

Seamless SEQR: Cash Register Developer's Manual

version: 2.4.2.1

July 9, 2013

Table of Contents

1. Introduction	1
2. Business concepts	2
3. Supported Flows	3
3.1. Handling QR Codes	3
3.2. Terminal Registration	3
3.3. Basic Mobile Payment transaction	4
3.4. Buyer Actions	12
3.5. Reconciliation	12
4. Implementation Notes	14
4.1. Overview	14
4.2. Client context (ClientContext)	14
4.3. principalId	15
4.4. The Invoice, Receipt, and Receipt document	15
4.5. The terminal	16
4.6. Handling getPaymentStatus	17
4.7. Supported conventions and standards	17
4.8. customer token	17
4.9. Loyalty membership	18
4.10. Receipt document	19
4.11. Reconciliation	19
4.12. Webshops	19
4.13. Payments inside apps	20
4.14. Testing	20
5. Interface methods	21
5.1. sendInvoice	21
5.1.1. sendInvoice Description	21
5.1.2. sendInvoice request fields	21
5.1.3. sendInvoice response fields	22
5.1.4. sendInvoice SOAP examples	22
5.2. updateInvoice	23
5.2.1. updateInvoice Description	23
5.2.2. updateInvoice Description	23
5.2.3. updateInvoice response fields	23
5.2.4. updateInvoice SOAP examples	23
5.3. getPaymentStatus	24
5.3.1. getPaymentStatus Description	24
5.3.2. getPaymentStatus request fields	24
5.3.3. getPaymentStatus response fields	24
5.3.4. getPaymentStatus SOAP examples	25
5.4. cancelInvoice	26
5.4.1. cancelInvoice Description	26
5.4.2. cancelInvoice request fields	26
5.4.3. cancelInvoice response fields	26
5.4.4. cancelInvoice SOAP examples	27
5.5. registerTerminal	27
5.5.1. registerTerminal Description	27
5.5.2. registerTerminal request fields	27
5.5.3. registerTerminal response fields	28
5.5.4. registerTerminal SOAP examples	28
5.6. unregisterTerminal	28
5.6.1. unregisterTerminal Description	28
5.6.2. unregisterTerminal request fields	29
5.6.3. unregisterTerminal response fields	29
5.6.4. unregisterTerminal SOAP examples	29
5.7. assignSeqId	30

5.7.1. assignSeqrId Description	30
5.7.2. assignSeqrId request fields	30
5.7.3. assignSeqrId response fields	30
5.7.4. assignSeqrId SOAP examples	30
5.8. commitReservation	31
5.8.1. commitReservation Description	31
5.8.2. commitReservation request fields	31
5.8.3. commitReservation response fields	31
5.9. submitPaymentReceipt	31
5.9.1. submitPaymentReceipt Description	31
5.9.2. submitPaymentReceipt request fields	32
5.9.3. submitPaymentReceipt response fields	32
5.9.4. submitPaymentReceipt SOAP examples	32
5.10. refundPayment	33
5.10.1. refundPayment Description	33
5.10.2. refundPayment request fields	33
5.10.3. refundPayment response fields	33
5.10.4. refundPayment SOAP examples	34
5.11. markTransactionPeriod	34
5.11.1. markTransactionPeriod Description	34
5.11.2. markTransactionPeriod request fields	34
5.11.3. markTransactionPeriod response fields	35
5.11.4. markTransactionPeriod SOAP examples	35
5.12. executeReport	36
5.12.1. executeReport Description	36
5.12.2. executeReport request fields	36
5.12.3. executeReport response fields	37
5.12.4. executeReport SOAP examples	37
5.13. Result codes	37
6. Developing clients in Java	40
6.1. Overview	40
6.2. Samples	40
6.2.1. MPaymentFlowSample.java	40
6.2.2. MPaymentFlowWithLoyaltySample.java	41
7. Developing clients in C/C++	45
7.1. Overview	45
8. Reconciliation reports	46
8.1. Reconciliation reports	46
8.2. Samples	47
8.2.1. STD_RECON_001 SOAP examples	47
8.2.2. STD_RECON_002 SOAP examples	48
8.2.3. STD_RECON_003 SOAP examples	50
8.2.4. STD_RECON_004 SOAP examples	52
8.2.5. STD_RECON_006 SOAP examples	53
8.2.6. STD_RECON_007 SOAP examples	55

List of Figures

3.1. SEQR sticker	3
3.2. Sequence diagram normal payment flow	5
3.3. Sequence diagram payment with loyalty card on the phone	6
3.4. Sequence diagram payment with swiping of loyalty card before payment, no loyalty card on the phone	7
3.5. Sequence diagram payment with swiping of loyalty card before payment, same loyalty card as on the phone	8
3.6. Sequence diagram payment with swiping of loyalty card before payment, different loyalty card than on the phone	9
3.7. Sequence diagram payment with swiping of loyalty card after user scanned the qr code, user has no loyalty card	10
3.8. Sequence diagram payment with swiping of loyalty card after user scanned the qr code, user has no loyalty card	11
4.1. Overview of the SEQR solution	14

List of Tables

3.1. Handling QR codes	3
3.2. Interface methods involved in the terminal registration procedure	4
3.3. Interface methods related to buyers' activities	12
3.4. Interface methods related to reconciliation process	13
4.1. Client context fields	15
4.2. principalId	15
4.3. The terminal	16
4.4. customer token	18
4.5. receipt document	19
5.1. Interface methods	21
5.2. sendInvoice request fields	21
5.3. sendInvoice response fields	22
5.4. updateInvoice request fields	23
5.5. updateInvoice response fields	23
5.6. getPaymentstatus request fields	24
5.7. getPaymentstatus response fields	24
5.8. cancelInvoice request fields	26
5.9. cancelInvoice response fields	26
5.10. registerTerminal request fields	27
5.11. registerTerminal response fields	28
5.12. unregisterTerminal request fields	29
5.13. unregisterTerminal response fields	29
5.14. assignSeqId request fields	30
5.15. assignSeqId response fields	30
5.16. commitReservation request fields	31
5.17. commitReservation response fields	31
5.18. submitPaymentReceipt request fields	32
5.19. submitPaymentReceipt response fields	32
5.20. refundPayment request fields	33
5.21. refundPayment response fields	33
5.22. markTransactionPeriod request fields	34
5.23. markTransactionPeriod response fields	35
5.24. executeReport request fields	36
5.25. executeReport response fields	37
5.26. Result codes	38
5.27. Result codes (continued)	39
8.1. Reconciliation reports	46

