

# PayCircle 1.0 Specification Payment Web Service – Concepts

PayCircle-ACA-001-002

**Draft**



PayCircle

*Standards that get  
m-commerce flowing*

## General

### Important Note

This document is work in progress.

PayCircle considers the XML sources (PayCircle-ACA-010-\*\*\*) as relatively mature and is currently working on supplementary documentation. The current document has been assembled to help better understand the XML sources, but is in no means an exhaustive documentation of the payment web service.

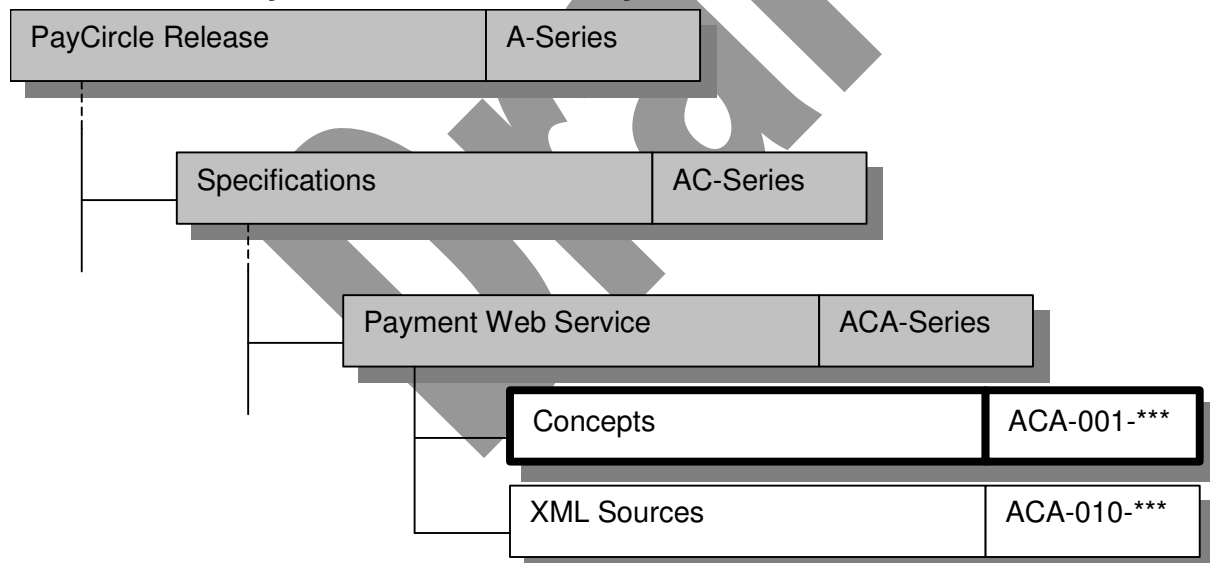
### Document History

Issue	Date	Remarks
001	27 Aug 2002	initial draft (kl)
002	12 Sep 2002	Draft for member review
003		

### Scope

This document is part of the Payment Web Service specification. It describes the underlying concepts. It is accomplished by the XML sources (PayCircle-ACA-010-\*\*\*).

### Position in the PayCircle Document Library



## Payment Models

The merchant can use two different payment models: Immediate payment or reservation and deferred payment.

**Immediate payment**, as the name suggests, does everything in a single rush. The request engine sends a single request to the payment engine. The request must contain all information that is necessary complete the payment transaction. The payment engine tries to complete the payment transaction immediately.

**Reservation and deferred payment** implements a more sophisticated lifecycle for a payment transaction. A transaction context is created explicitly, and then the request engine can control payment transaction by a sequence of reserve, debit and credit operations. Additional operations allow to obtain the status of the transaction context. Closing the transaction context terminates the lifecycle of the transaction.

## Immediate Payment

### Direct Debit

The request engine invokes this operation to initiate a payment. The account of the customer specified in the request will be debited, while the account of the merchant specified in the request will be credited.

### Direct Credit

The request engine invokes this operation to initiate a refund. The account of the merchant will be debited and the account of the customer will be credited.

## Reservation and deferred payment

### Create Session

This operation will create a transaction context for a payment transaction. At the time the transaction context is created, the merchant account, the customer account involved in the transaction are specified. They will not change over the transaction context's lifecycle.

### Reserve

This operation will reserve money from the customer's account that is specified in the transaction context. To reserve money, the payment engine shall debit the specified amount from the customer's account and credit it to an auxiliary account.

The reserve operation may be repeated at any time to top-up the existing reservation.

