* **$PaytooAccount**, *float* **$amount**, *string* **$currency**, *string* **$ref_id**, *string* **$description**, [*string* **$addinfo** = null])

- PaytooAccountType $PaytooAccount Customer informations
- float $amount: The requested amount
- string $currency: Currency code, only USD and EUR allowed
- string $country: Country code (2 alpha characters as per the ISO 3166-1)
- string $ref_id: An id that uniquely refers to this transaction on your side (an order id for example)
- string $description: Short description of the transaction (will appears on the customer end)
- string $addinfo: Additional information you may want to store

Settlement

The Settlement operation performs a settlement on a previous sale transaction that was processed as a pre-auth. You cannot perform this operation on a transaction with a state of anything other than pre-auth. Once a transaction has been settled, the merchant will be credited the amounts owed and the customer account will be debited.

MerchantApiResponse Settlement (integer $request_id, [float $amount=null])

- integer $request_id: Request ID of the approved Auth
- float $amount: The final amount of the transaction. It can be lower than the pre-auth amount. If not specified, it will be the same amount as the pre-auth.

Redeem

Redeem a Poutche Voucher

MerchantApiResponse Redeem (string $voucher)
string $voucher: The Poutche voucher

RequestPayment

This function is used to request a payment to anyone using any payment method. A link will be sent to the customer to make his payment.

MerchantApiResponse RequestPayment (*PaytooAccountType* **$PaytooAccount**, *float* **$amount**, *string* **$currency**, *string* **$ref_id**, *string* **$description**, [*string* **$addinfo** = null])

- PaytooAccountType $PaytooAccount Customer informations
- float $amount: The requested amount
- string $currency: Currency code, only USD and EUR allowed
- string $ref_id: An id that uniquely refers to this transaction on your side (an order id for example)
- string $description: Short description of the transaction (will appears on the customer end)
- string $addinfo: Additional information you may want to store

SingleTransaction

Deprecated, replaced by function WalletSingleTransaction.
This function is used to process single payments. In most cases this will be a purchase in a web shop but it can be any sort of payment.

MerchantApiResponse SingleTransaction (string $from, integer $security_code, float $amount, string $currency, string $ref_id, string $description, [string $addinfo = null])

- string $from: PayToo account Identifier: it can be a wallet number, a registered phone number, an email address or a paytoo mobile phone number
- integer $security_code: The 6 digits security code of the PayToo account
- float $amount: The requested amount
- string $currency: Currency code, only USD and EUR allowed
- string $ref_id: An id that uniquely refers to this transaction on your side (an order id for example)
- string $description: Short description of the transaction (will appears on the customer end)
- string $addinfo: Additional information you may want to store

StopRecurringTransaction

This function is used to stop a recurring payment.

MerchantApiResponse StopRecurringTransaction (integer $request_id, [string $reason=null])

- integer $request_id: Request ID of the approved initial request
- string $reason: The optional reason why you stop this recurring payment (will appears on the customer end)amount. If not specified, it will be the same amount as the pre-auth.

ValidatePayTooPIN

Validate a PayToo PIN previously requested with function RequestPayTooPIN

MerchantApiResponse ValidatePayTooPIN (string $cellphone, string $pin)

- string $cellphone: The cell phone on which the PayToo PIN has been sent
- string $pin: The PayToo PIN to validate

WalletSingleTransaction

This function is used to request a single payments from a wallet. In most cases this will be a purchase in a web shop but it can be any sort of payment.

MerchantApiResponse WalletSingleTransaction (string $from, float $amount, string $currency, string $ref_id, string $description, [string $addinfo = null])

- string $from: PayToo account Identifier: it can be a wallet number, a registered phone number, an email address or a paytoo mobile phone number
- float $amount: The requested amount
- string $currency: Currency code, only USD and EUR allowed
- string $ref_id: An id that uniquely refers to this transaction on your side (an order id for example)
- string $description: Short description of the transaction (will appears on the customer end)
- string $addinfo: Additional information you may want to store

WalletRecurringTransaction

This function is used to request a recurring payment from a wallet. In most cases this will be a monthly bill. For each recurring payment you will receive a post back notification.

MerchantApiResponse WalletRecurringTransaction (string $from, *float* **$initial_amount**, *float* **$amount**, *string* **$currency**, *string* **$periodicity**, *int* **$cycles**, *string* **$start_date**, *string* **$ref_id**, *string* **$description**, [*string* **$addinfo** = null])

- string $from: PayToo account Identifier: it can be a wallet number, a registered phone number, an email address or a paytoo mobile phone number
- float $initial_amount: The requested amount
- float $amount: The requested amount
- string $currency: Currency code, only USD and EUR allowed
- string $periodicity Recurring period (days, weeks, months, years)
- int $cycles Number of cycles/periods (>1 or 0 for unlimited)
- string $start_date Date of the first transaction (format is YYYY-MM-DD)
- string $ref_id: An id that uniquely refers to this transaction on your side (eg: an order id)
- string $description: Short description of the transaction (will appears on the customer end)
- string $addinfo: Additional information you may want to store

Void

The Void operation performs a void on a previous transaction that was processed as a pre-auth OR a pending micro payment. You cannot perform this operation on a transaction with a state of anything other than pre-auth. Voiding a transaction means the merchant is not credited the amounts and the customer will not receive a transaction on their account.

MerchantApiResponse Void (integer $request_id)

- integer $request_id: Request ID of the approved Auth

IsPaytooAccount

The IsPaytooAccount operation will check if a given identifier is a PayToo account or not and will simply return "true" or "false"

boolean **IsPaytooAccount** (*string* **$account_id**)

- *string* ***$account_id****: PayToo account Identifier: it can be a wallet number, a registered phone number, a prepaid card, an email address or a paytoo mobile phone number*

3.3 Complex Type

3.3.1 MerchantApiResponse

Response type returned by all functions.

Propertie	Type	Description
status	string	Has the value "OK" for successful, "PENDING" for pending transaction or "ERROR" for failure
request_id	int	Request ID is the unique ID of the request
request_status	string	Request status, it can be 'pending', 'accepted', 'rejected', 'cancelled', 'completed', 'refunded'
tr_id	int	Transaction ID is the unique ID of transaction associated to the request
sub_account_id	string	Sub Account ID for which you have associated this request
ref_id	string	Reference ID is the unique ID you've passed for this request
msg	string	Message
info	string	Additional informations
phone_number	string	* The phone number reserved for a micro payment in case of a MicroPayment response * The customer registered phone number or simcard phone number in all other case
w_number	string	Wallet number of the associated Paytoo account
PaytooTransaction	PaytooTransactionType	PayToo Transaction informations associated to a request/transaction
PaytooAccount	PaytooAccountType	PayToo Account informations associated to a request/transaction
PaytooRequest	PaytooRequestType	PayToo Request with full information
hash	string	Hash code to check the authenticity of the IPN response – not filled with the API

3.3.2 PaytooCreditCardType

Propertie	Type	Description
cc_id	int	Internal unique ID in the PayToo system for this credit card
cc_type	string	Type of credit card ('VISA', 'MC', 'AMEX')
cc_currency	string	Currency of the credit card (3 char)
cc_holder_name	string	Credit Card holder name
cc_number	string	Credit Card number
cc_month	string	Credit Card month of expiration (2 char)
cc_year	string	Credit Card year of expiration (4 char)
cc_cvv	string	Credit Card CVV2 (empty in response)
cc_track_1	string	Raw Magnetic Stripe Data (track 1)
cc_track_2	string	Raw Magnetic Stripe Data (track 2)

Propertie	Type	Description
cc_track_3	string	Raw Magnetic Stripe Data (track 3)
cc_default	int	Define if the card is the default one (0 = no, 1 = yes)
cc_status	string	Credit Card status - readonly
cc_status_infos	string	Credit Card status infos ('pending','accepted','rejected','locked','expired') - readonly
cc_creation	string	Credit Card creation date - readonly
cc_lastupdate	string	Credit Card last update - readonly