List of Examples

5.1. An example of a sendInvoice SOAP request	22
5.2. An example of a sendInvoice SOAP response	22
5.3. An example of a updateInvoice SOAP request	23
5.4. An example of a updateInvoice SOAP response	24
5.5. An example of a getPaymentStatus SOAP request	25
5.6. An example of a getPaymentStatus SOAP response	26
5.7. An example of a cancelInvoice SOAP request	27
5.8. An example of a cancelInvoice SOAP response	27
5.9. An example of a registerTerminal SOAP request	28
5.10. An example of a registerTerminal SOAP response	28
5.11. An example of a unregisterTerminal SOAP request	29
5.12. An example of a unregisterTerminal SOAP response	29
5.13. An example of a assignSeqId SOAP request	30
5.14. An example of a assignSeqId SOAP response	31
5.15. An example of a submitPaymentReceipt SOAP request	32
5.16. An example of a submitPaymentReceipt SOAP response	33
5.17. An example of a refundPayment SOAP request	34
5.18. An example of a refundPayment SOAP response	34
5.19. An example of a markTransactionPeriod SOAP request, for per shop reconciliation	35
5.20. An example of a markTransactionPeriod SOAP response, for per shop reconciliation	35
5.21. An example of a markTransactionPeriod SOAP request, for per terminal reconciliation	36
5.22. An example of a markTransactionPeriod SOAP response, for per terminal reconciliation	36
5.23. An example of a executeReport SOAP request	37
5.24. An example of a executeReport SOAP response	37
8.1. An example of executing report STD_RECON_001 SOAP request	47
8.2. An example of executing report STD_RECON_001 SOAP response	47
8.3. An example of executing report STD_RECON_002 SOAP request	48
8.4. An example of executing report STD_RECON_002 SOAP response	49
8.5. An example of executing report STD_RECON_003 SOAP request	50
8.6. An example of executing report STD_RECON_003 SOAP response	51
8.7. An example of executing report STD_RECON_004 SOAP request	52
8.8. An example of executing report STD_RECON_004 SOAP response	52
8.9. An example of executing report STD_RECON_006 SOAP request	53
8.10. An example of executing report STD_RECON_006 SOAP response	54
8.11. An example of executing report STD_RECON_007 SOAP request	55
8.12. An example of executing report STD_RECON_007 SOAP response	56

Chapter 1. Introduction

This document describes SEQR, the Seamless Mobile Payment Solution, and provides guidelines for the development of cash register clients that want to receive payments through the SEQR system.

Chapter 2 (Business Concepts) sets the scene; chapters 3 (Supported Flows) and 4 (Implementation Notes) provide guidance for a developer, with chapter 5 providing the library of interface methods (and sample code).

Chapter 6 and 7 describe the development of clients in Java and C/C++. Chapter 8 discusses reconciliation details.

Although focusing on the development of external cash register clients, this manual also supports the development of clients for webshops (see section 4.9) or a gateway for payment terminals.

“NOTES” and “TIPS” to developers can be found in chapters 3, 4, and 5.

Chapter 2. Business concepts

A seller sells products (potentially including services) to a buyer.

A Mobile Payment (or SEQR payment) is a transaction between a seller, whose cash register generates a invoice, and a buyer, whose can use a mobile device (for example, a smart phone) to read a QR (Quick Response) code which then allows the buyer to receive the invoice and approve payment.

An invoice is used to specify the products being bought, the cost of each product, any applied discounts, the cash register at which the invoice was created, and the amount to be paid by the buyer.

A receipt is a proof of a completed payment.

A receipt document is a seller's document which includes information of interest concerning a transaction.

Discounts can be the result of, for example, loyalty programmes (based upon repeat business and covering both a discount on the current purchase and an allotment of points that can be used for future purchases), paper or paperless (electronic) coupons of varies types, and other seller specific sales campaigns. Discounts use:

- a buyer's membership number (e.g. the seller's loyalty programme number) which is used by a seller to identify the buyer
- electronic coupons assigned to the buyer's membership number or paper coupons (presented by the buyer at the time of purchase) are handled internally by the seller as part of the calculation of the amount included in the invoice sent to the SEQR service.

The buyer's handling of membership numbers and coupons are not described within this document.

The total invoice amount is the amount that the seller has to pay the buyer after all discounts have been applied.

A terminal is the logical payment point that wants to receive a payment, in a store this would typically correspond to a cash register or a payment terminal whereas a webshop might only need one terminal unless different parts of the store wants to be presented as separate payment points. The cash register is the seller client within the terminal used to produce the invoice and communicate with the SEQR service using the interface methods defined in section 5.

A QR (Quick Response) Code is a matrix (or two-dimensional) bar code which identifies the specific terminal used by the seller/buyer for a specific Mobile Payment transaction.

The Financing Core System represents the buyer's credit line provider.

Reconciliation is the process used to report on the total number of invoices from an individual cash register handled by the SEQR service during a specified period.

Chapter 3. Supported Flows


The following flows are described:

- Handling QR codes
- Terminal Registration
- Basic Mobile Payment Transaction
- Additional Buyer Actions.
- Reconciliation

Handling QR codes is a manual routine not requiring client support.

3.1. Handling QR Codes

Table 3.1. Handling QR codes

<p>Seamless sends each seller a number of pre-printed SEQR stickers (see figure).</p> <p>Each sticker includes the QR Code matrix (a two-dimensional bar code) and the SEQR ID linear bar code.</p> <p>The SEQR ID is used to register the terminal; the QR Code is scanned by the buyer to receive the cash register generated invoice.</p> <p>The SEQR service uses a static POS (Point of Sale) QR code with an HTTP://SEQR.SE/Q[SEQR ID] URL.</p> <p>TIPS The SEQR sticker should be placed in a visible place in front of the cash register.</p>	<p>Figure 3.1. SEQR sticker</p> 
---	--

3.2. Terminal Registration

Terminals used in Mobile Payment must be registered in the SEQR service; terminal registration consists of:

- Terminal Registration which assigns a (unique) ID to the terminal.
- SEQR ID Assignment which assigns a SEQR ID to a terminal to identify the unique QR Code to be used at the terminal.