### Debit

The request engine invokes this operation to request a payment from the reserved money. The operation will only proceed if enough money has been reserved and not yet been debited.

The debit operation may debit all or only part of the amount that has been reserved earlier. The operation may be repeated.

However this operation does not result in any debit or credit. It schedules the requested amount for a later payment.

### Credit

This operation is the reverse to the Debit operation.

### GetAmountLeft

This operation returns the amount of money that has been reserved within the transaction context and not yet been debited.

### GetLifetimeLeft

This operation returns the remaining lifetime of the transaction context.

## Mapping

The following table shows which elements in the WSDL source code implement which of the concepts introduced above.

Term in Model	Element in WSDL Sources
Immediate Payment	portType "payment" in payment.wsdl
Direct debit	operation "debitAmount"
Direct credit	operation "creditAmount"
Reservation and deferred payment	portType "amountSession" in amountSession.wsdl
Create reservation	operation "createAmountSession"
Reserve	operation "reserve"
Debit	operation "debit"
Credit	n/a
Get Amount Left	operation "getAmountLeft"
Get Lifetime Left	operation "getLifeTimeLeft"

## Additional Concepts

### Security

Remark: A security concept still needs to be worked out. The following elaboration reflects the current working hypothesis

It is well recognized that a payment web service needs to be very careful about security. Issues that the web service needs to address are:

- Privacy
- Integrity
- Reliability

There is a variety of possibilities to achieve privacy and integrity, such as IP tunneling, private leased lines, HTTPS, digital signatures, etc. To ensure that the payment web service specification can be applied to the widest range of possible infrastructures, no security concepts are merged into the specification itself. Ensuring privacy and integrity is seen as a task of the application servers where the web service and the client are running, and needs to be negotiated between the respective owners (merchant and payment service provider). However, this specification gives recommendations which security instruments a payment web service implementation shall provide as a minimum.

Reliability is generally considered a feature of a particular vendor implementation. However, the payment web service allows switchover between payment web service instances.

### Privacy

Privacy is expected to be achieved by using HTTPS.

### Integrity

Integrity is expected to be achieved by enclosing digital signatures in HTTP headers. At the same time, the digital signature allows to authenticate the sender of a message.

### Identification of actors

In several operations of the payment web service, actors need to be identified (customers who shall pay for a service, or merchants who shall receive a payment). There are different ways of identifying actors: By their mobile phone number, by the IP address they use, or by an identity token that has been issued by an identity provider, etc.

The payment web service specification shall be prepared to support any type of identification for actors. Therefore, actors are specified as URIs. All parameters that are used to identify actors have the XML-type anyURI. The parameters are then formatted as follows:

In case the actor is identified by a telephone number, the number is given in the "tel:" URI scheme according to RFC 2806.

In case the actor is identified by an email address, the "mailto:" URI scheme is used as described in RFC 2368.

Formatting for other identification schemes are for further study. The use of the "urn:" URI scheme shall be considered for names. For identifications issued by identity providers (according to passport or liberty), registration of proper URI schemes by the respective specification bodies is expected and shall be used in the payment web services as well.

Remark: The current draft uses the type string instead of anyURI, since it has been syntax-checked with Axis beta2. Axis beta 2 does not yet support anyURI. Axis beta 3 does support anyURI, and the final of the WSDL sources will use anyURI.

## Request Numbering

All operation that change accounts have a parameter called "requestNumber". The request number serves the following purpose:

- It allows the web service client to correlate output or fault messages with the input message that caused them.
- It allows the web service client to re-send messages safely. The web server can determine if a message has received more than once; the duplicate message will be discarded. This is important in case of failures: When no response is received from the server, the client cannot determine if the server has processed the payment request, but failed to send a response, or if the processing itself has failed.

The client allocates the request number. The server is responsible for distinguishing the sequence numbers from different clients. The client shall use a new request number with each request except when he wants to re-sent an earlier message.

The request numbers are specified as XML-type integer.

Remark: For the final specification, the request numbers may be uuid's. In this case, the data type of the request number parameters would change to anyURI. The "uuid:" URI scheme would be used for these parameters.

## Session Identification

For operations implementing the "reservation and deferred payment" model, the client needs to specify the transaction context a particular message is addressed to. This is done by means of the "chargingSessionID" parameter, which appears in all respective operations.

As the "requestNumber" parameter, these parameters are of the type integer. The client allocates them.

Remark: As for the request number mechanism, uuid's could be used instead of integers.