3.3.3 PaytooTransactionType

Propertie	Type	Description
tr_id	int	Internal unique ID in the PayToo system for this transaction
tr_type	string	Transaction type. Possible values are: "wallet2partner" or "partner2wallet"
tr_from_type	string	Transaction source type. Usualy "wallet"
tr_from_id	string	Transaction source id. Usualy the wallet number
tr_from_currency	string	The main currency of the source account
tr_to_type	string	Transaction destination type. Usualy "merchant"
tr_to_id	string	Transaction destination id.
tr_to_currency	string	Transaction destination currency.
tr_requested_original	float	Transaction original requested amount
tr_requested_currency	string	Transaction original requested currency
tr_amount_requested	float	Transaction converted requested amount (used currency is defined in $tr_from_currency)
tr_amount_transfered	float	Transaction converted transfered amount (used currency is defined in $tr_to_currency)
tr_amount_total_cost	float	Transaction total cost including fees (used currency is defined in $tr_from_currency)
tr_amount_refunded	float	Transaction refunded amount (used currency is defined in $tr_from_currency)
tr_change_rate	float	Transaction change rate from $tr_requested_currency and the $tr_from_currency if not the same
tr_fees	float	Transaction fees that generaly apply to the sender ($tr_from_id)
tr_fees_currency	string	Transaction fees currency (generaly the same as $tr_from_currency)
tr_fees_type	string	Type of fees ('fixed', 'percent', 'both')
tr_fees_rate_fixed	float	Fixed part of the fees
tr_fees_rate_percent	float	Variable part of the fees
tr_fees_level	int	User level applied to the fees and limits (only apply when 'from' is a paytoo wallet)
tr_date_created	string	Transaction creation date
tr_date_updated	string	Transaction last update
tr_date_completed	string	Transaction processing date

Propertie	Type	Description
tr_date_refunded	string	Transaction refund date
tr_notif_sender	string	Type of notification sent to the sender
tr_notif_receiver	string	Type of notification sent to the receiver
tr_status	string	Transaction status ('waiting', 'rejected', 'payed', 'refunded', 'loaded', 'confirmed', 'completed', 'cancelled')
tr_status_msg	string	Transaction status additional informations

3.3.4 PaytooAccountType

Properties	Type	Mandatory	Description
user_id	int		Internal unique ID in the PayToo system
wallet	string		PayToo Wallet number
currency	string		Wallet currency.
balance	string		Wallet balance in realtime. Depending of your permissions, this field can be blank.
registered_phone	string	Yes, if no email	PayToo registered phone number.
max_pin	string		PayToo Max PIN. PIN used for the cellphone registration (readonly).
sim_phonenumber	string		Paytoo SIM card phone number
prepaidcard	string		Prepaid card number or reference
email	string	Yes, if no cellphone	Email address – UNIQUE per customer.
password	string		Password.
gender	string		Gender: 'M' for Male / 'F' ofr Female.
firstname	string	Yes	First name
middlename	string		Middle name
lastname	string	Yes	Last name
address	string	Yes	Address
city	string	Yes	City
zipcode	string	Yes	Zip/Postal code
country	string	Yes	Country code (2 chars)
state	string	Yes, for US	State code (2 chars / for US citizen only)
phone	string		Phone number. Used for contact purpose only.
birthday	string		Birthday. Format: YYYY-MM-DD
security_code	string		Wallet Security code (6 digits).
question1	int		First security question (cf. documentation for possible values).
answer1	string		Answer for the first security question (max 60 chars).
question2	int		Second security question (cf. documentation for possible values).

Properties	Type	Mandatory	Description
answer2	string		Answer for the second security question (max 60 chars).
question3	int		Third security question (cf. documentation for possible values).
answer3	string		Answer for the third security question (max 60 chars).
citizenship	string		Citizenship country code (2 chars).
id_type	string		ID Type. Possible value: Drivers License, National ID Card, Passport, Social Security Card
id_issued_by_country	string		Country where the ID has been issued (iso2 code).
id_issued_by_state	string		State where the ID has been issued (iso2 code). Required when id_issued_by_country is United States
id_number	string		ID Number.
id_expiration	string		Date of expiration of the ID. Format: YYYY-MM-DD.
ssn_number	string		Social Security Number (US resident only)
dist_id	string		Distributor ID (read only)
res_id	string		Reseller ID (read only)
pos_id	string		Point of sales ID
document1	PaytooDocumentType		First identity document
document2	PaytooDocumentType		Second identity document
document3	PaytooDocumentType		Third identity document
custom_field1	string		Partner custom field (to store your ID/reference for example)
custom_field2	string		Partner custom field (to store additional infos)
custom_field3	string		Partner custom field (to store additional infos)
custom_field4	string		Partner custom field (to store additional infos)
custom_field5	string		Partner custom field (to store additional infos)
level	Integer		User wallet level (read only)

3.3.5 PaytooDocumentType

Propertie	Type	Description
id	int	Internal unique ID in the PayToo system
type	string	Type
name	string	File name (with extension)
size	string	File size
extension	string	File extension
file	string	base64 encoded file content

3.3.6 PaytooRequestType

Propertie	Type	Description
request_id	integer	Internal unique ID in the PayToo system for this request
tr_id	integer	Internal unique ID in the PayToo system for the associated transaction
user_id	integer	Internal unique ID of a customer in the PayToo system
ref_id	string	Your reference ID
sub_account_id	string	Sub Account ID
date	string	Date of the request (format: YYYY-MM-DD HH:MM:SS)
refund_date	string	Refund date of the request (format: YYYY-MM-DD HH:MM:SS)
expiration	string	Expiration date (format: YYYY-MM-DD HH:MM:SS)
statement_id	integer	Statement ID (if transaction has been settled to the merchant)
refund_statement_id	integer	Refund statement ID (if transaction has been refunded on a statement)
method	string	Request method (api, gateway, terminal)
type	string	Transaction type. Possible values are: "micro", "small", "regular", "creditcard"
is_pre_auth	boolean	True if the request has been initiated as a pre-auth transaction
is_recurring	boolean	True if the request has been initiated as a recurring transaction
is_a_cycle	boolean	True if the request is part of a recurring cycle
recurring_id	integer	Main/parent recurring request ID (filled if $is_a_cycle is true)
currency	string	Request currency
amount	float	Original requested amount or Initial amount for recurring request
description	string	Description
addinfo	string	Additional infos that you have provided
status	string	Request status
status_infos	string	Additional infos on the status
recurring_amount	float	Recurring amount
recurring_cycles	integer	Number of recurring cycles
recurring_period	string	Recurring period ('days', 'weeks', 'months', 'years')
recurring_start	string	Recurring payment start date (format: YYYY-MM-DD HH:MM:SS)
recurring_end	string	Recurring payment end date (format: YYYY-MM-DD HH:MM:SS)
recurring_status	string	Recurring status. Possible values are: "pending", "open", "stopped", "terminated"
recurring_info	string	Recurring additional info
transaction	PaytooTransactionType	Associated transaction - Only filled with function GetRequests()