TIPS the following flow is recommended (but not mandatory):

Run the cashregister (CR) application:

1. CLICK on the option for registering the cashregister/enabling SEQR payments within the CR.
2. PROVIDE the shop (reseller) authentication information requested by the CR (that is the reseller authentication information that is provided to each shop)

3. The CR registers the cash register within the SEQR service and generates and saves a random password on the cash register for future communications.
4. ENTER the SEQR ID requested by the CR (the SEQR ID can be entered manually or can be scanned from the barcode from the SEQR sticker)
5. The CR calls the SEQR service to assign the SEQR ID to the terminal (using the cash register authentication registered in step 3)
6. The cash register is now registered

These steps are reflected in the following series of interface methods (with only important fields shown):

Table 3.2. Interface methods involved in the terminal registration procedure

registerTerminal request	externalTerminalId (identifier of the terminal in the client system, e.g. "Store 111/Till 4") password (for future communications with the SEQR service) name (to appear on the buyer's mobile device, e.g. "My Restaurant, cash register 2")
registerTerminal response	resultCode terminalId (for communications with the SEQR service)
assignSeqrId request	seqrId (obtained from SEQR sticker).
assignSeqrId response	resultCode

3.3. Basic Mobile Payment transaction

The basic Mobile Payment transaction consists of (see sequence diagrams below for more exact details):

1. The cash register generates an invoice including the amount and sends the invoice to the SEQR service if the user has swiped a loyalty card the token is sent as well.
2. The buyer uses its mobile device to read the QR Code on the SEQR sticker placed next to the cash register.
3. The buyer's mobile device sends in the buyers loyalty tokens to the SEQR Service if a loyalty exists. Note that the information is not sent if a loyalty card has been swiped. The swiped loyalty card always overrides the phone.
4. The SEQR Service forwards the buyers loyalty tokens to the cash register if no loyalty card has been swiped and loyalty card exists on the phone.
5. The cash register processes the buyers loyalty tokens and where it decides that a discount needs to be applied sends an updated invoice including the total invoice amount
6. The SEQR service sends the invoice / amount to the mobile device for approval by the buyer
7. The buyer approves the amount using its PIN
8. The SEQR service informs the cash register that the payment has been confirmed
9. The SEQR service informs the buyer's mobile device that the confirmation has been sent
10. The cash register uploads the final receipt to the SEQR service which then informs the buyer's mobile device