3.3.7 PaytooRequestSearchCriterias

Propertie	Type	Description
request_id	integer	Request ID
tr_id	integer	Transaction ID
ref_id	string	Reference ID
sub_account_id	string	Sub Account ID
user_id	integer	User ID
statement_id	integer	Statement ID (if transaction has been settled to the merchant)
refund_statement_id	integer	Refund statement ID (if transaction has been refunded on a statement)
method	string	Request method (api, gateway, terminal)
type	string	Transaction type. Possible values are: "micro", "small", "regular", "creditcard"
is_pre_auth	boolean	Has the request been initiated as a pre-auth transaction ?
is_recurring	boolean	Has the request been initiated as a recurring transaction ?
recurring_req_id	integer	Search for the cycles/transactions associated to the provided recurring request id
recurring_status	string	Recurring status. Possible values are: "pending", "open", "stopped", "terminated"
is_a_cycle	boolean	Does the request is part of a recurring cycle ?
date_created	string	Transaction creation date. Format: YYYY-MM-DD HH:MM:SS
from_date	string	Look for requests created after the provided date. Format: YYYY-MM-DD HH:MM:SS
to_date	string	Look for requests created before the provided date. Format: YYYY-MM-DD HH:MM:SS
date_refunded	string	Transaction refund date. Format: YYYY-MM-DD HH:MM:SS
refund_from_date	string	Look for requests refunded after the provided date. Format: YYYY-MM-DD HH:MM:SS
refund_to_date	string	Look for requests refunded before the provided date. Format: YYYY-MM-DD HH:MM:SS
amount	float	Transaction Amount
amount_comparator	string	Amount Comparator. Possible values: =,>,<,<=,=> (default is =)
status	string	Request status, it can be 'pending', 'accepted', 'rejected', 'cancelled', 'completed', 'refunded'
tr_type	string	Transaction type. Possible values are: "wallet2partner", "wallet2merchant", "creditcard2merchant", "creditcard2partner", "partner2wallet"
tr_from_type	string	Transaction source type. Possible values are: "wallet", "creditcard"
tr_status	string	Transaction status ('waiting', 'rejected', 'payed', 'refunded', 'loaded', 'confirmed', 'completed', 'cancelled')
customer_firstname	string	Customer first name
customer_lastname	string	Customer last name
customer_email	string	Customer Email
pay_auth_code	string	Payment processor auth code
cc_number	string	Credit Card Number (for partial number, placeholder is *)

3.3.8 PaytooBankAccountType

Propertie	Type	Description
bank_id	int	Internal unique ID in the PayToo system for this bank account
bank_type	string	Bank account type (Checking or Saving)
bank_currency	string	Currency of the credit card (3 char)
bank_name	string	Bank name
bank_country	string	Bank country
bank_state	string	Bank state
bank_location	string	Bank city
bank_address	string	Bank address
Bank_zip	string	Bank zip/postal code
bank_phone	string	Bank phone number
bank_code	string	Bank code
bank_account	string	Bank account number or IBAN
bank_swift	string	Bank swift
bank_routing	string	Bank routing number
bank_status	string	Bank account status - readonly
bank_status_infos	string	Bank account status infos ('pending','accepted','rejected','locked') - readonly
bank_creation	string	Bank account creation date - readonly
bank_lastupdate	string	Bank account last update - readonly

4 Instant Payment Notification (IPN)

Instant Payment Notification (IPN) is a message service that notifies you of events related to PayToo transactions. You can use it to automate back-office and administrative functions, such as fulfilling orders, tracking customers, and providing status and other information related to a transaction.

What is IPN?

IPN notifies you when an event occurs that affects a transaction. Typically, these events represent various kinds of payments; however, the events may also represent authorizations, Fraud Management Filter actions and other actions, such as refunds, disputes, and chargebacks.

IPN is a message service that PayToo uses to notify you about events. These events include the following:

- Payments, including Micro Payment and recurring or indirect payments
- Refunds
- Rejected request/transaction

In many cases, the action that causes the event, such as a payment, occurs on your website; however, your website is not the only source of events. In many cases, events can be generated by the PayToo Gateway, the PayToo API, by Paytoo agents or by the customer itself.

You detect and process IPN messages with a *listener*, sometimes called a handler, which is a script or program that you write. It waits for messages and passes them to various back-end or administrative processes that respond the messages.

The actions to take when your listener is notified of an event are specific to your needs. Examples of the kinds of actions you might take when your listener receives an IPN message include the following:

- Trigger order fulfillment or enable media downloads when a check clears or a payment is made
- Update your list of customers
- Update accounting records
- Create specialized “to do” lists based on the kind of event

You are typically notified of events by email as well, but the IPN message service enables you to automate your response to events.

IPN Message

An IPN message consists of variables that describe the transaction. These variables contain information about you, your customer, and the details of the transaction itself.

To check the authenticity of the message, we include a checksum control in the message, also called a "hash". To calculate this checksum / hash, please refer to the next paragraph.

Prior to API 0.7, the IPN message contained only few parameters. These parameters have been kept for backward compatibility. You will find below the both IPN format (both are automatically sent).

Former IPN parameters

These parameters are sent directly in the POST (for example in $_POST using php).

Variable	Description	Example
request_id	Request ID is the unique ID of the request	*1234*
tr_id	Transaction ID is the unique ID of transaction associated to the request	*1234*
order_ref	Reference ID is the unique ID you've passed for this request	*1234*
w_number	The customer wallet number	*12345678*
phone_number	The customer registered phone number or simcard phone number	*31620512335*
tr_status	The transaction status (*cancelled, rejected, pending, payed, completed*)	*payed*
tr_amount	The final transaction amount = the amount you've received	*10.00*
tr_currency	The final transaction currency	*EUR*
tr_requested_amount	The original transaction amount you've requested	*12.50*
tr_requested_currency	The origianl transaction currency you've requested	*USD*
tr_change_rate	The change rate is applicable	*1.25*
hash	The checksum control of the message	*jklmwxcvbnertg...*

New IPN parameters

The new IPN parameters are sent in a POST variable/array named 'MerchantApiResponse' (for example $_POST['MerchantApiResponse'] using PHP).

This variable/array contains exactly the same parameters as the ***MerchantApiResponse*** type the described in the paragraph 3.3.1, so it will contain the full request, transaction and user infos.

We recommend to use this new IPN as it contain more information about your request, the transaction and the customer.

Checksum calculation

In order to calculate the checksum of message and to compare your result with the given hash, you must first define a "key" into your admin panel. If this key is not defined, you will not receive this hash.

For hash present to the former IPN *($_POST['hash'] using PHP)*

The hash can be found by concatenating all the variables in the same order than in this previous table.

For example, in PHP, you will use the following code:

```
$string = '';

$key = 'yourprivatekey-notyourpassword!';

$fields = array ('request_id', 'tr_id', 'order_ref',
'w_number', 'phone_number', 'tr_status', 'tr_amount', 'tr_currency',
'tr_requested_amount', 'tr_requested_currency', 'tr_change_rate');

foreach($fields as $k => $v) $string .= $_REQUEST[$v];

$string .= $key;

$hash = md5($string);

if ($hash == $_REQUEST['hash']) echo "The request is valid";
```

For the new hash present in the MerchantApiResponse
($_POST['MerchantApiResponse']['hash'] using PHP)

The calculation of this hash has been simplified, it require only 5 parameters.
The hash can be found by concatenating the following variables (in the given order and without any other characters or spaces) :

- request_id
- amount
- currency
- status
- your private key

For example, in PHP, you will use the following code:

```
$mykey = 'yourprivatekey-notyourpassword!';

$request = $_POST['MerchantApiResponse']['PaytooRequest'];

$myhash = md5($request['request_id']. $request['amount']. $request['currency'].
$request['status'].$mykey);

if ($myhash == $request['hash']) echo "The request is valid";
```