Figure 3.2. Sequence diagram normal payment flow

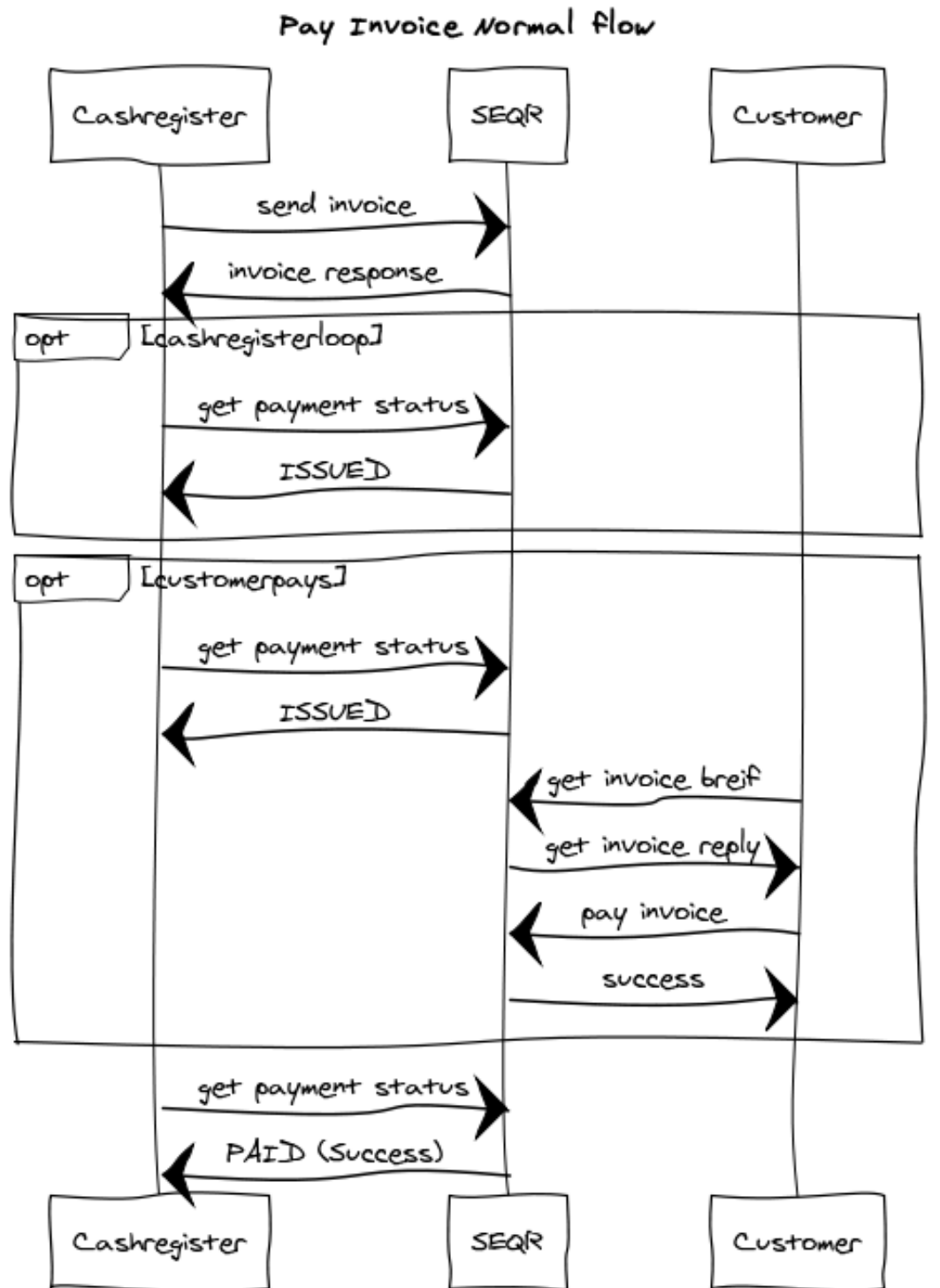
www.websequencediagrams.com

Figure 3.3. Sequence diagram payment with loyalty card on the phone

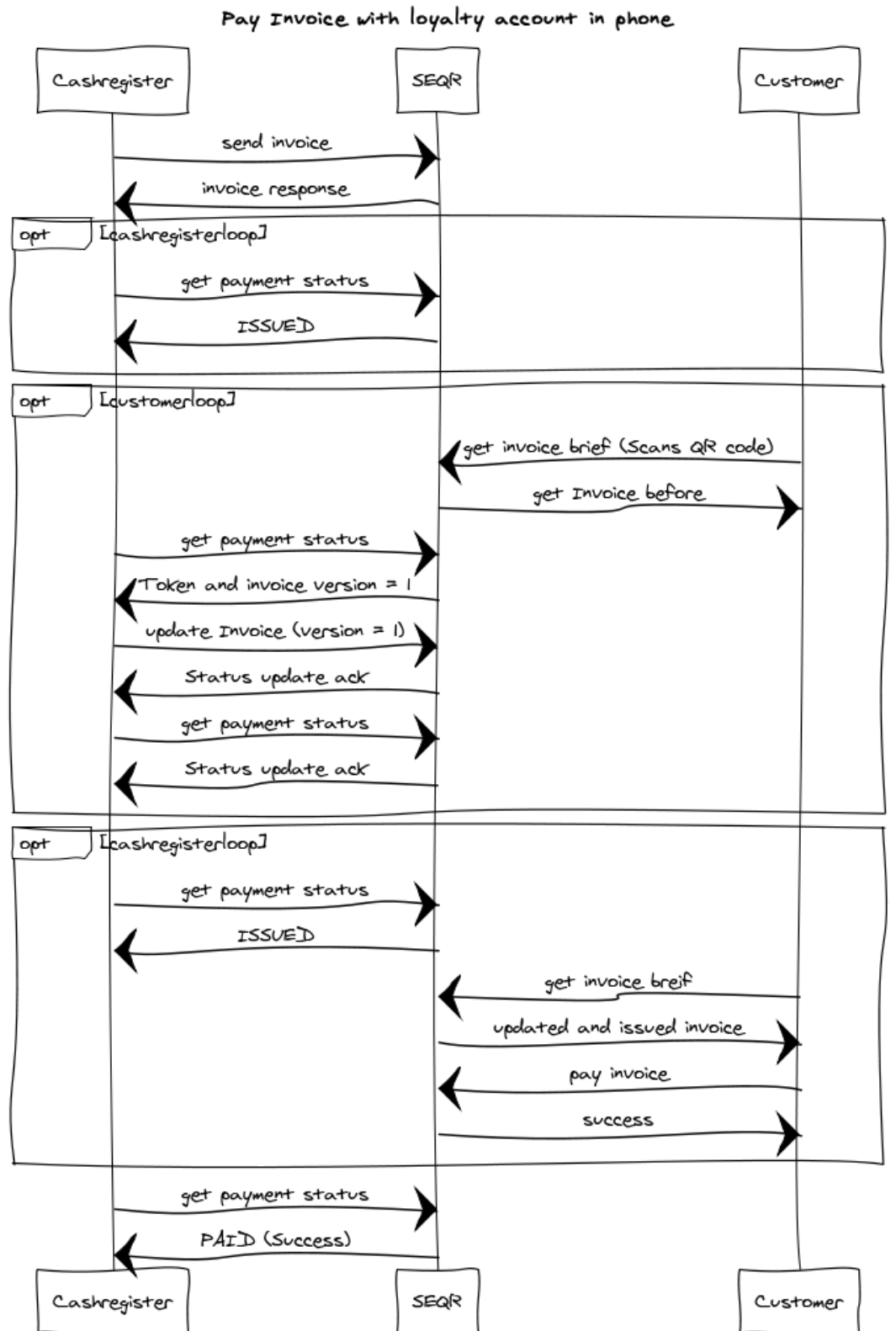
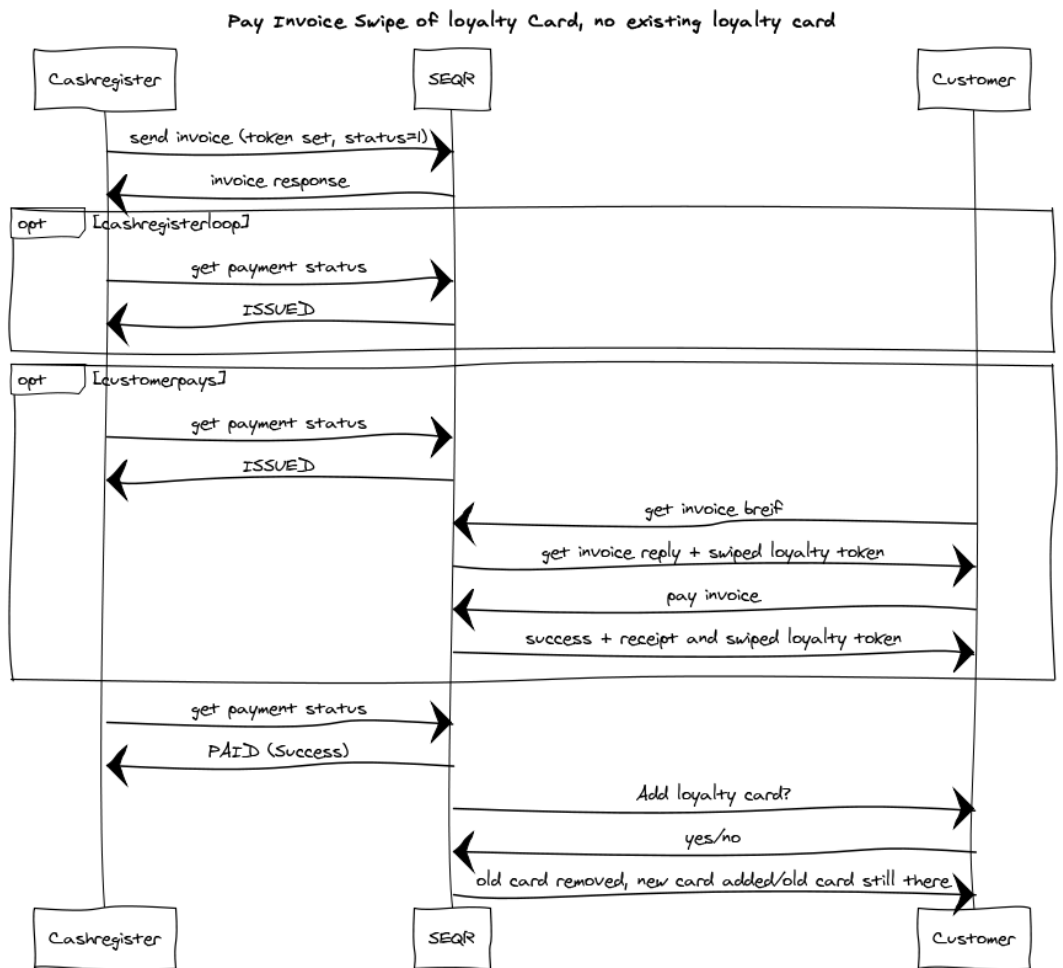