5 Code Samples

5.1 PHP Sample

```
<?php
class PaytooAccountType {
   var $user_id = null;
   var $wallet = null;
   var $currency = null;
   var $balance = null;
   var $registered_phone = null;
   var $sim_phonenumber = null;
   var $prepaidcard = null;
   var $email = null;
   var $password = null;
   var $gender = null;
   var $firstname = null;
   var $middlename = null;
   var $lastname = null;
   var $address = null;
   var $city = null;
   var $zipcode = null;
   var $country = null;
   var $state = null;
   var $phone = null;
   var $birthday = null;
   var $security_code = null;
   var $question1 = null;
   var $answer1 = null;
   var $question2 = null;
   var $answer2 = null;
   var $question3 = null;
   var $answer3 = null;
   var $citizenship = null;
   var $id_type = null;
   var $id_issued_by_country = null;
   var $id_issued_by_state = null;
   var $id_number = null;
   var $id_expiration = null;
   var $ssn_number = null;
   var $max_pin = null;
   var $dist_id = null;
   var $res_id = null;
   var $pos_id = null;
   var $document1 = null;
   var $document2 = null;
   var $document3 = null;
   var $custom_field1 = null;
   var $custom_field2 = null;
   var $custom_field3 = null;
   var $custom_field4 = null;
   var $custom_field5 = null;
   var $level = null;
}
```

```
class PaytooCreditCardType {
   var $cc_id = null;
   var $cc_type = null;
   var $cc_currency = null;
   var $cc_holder_name = null;
   var $cc_number = null;
   var $cc_month = null;
   var $cc_year = null;
   var $cc_cvv = null;
   var $cc_default = null;
   var $cc_status = null;
   var $cc_status_infos = null;
   var $cc_creation = null;
   var $cc_lastupdate = null;
   var $cc_track_1 = null;
   var $cc_track_2 = null;
   var $cc_track_3 = null;
}

try {
   ini_set('soap.wsdl_cache_enabled', 0);
   $soap = new SoapClient("https://go.paytoo.info/api/merchant/?wsdl",
array("classmap"=>array("PaytooAccountType"=>"PaytooAccountType",
"PaytooCreditCardType"=>"PaytooCreditCardType")));
   $merchant_id = '00000000';
   $api_password = 'APIPASSWORD';
   // Authentification
   $response = $soap->auth($merchant_id, $api_password);
   if($response->status=='OK') {
      echo "Connected\n";

      $CreditCard = new PaytooCreditCardType ();
      $CreditCard->cc_type = "VISA"; // mandatory
      $CreditCard->cc_holder_name = "DEMO USER"; // mandatory
      $CreditCard->cc_number = "4444333322221111"; // mandatory
      $CreditCard->cc_cvv = "123"; // mandatory
      $CreditCard->cc_month = "12"; // mandatory
      $CreditCard->cc_year = "14"; // mandatory
      $Customer = new PaytooAccountType ();
      $Customer->email = "demo@paytoo.com "; // mandatory
      $Customer->firstname = "Demo"; // mandatory
      $Customer->lastname = "User"; // mandatory
      $Customer->address = "200 SW 1st Avenue";
      $Customer->city = "Fort Lauderdale";
      $Customer->zipcode = "33301";
      $Customer->state = "FL";
      $Customer->country = "US";
      $amount = 6.00; // mandatory
      $currency = 'USD'; // mandatory

      echo "Processing Credit Card Sale\n";
      $ref_id = rand(1000, 9999); // mandatory
      $description = "Order #".$ref_id." with Paytoo Merchant";
      $addinfo = "";
```

```
    $response = $soap->CreditCardSingleTransaction ( $CreditCard, $Customer, $amount, $currency, $ref_id,
$description );
    if ($response->status == 'OK') {
      echo "Transaction has been processed\n";
      echo "Request ID: " . $response->request_id . "\n";
      echo "Tr. ID: " . $response->tr_id . "\n";
    } else {
       echo $response->status . " - " . $response->msg . "\n";
    }

    echo "Processing Credit Card Pre-Auth\n";
    $ref_id = rand(1000, 9999); // mandatory
    $description = "Order #".$ref_id." with Paytoo Merchant";
    $addinfo = "";
    $response = $soap->CreditCardPreAuth ( $CreditCard, $Customer, $amount, $currency, $ref_id, $description
);
    if ($response->status == 'OK' && $response->request_status == 'accepted') {
      echo "Request has been accepted\n";
      echo "Request ID: " . $response->request_id . "\n";
      $request_id = $response->request_id;
      echo "Capture the request ".$request_id."\n";
      $response = $soap->Settlement($request_id, $amount-2); // Different amount
      if ($response->status=='OK') {
        echo "Transaction has been captured\n";
        echo "Tr. ID: " . $response->tr_id . "\n";
        echo "Initial requested amount: ".$response->PaytooRequest->amount."\n";
        echo "Final processed amount: ".$response->PaytooTransaction->tr_amount_transfered."\n";
      } else {
        echo $response->status." - ".$response->msg."\n";
      }
    } else {
       echo $response->status . " - " . $response->msg . "\n";
    }

    echo "Processing Wallet Sale\n";
    $response = $soap->SingleTransaction('08587726', '123456', 13, 'USD', '1234', 'Order 1234');
    if ($response->status == 'PENDING') {
      echo "Step 2a: Transaction accepted but it must be confirmed\n";
      echo "Request ID: " . $response->request_id . "\n";
      $request_id = $response->request_id;
      // OTP is always 888888 on sandbox
      $response2 = $soap->ConfirmTransaction($request_id, '888888');
      if ($response2->status == 'OK') {
        echo "Finish: transaction has been processed\n";
        echo "Tr. ID: " . $response2->tr_id . "\n";
      } else {
        echo "ConfirmTransaction error: ".$response2->status . " - " . $response2->msg . "\n";
      }
    } elseif ($response->status == 'OK') {
      echo "Finish: transaction has been processed\n";
      echo "Tr. ID: " . $response->tr_id . "\n";
    } else {
      echo "SingleTransaction error: ".$response->status . " - " . $response->msg . "\n";
    }
```

```
    echo "Processing Wallet Pre-Auth\n";
    $response = $soap->PreAuth('08587726', '123456', 20, 'USD', '1234', 'Order 1234');
    if ($response->status == 'PENDING') {
      echo "Step 2a: Transaction accepted but it must be confirmed\n";
      echo "Request ID: " . $response->request_id . "\n";
      $request_id = $response->request_id;
      // OTP is always 888888 on sandbox
      $response2 = $soap->ConfirmTransaction($request_id, '888888');
      if ($response2->status == 'OK') {
        echo "Step 3a: Transaction has been confirmed, it can be processed\n";
         $response3 = $soap->Settlement($request_id);
        if ($response3->status == 'OK') {
          echo "Finish: transaction has been processed\n";
          echo "Tr. ID: " . $response3->tr_id . "\n";
        } else {
          echo "Settlement error: ".$response3->status . " - " . $response3->msg . "\n";
        }
      } else {
        echo "ConfirmTransaction error: ".$response2->status . " - " . $response2->msg . "\n";
      }
    } elseif ($response->status == 'OK') {
      echo "Step 2b: Transaction has been accepted, no confirmation required, it can be processed\n";
      $request_id = $response->request_id;
      $response2 = $soap->Settlement($request_id);
      if ($response2->status == 'OK') {
        echo "Finish: transaction has been processed\n";
        echo "Tr. ID: " . $response2->tr_id . "\n";
      } else {
        echo "Settlement error: ".$response2->status . " - " . $response2->msg . "\n";
      }
    } else {
       echo "PreAuth error: ".$response->status . " - " . $response->msg . "\n";
    }

    $soap->logout();
    echo "Logout\n";
  } else {
    echo "Auth error: ".$response->status . " - " . $response->msg . "\n";
  }
} catch (Exception $e) {
  var_export($e);
}

?>
```

5.2 C# Sample

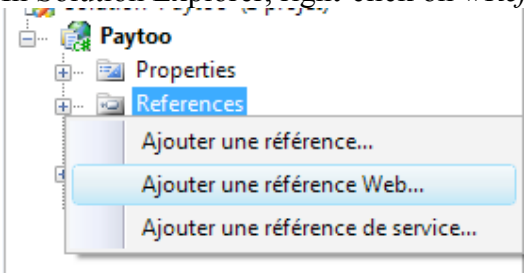
To simplify development of XML Web service client applications, Visual Studio provides Web references. Web references differ from traditional references and components; instead of referencing a component or a class library installed on the local computer, a Web reference provides access to a resource that is available using an Internet protocol such as SOAP or HTTP.

In practice, a Web reference is a generated proxy class that locally represents the exposed functionality of an XML Web service. The proxy class defines methods that represent the actual methods exposed by an XML Web service. When your application creates an instance of the proxy class, your application can call the XML Web service methods as if the XML Web service were a locally available component.