Figure 3.4. Sequence diagram payment with swiping of loyalty card before payment, no loyalty card on the phone



www.websequencediagrams.com

Figure 3.5. Sequence diagram payment with swiping of loyalty card before payment, same loyalty card as on the phone

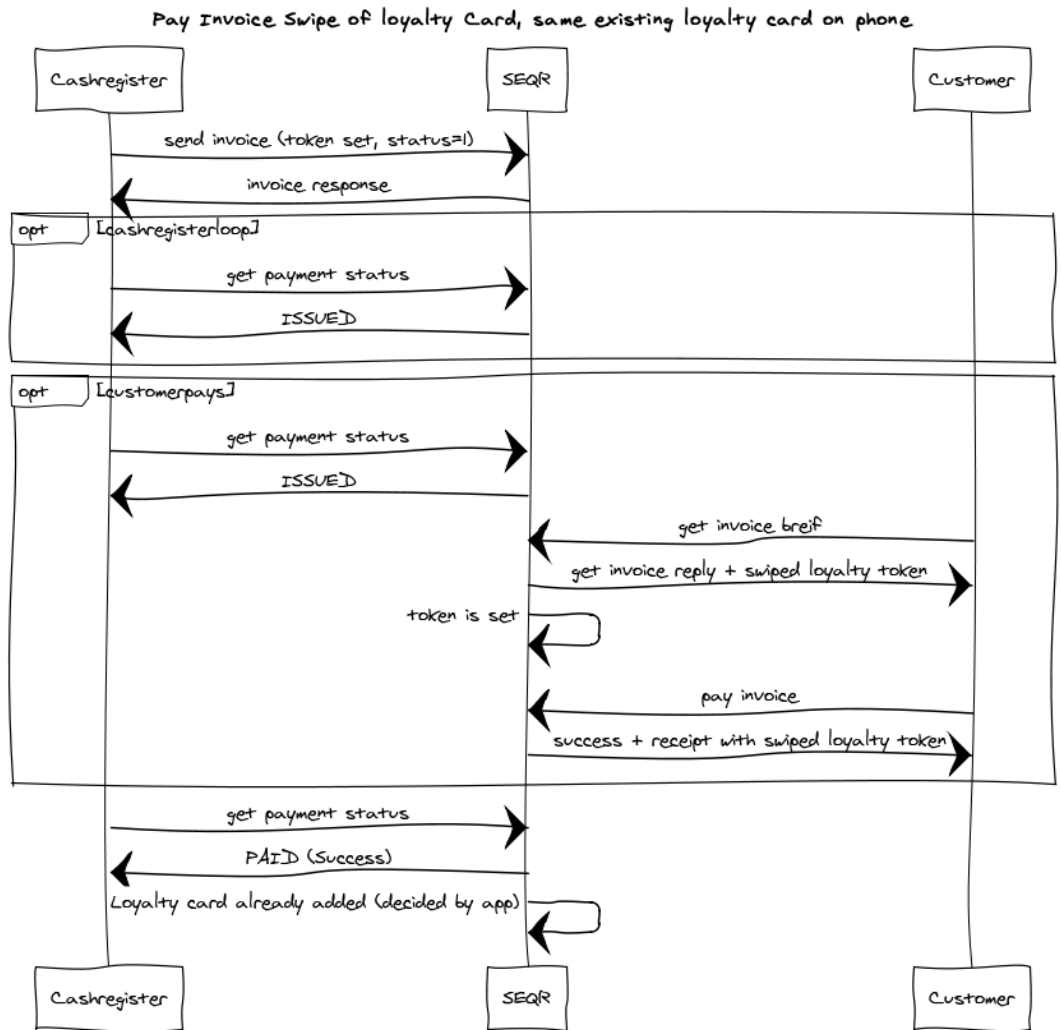


Figure 3.6. Sequence diagram payment with swiping of loyalty card before payment, different loyalty card than on the phone

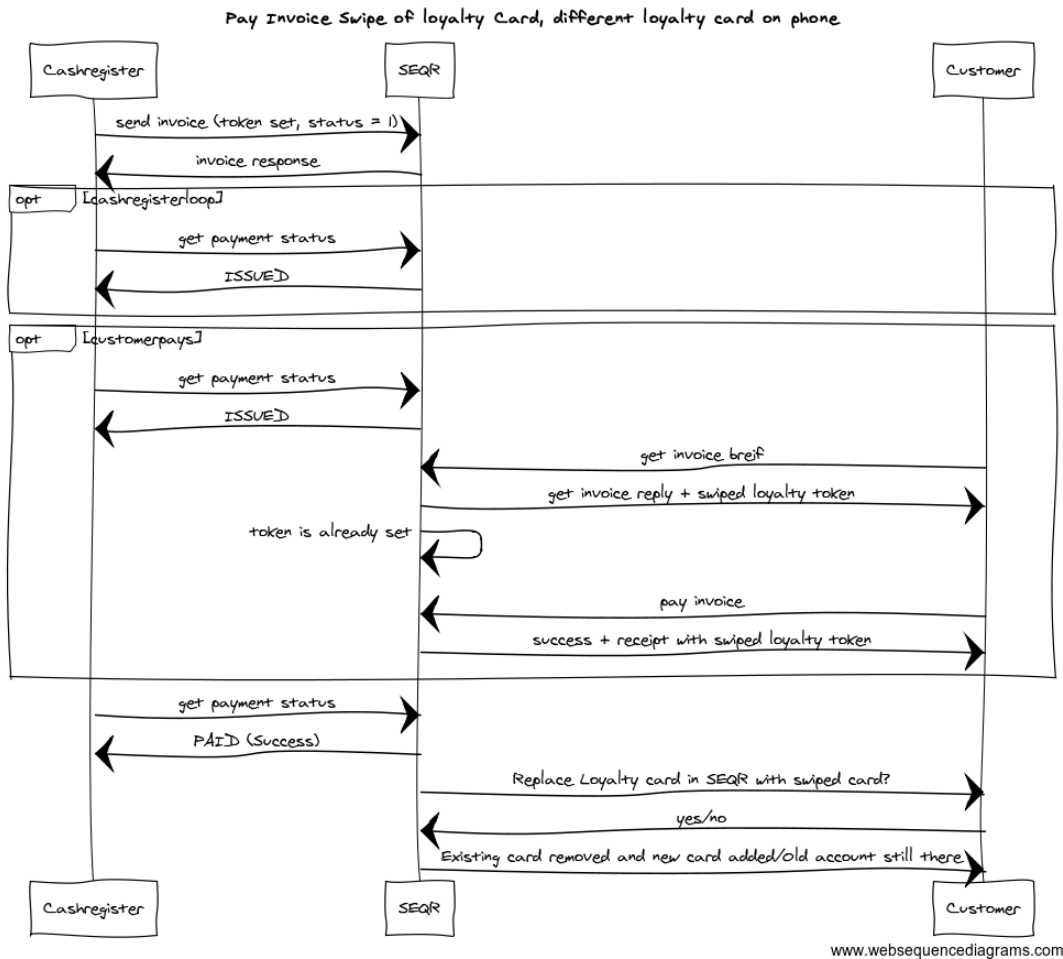
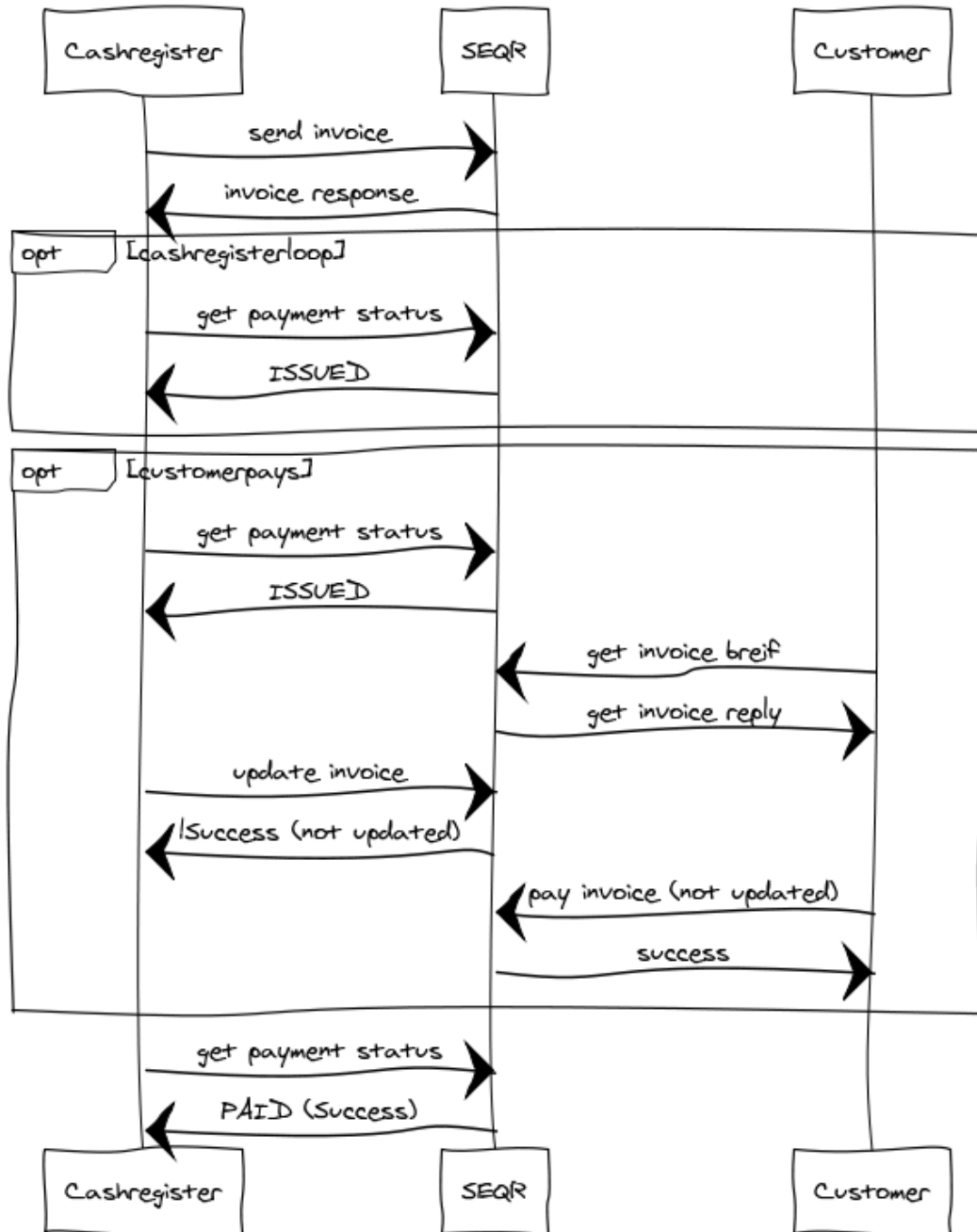


Figure 3.7. Sequence diagram payment with swiping of loyalty card after user scanned the qr code, user has no loyalty card