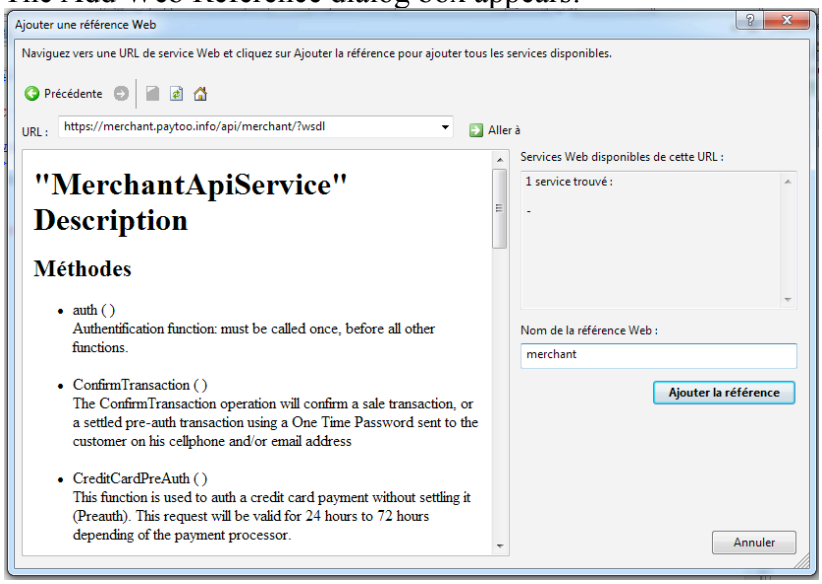
How to add the Web Reference

The Add Web Reference dialog box enables you to browse for Web services and add Web references to your Web site. A Web reference enables you to use objects and methods provided by a Web service in your code.
To open the Add Web Reference dialog box

- In Solution Explorer, right-click on « *References* » and then click **Add Web Reference**:

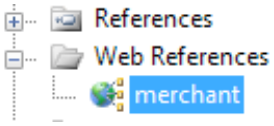


- The Add Web Reference dialog box appears:



- Type the URL of the web service :
  - https://go.paytoo.info/api/merchant/?wsdl for the sandbox
  - https://go.paytoo.com/api/merchant/?wsdl for the production

- Click on the « Go » button, you should see a description of each methods of the API
- If you can, you can rename the reference in « Web reference name »
- And then, click on « Add Reference »
- The new reference will appear in « Solution Explorer »



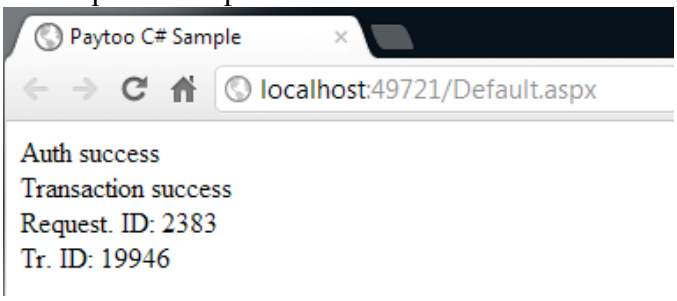
After adding a Web reference to your current project, you can use any element or functionality provided by that Web service within your application. Look at the code below as an example.

Code sample

In the following snippet, we assume that your project is named « Paytoo » and that your web reference is named « merchant ».

```
Paytoo.merchant.MerchantApiService proxy = new Paytoo.merchant.MerchantApiService();
proxy.CookieContainer = new System.Net.CookieContainer();
Paytoo.merchant.MerchantApiResponse authResponse;
authResponse = proxy.auth("12345678", "demo12", "");
if (authResponse.status == "OK") {
   Response.Write("Auth success");
   Paytoo.merchant.PaytooCreditCardType creditcard = new Paytoo.merchant.PaytooCreditCardType();
   creditcard.cc_holder_name = "PAYTOO DEMO";
   creditcard.cc_type = "VISA";
   creditcard.cc_number = "4444333322221111";
   creditcard.cc_cvv = "123";
   creditcard.cc_month = "12";
   creditcard.cc_year = "14";
   Paytoo.merchant.PaytooAccountType customer = new Paytoo.merchant.PaytooAccountType();
   customer.email = "demo@paytoo.com";
   customer.firstname = "Demo";
   customer.lastname = "Paytoo";
   customer.address = "200 SW 1st Avenue";
   customer.zipcode = "33301";
   customer.city = "Fort Lauderdale";
   customer.state = "FL";
   customer.country = "US";
   customer.registered_phone = "19546369200";
   Paytoo.merchant.MerchantApiResponse txResponse;
   txResponse = proxy.CreditCardSingleTransaction(creditcard, customer, 5, "USD", "1234", "Order 1234", "");
   if (txResponse.status == "OK") {
      Response.Write("<br/>Transaction success");
      Response.Write("<br/>Request. ID: " + txResponse.PaytooRequest.request_id);
      Response.Write("<br/>Tr. ID: " + txResponse.PaytooTransaction.tr_id);
   } else {
      Response.Write("<br/>Tr. error: " + txResponse.msg);
   }
} else {
   Response.Write("Auth error: " + authResponse.msg);
}
```

The expected output of this code should look like this:



For a Pre-Auth, the code/process will look like this:

```
Paytoo.merchant.MerchantApiResponse preauthResponse;
preauthResponse = proxy.CreditCardPreAuth(creditcard, customer, 5, "USD", "1234", "Order 1234", "");
if (preauthResponse.status == "OK") {
   Response.Write("<br/>The Pre-Auth request has been accepted. Funds have been reserved.");
   Response.Write("<br/>Request. ID: " + preauthResponse.PaytooRequest.request_id);
   Paytoo.merchant.MerchantApiResponse txResponse;
   txResponse = proxy.Settlement(preauthResponse.PaytooRequest.request_id);
   if (txResponse.status == "OK") {
      Response.Write("<br/>The transaction has been completed. The funds have been captured.");
      Response.Write("<br/>Tr. ID: " + txResponse.PaytooTransaction.tr_id);
      Response.Write("<br/>Tr. amount: " + txResponse.PaytooTransaction.tr_amount_transfered);
   }
} else {
   Response.Write("<br/>Tr. error: " + txResponse.msg);
}
```

Important remark about web service session using C#

The XML Web service method uses session state for the authentication, then a cookie is passed back in the response headers to the XML Web service client that uniquely identifies the session for that XML Web service client. In order for an XML Web service to maintain session state for a client, the client must store the cookie. Clients receive the HTTP cookie by creating a new instance of CookieContainer and assigning that to the CookieContainer property of the proxy class before calling the XML Web service method. In order to cummincate with the Yackie Web Service, you must maintain session state beyond when the proxy class instance goes out of scope, the client must store the HTTP cookie between calls to the XML Web service. For instance, a Web Forms client can store the HTTP cookie by saving the CookieContainer in its own session state, like in the above example.

5.3 Java

In order to consume SOAP Web Service with Java, PayToo recommand the use of the Axis Web Services tools. This quick tutorial will show you how to integrate the PayToo Partner API using Eclipse WTP and Axis.