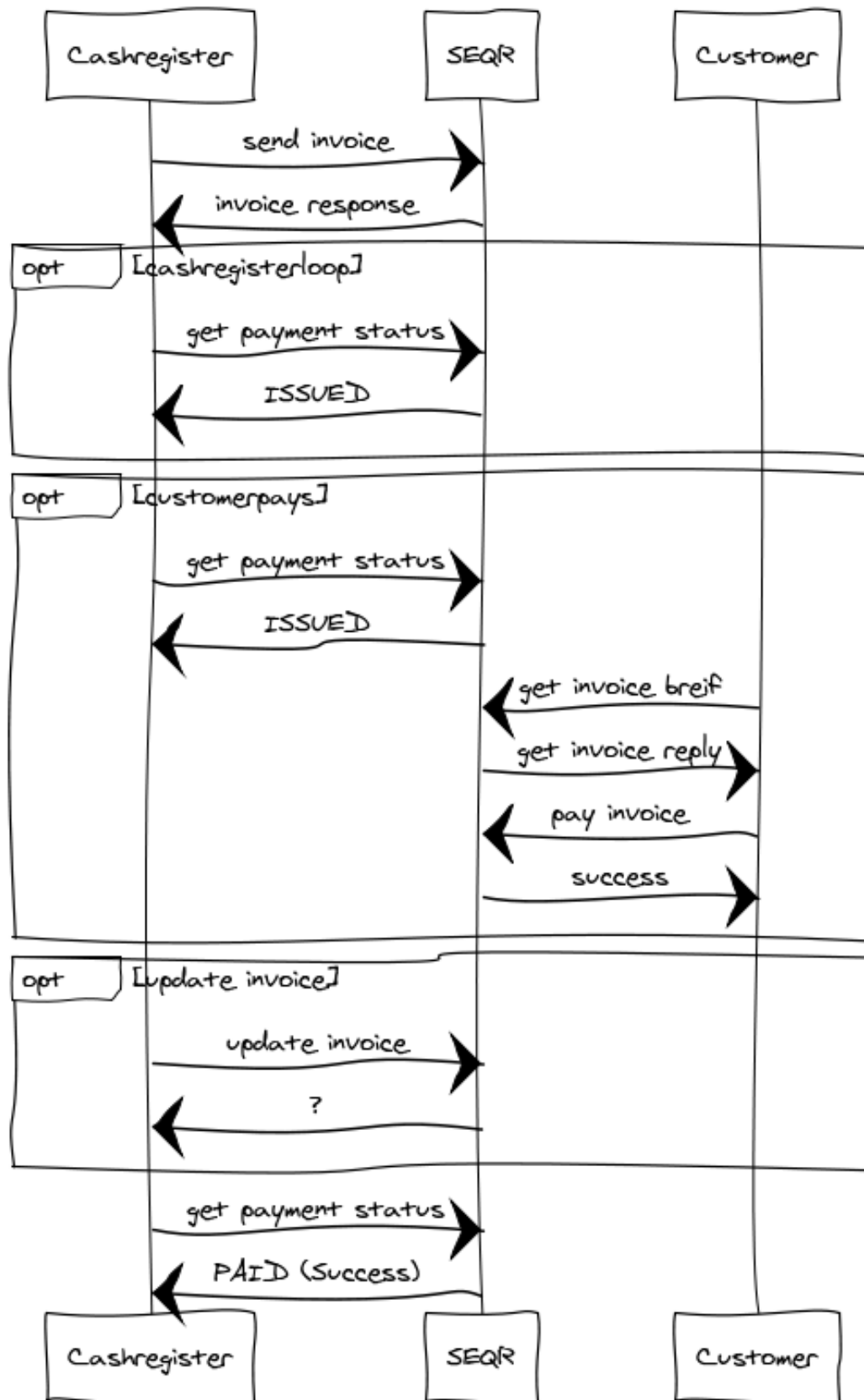
Pay Invoice Swipe of loyalty Card after user scanned, no existing loyalty card. Trying to update after user scanned invoice



www.websequencediagrams.com

scanned

Pay Invoice Swipe of loyalty Card after user scanned,
no existing loyalty card



www.websequencediagrams.com

getPaymentStatus is sent repeatedly by the client (see 4.6).

3.4. Buyer Actions

Possible scenarios that may be performed on an invoice:

- Send invoice: The invoice payment has been initiated by the cashregister. All products sold are included in the invoice.
- Remove from invoice: Ask for one, or more, products to be removed from the invoice (“sorry, I’ve changed my mind, can you take away product A”)
- Add to invoice: Ask for one, or more, products to be added from the invoice (“sorry, I left product A in my basket, can you add it to the invoice”)
- Cancel invoice: Cancel payment of the invoice at any time prior to the confirmation that the payment has been made (“sorry, I’ve changed my mind, I don’t want to buy anything”)
- Refund: The buyer is asked to provide a refund for one, or more products from the invoice after confirmation that the payment has been made.

The refund might occur shortly after the payment, “sorry, I’ve just paid for product B but have changed my mind”, or after a longer period, “sorry, I bought product B yesterday and have changed my mind”.

The handling of refunds is dependent upon the seller’s policy.

The SEQR service periodically (e.g. daily) informs the seller of the list of refunds. the seller uses this list within its normal refund routines. With the exception of this report, SEQR provides no additional support for handling refunds.

These buyer activities are supported with the following interface methods:

Table 3.3. Interface methods related to buyers' activities

Buyer activity	Interface method	Comments
Send invoice	sendInvoice	
Remove from invoice	updateInvoice	
Add to invoice	updateInvoice	
Cancel invoice	cancelInvoice	
Refund	refundPayment	Depending upon the seller’s processes a payback may take several days

3.5. Reconciliation

The following are steps to perform reconciliation against SEQR from cashregisters

1. At the end of a working shift or shop hour, a cashier presses a button 'Close & Reconcile' on a cashregister. The cashregisters application send markTransactionPeriod request to SEQR to mark end of transactions list for this period. SEQR returns with a unique reference number, ersReference.
2. The cashregister application waits for a couple of seconds (around 3 seconds) in order to make sure that all transactions are ready to process for reconciliation report.
3. The cashregister application calls executeReport using ersReference from step1 to fetch reconciliation report representing transaction summary since the previous reconciliation until the end of transaction list for this period.

4. In case the reconciliation report is not ready , SEQR will return with result code 2 (REPORT_NOT_READY). The cashier should wait for couple seconds (around 5 seconds) more and repeat step3 again.