Set Up

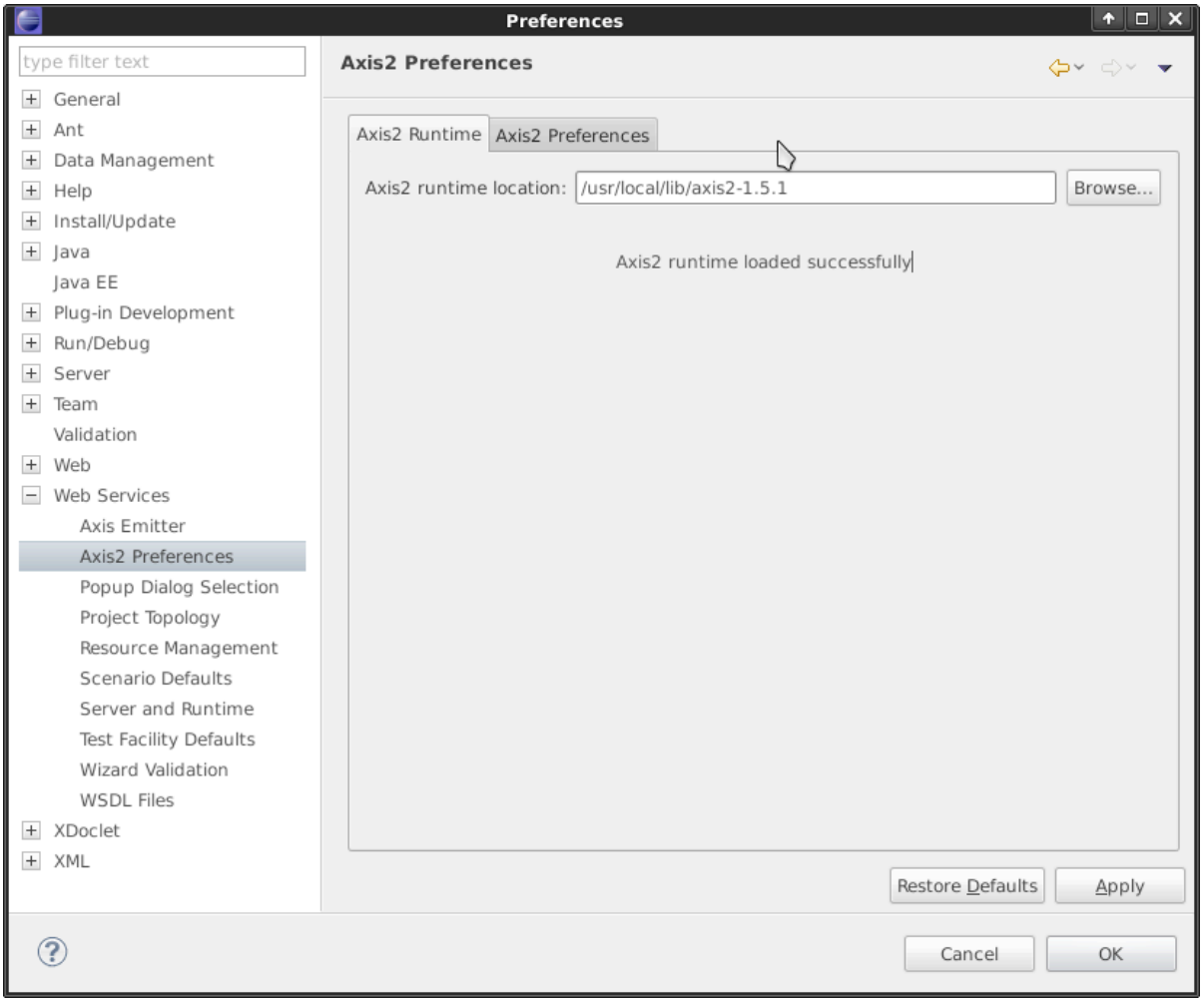
Before creating the Web service, there are two prerequisites:

Eclipse WTP 2.0 (you can use also Netbean but this tutorial is designed for Eclipse).
Configure Apache Tomcat inside Eclipse WTP

Creating a bottom up JAVA Web service client using Axis WTP Tools

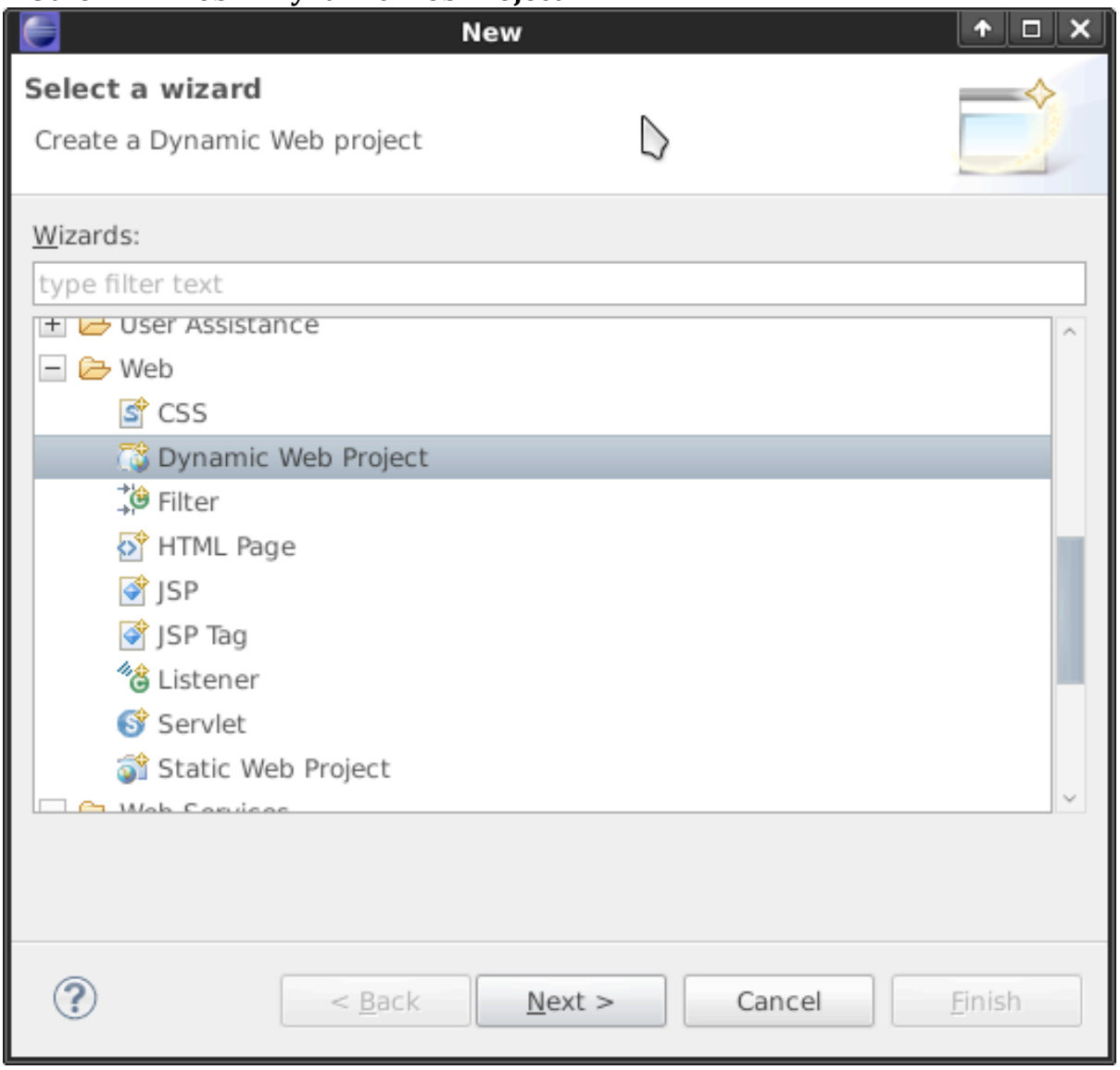
This tutorial need a Axis runtime. You can download the latest axis binary distribution from here.

- Download the latest Axis runtime from the above link and extract it.
- Now we point Eclipse WTP to downloaded Axis Runtime. Open **Window -> Preferences -> Web Services -> Axis2 Preferences**

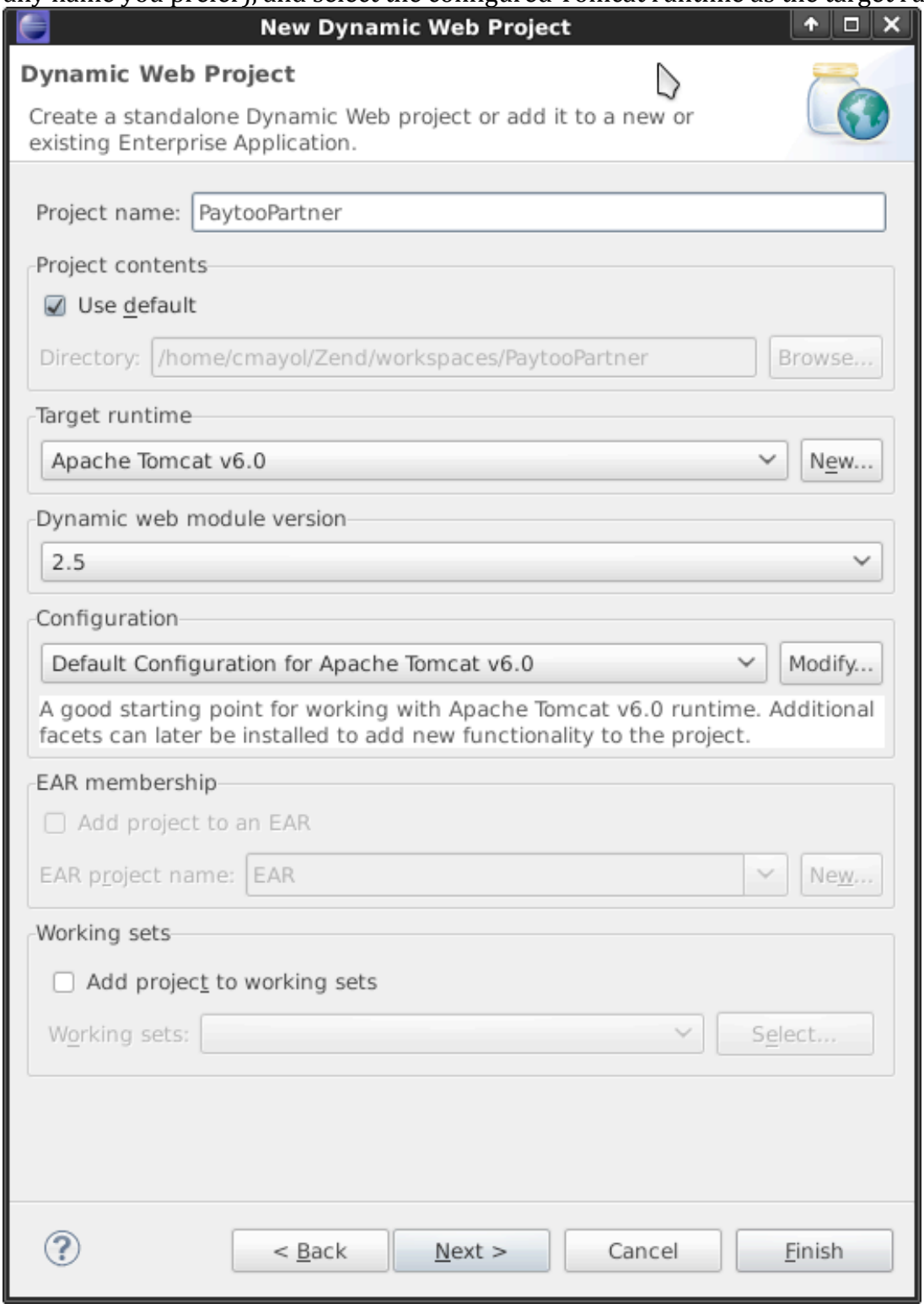


Select the Axis2 Runtime tab and point to the correct Axis runtime location. Alternatively at the Axis2 Preference tab, you can set the default setting that will come up on the Web Services Creation wizards. For the moment we will accept the default settings.

- Click OK
- Next we need to create a project with the support of Axis features. Open **File -> New -> Other... -> Web -> Dynamic Web Project**

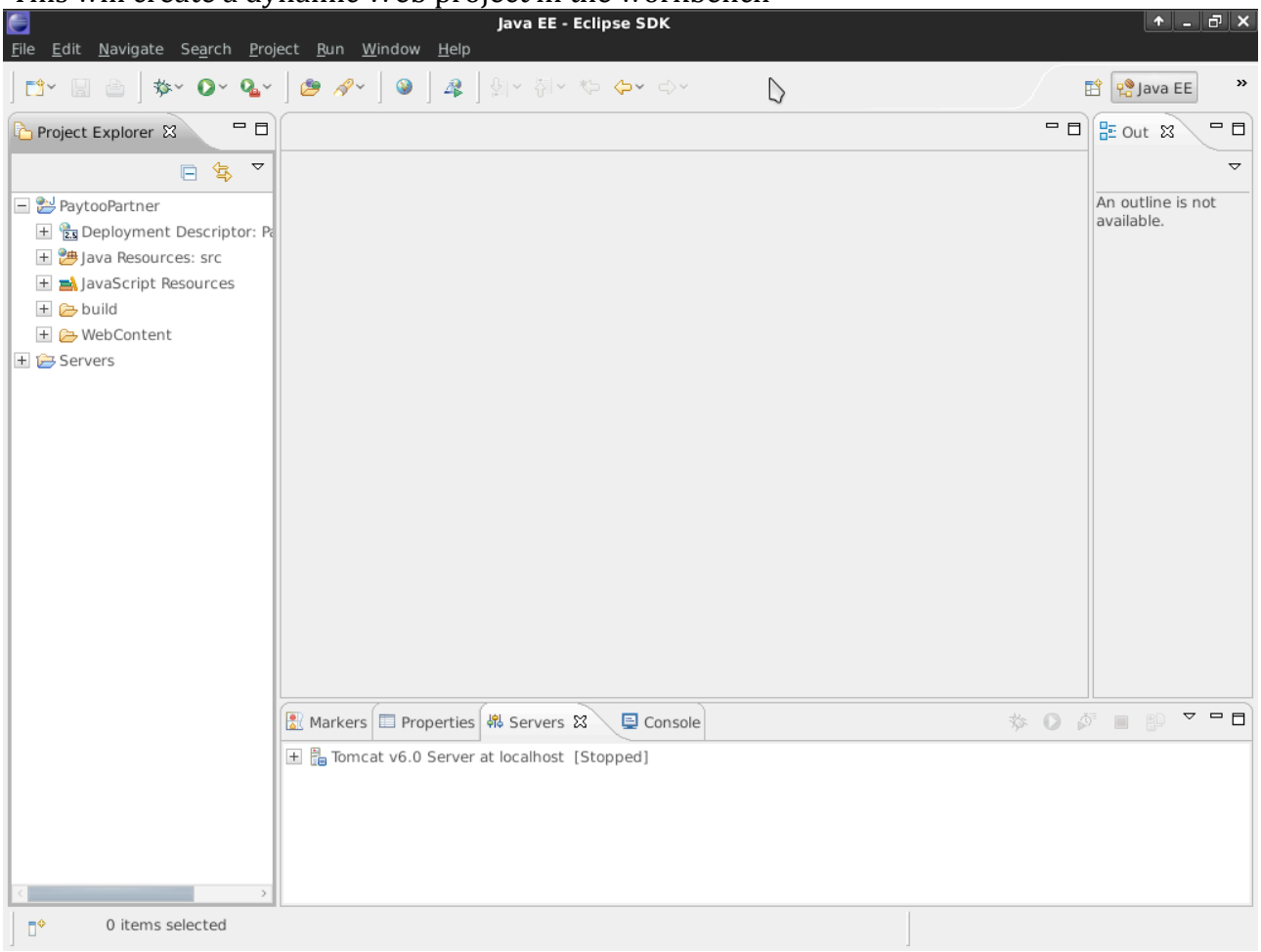


- Select the name **PaytooPartner** as the Dynamic Web project name (you can specify any name you prefer), and select the configured Tomcat runtime as the target runtime.

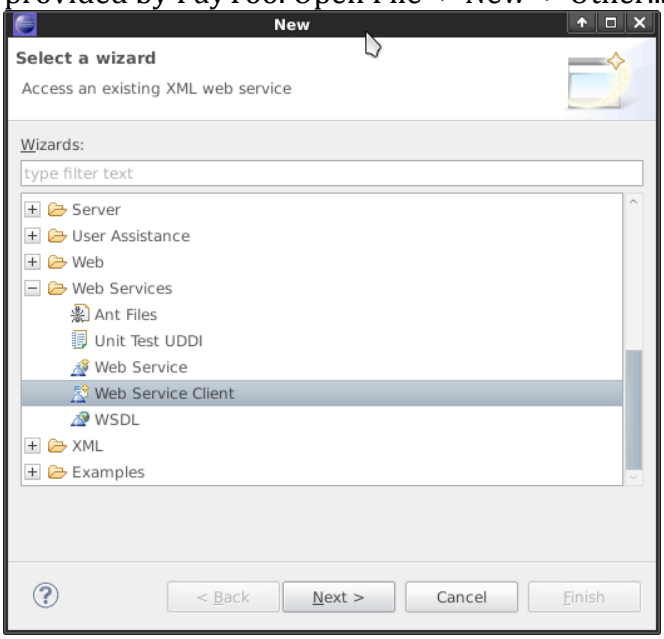


Click Finish.

- This will create a dynamic Web project in the workbench



- Now we'll generate the client for the PayToo Partner service by referring the ?wsdl provided by PayToo. Open File -> New -> Other... -> Web Services -> Web ServiceClient

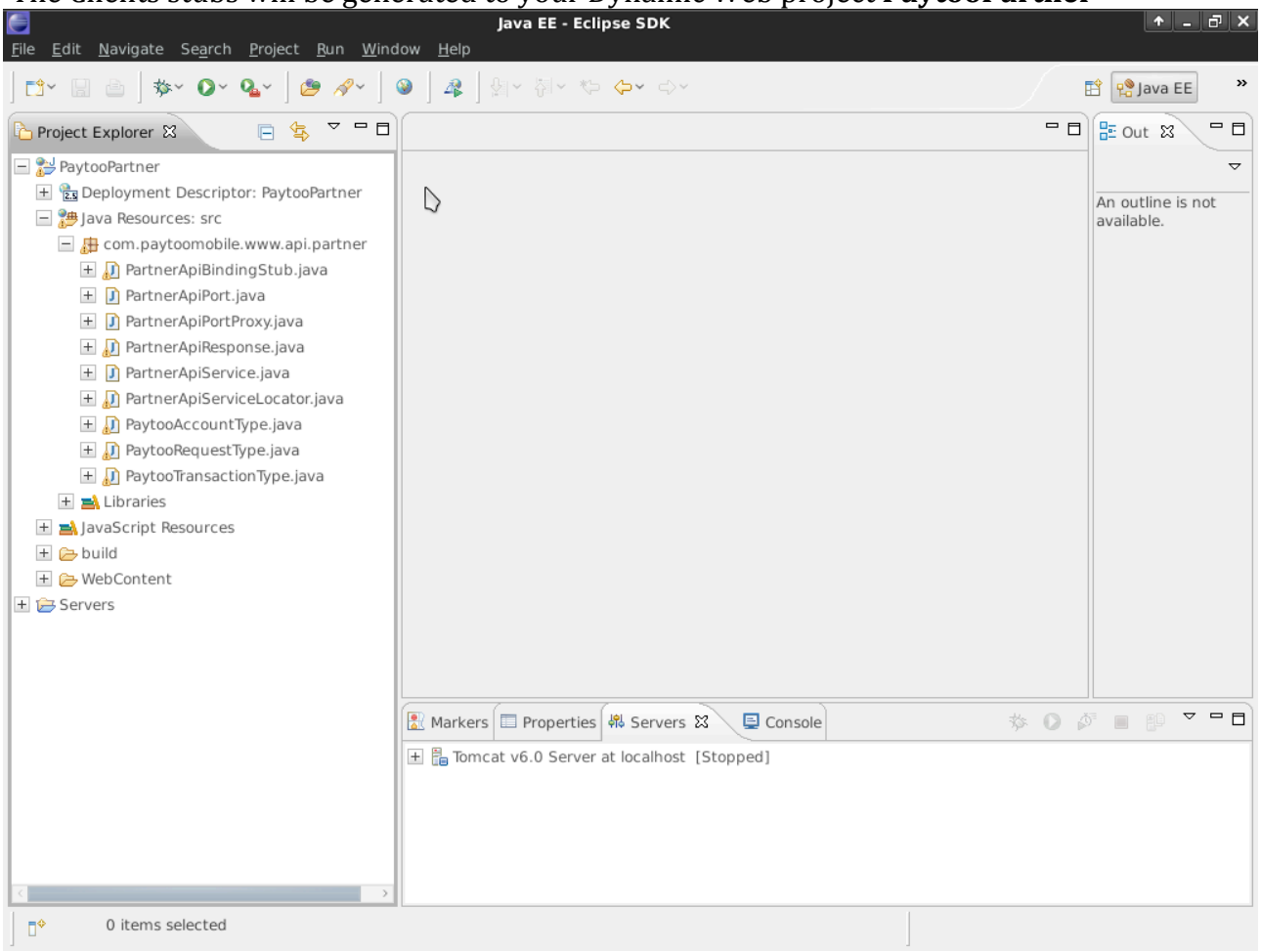


Click Next

- Paste the PayToo Partner WSDL URL, make sure the Web service runtime is set to Apache Axis and the server is set correctly.

(please note that the URL in this screenshot is not accurate)
Click Finish

- The Clients stubs will be generated to your Dynamic Web project **PaytooPartner**



- Now we are going to write Java main program to invoke the client stub. Right click on the newly created package and create a new Class
- Enter a name for this Class, for example « Client » and click Finish.
- The Class is automaticly openend into the editor. Paste the sample code you will find bellow and save the file. Of course, you will have to change your username and password according to your account informations.
- Then select the Client.java file, right-click and select Run As -> Java Application.

Here's what you get on the server console:

Sample code

```
package com.paytoo.go.api.merchant;

import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException; import org.apache.axis.AxisFault;

public class Client {

public static void main(String[] args) {

try {

URL ws;
ws = new URL("https://go.paytoo.com/api/merchant/");
String ok = new String("OK");
PartnerApiBindingStub stub = new PartnerApiBindingStub(ws, null);
stub._setProperty(org.apache.axis.client.Stub.SESSION_MAINTAIN_PROPERTY,
Boolean.TRUE);

PartnerApiResponse response = stub.auth("12345678", "password");
String status = response.getStatus();

if (status.equals(ok)) {
  System.out.println("Connection OK");
  boolean ispaytoo = stub.isPaytooAccount("31637008413");
  if (ispaytoo) {
    System.out.println("It is a PayToo account !");
  } else {
    System.out.println("It is NOT a PayToo account !!");
  }
} else {
  System.out.println("Authentification failed: " + status + " - " + response.getMsg
}

} catch (MalformedURLException e) {
  e.printStackTrace();
} catch (AxisFault e) {
  e.printStackTrace();
} catch (RemoteException e) {
  e.printStackTrace();
}

}
```

5 Mandatory fields and API restrictions

5.1 Mandatory fields for customer account

Due to regulation laws, the merchant must provide the customer's first name, last name and full address (city, zipcode, country and state for US customer).

Please also note that you must provide either a unique cell phone or a unique email address for each customer.

5.2 Restrictions

- PO Box are not allowed. Transaction created with such address will be rejected.
- Update of PayToo account is not allowed. So if a PayToo account match one of your customer, information like the address will not be updated.
- No more than 3 transactions of the same kind in one day, and 5 in one week, 10 in the month.

6 Third parties modules

Lime Light CRM

Full documentation on how to configure the PayToo Gateway, click on the link below:

http://help.limelightcrm.com/entries/21432858-configuring-paytoo-gateway

Magento

OFFICIAL extension from PayToo Corp.

Utilizes Go.PayToo gateway, API and instant notifications to provide you with the ability to do all of your sales management from Magento.

It has been extensivly tested, is used by many Go.PayToo merchants and is updated with each Magento community release.

http://www.magentocommerce.com/magento-connect/catalog/product/view/id/17243/s/paytoo-payment-gateway-9250/?___store=mc_default_store

7 Administration

The administrative merchant front end of PayToo is located at:
http://go.paytoo.com/
(or http://go.paytoo.info for the sandbox).

Just sign in with the merchant details you have obtained from us.

The main features are:

- Transaction monitoring
- Refunding transactions
- Viewing transaction details
- Reviewing reasons for declined transactions
- Manual processing via virtual terminal

8 Disclaimer

PayToo and the authors assume no liability for errors or omissions, or for damages, resulting from the use of this guide or the information contained in this guide.