Table 3.4. Interface methods related to reconciliation process

Reconcile activity	Interface method
Close period	markTransactionPeriod
Get reconciliation report	executeReport

There are two ways of reconciliation against SEQR service:

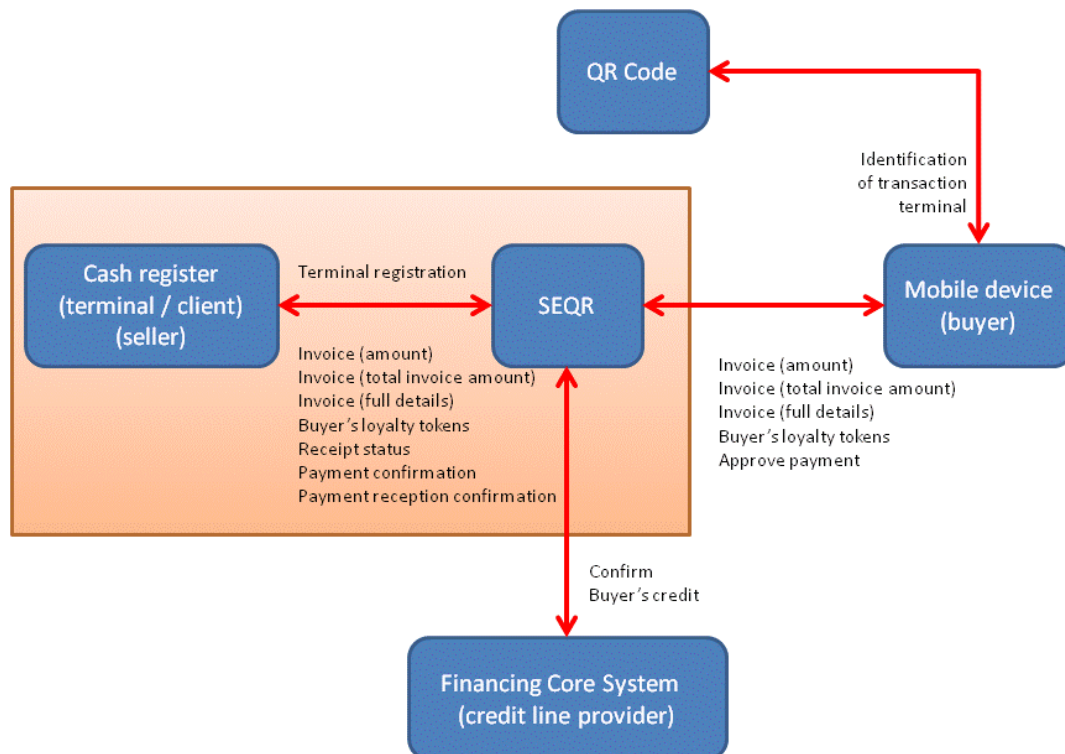
- Per shop reconciliation: Only one master cashregister perform reconciliation process. The reconciliation report will show transactions summary for every cashregisters in the shop
- Per terminal reconciliation: Every cashregister in a shop perform reconciliation process. The reconciliation report will show transactions summary only for the specific cashregister. Note that only a few of the reports can run in this mode. Please refer to **Reconciliation Reports** for more information.

Chapter 4. Implementation Notes

4.1. Overview

The following figure illustrates the SEQR Solution, in particular, the location of the ERSWSEExternalClientService exposed by the SEQR External Client Proxy; this proxy supports communication with external clients via the SOAP protocol in Web Services environment.

Figure 4.1. Overview of the SEQR solution



Where a protocol other than SOAP is preferred, the developer should create a proxy to serve as an intermediate point between the client and the SEQR service. The client and the proxy can then use the preferred protocol while the proxy and the SEQR service will use SOAP.

Communication in the production environment must use a secure channel, namely HTTPS.

The SEQR solution includes two web service locations, one for integration and testing and one for production. All sellers use the same web service locations.

4.2. Client context (ClientContext)

A principal is the main actor in each request to the SEQR service and represents either a seller or a buyer. Each request will have at least an initiator principal.

The ClientContext structure is used in all requests to identify, authenticate and authorize the client initiating the transaction. For authentication the credentials of the initiator principal are used. As all transactions take place over a secure channel (typically HTTPS) the ClientContext is sent in clear text.

Table 4.1. Client context fields

Name	Description
clientId	Client id identifies the software with which the SEQR service is communicating, for example “CashRegisterManager version 1.3.4
Channel	The channel used to send a request. TIPS always use ClientWS.
clientRequestTimeout	The client side timeout for the request. If the response is not received before the timeout the client will attempt to abort the request. TIPS it is recommended that the clientRequestTimeout should be set to 0, so there will not be any client forced timeouts in the SEQR service.
initiatorPrincipalId	Principal ID of the principal that initiates the request (see section 4.3).
Password	The password used to authenticate the initiator principal.
clientReference	The client reference for the transaction. TIPS the clientReference should be unique at least for the specific client id. NOTE the SEQR service does not check this field. The field has a maximum length of 32 characters. The field is optional.
clientComment	Client comment included within the request. The field is optional.

4.3. principalId

Principal refers to a actor; within a specific transaction the actor / principal can have different roles; initiator, sender, or receiver.

principalId is used for authentication of the principal (seller) and contains the id and type, as well as an optional user id. The combination is unique within the SEQR service.

Table 4.2. principalId

Type	Description/Use	userId mandatory
RESELLERUSER	Used when seller will use one of its user account.	Yes
TERMINALID	Used when seller will use his terminal id.	No

The Reselleruser is used in the registerTerminal, markTransactionPeriod and executeReport interface methods; other interface methods use the TerminalId

4.4. The Invoice, Receipt, and Receipt document

The invoice specifies what is being bought and the amount to be paid, the receipt is a proof that the buyer has paid, and the receipt document is the seller’s version of the receipt an may include information of interest for the seller not included within the receipt or information only available after the receipt was created.

- The seller creates (before the buyer scans in the QR Code) and updates (after the seller has received loyalty and coupon information about the buyer).
- The SEQR service generates the receipt after payment and sends the receipt to the cash register and to the buyer's mobile device.
- The seller creates the receipt document after receiving the SEQR generated receipt and sends the receipt document to the SEQR service.

The invoice includes:

- Time and Date when the invoice was issued
- Invoice title
- Seller's (client system) invoice identifier
- Total invoice amount (after all discounts have been applied) to be paid by the buyer
- Cashier identifier, e.g. Mikael
- An optional URL to launch in the app after the payment is completed. This is mainly used for webshop payments (to jump back to the webshop on the phone) or for in-app payments (to jump back to the app using an URL scheme).

The receipt includes:

- ID of the Terminal at which the invoice was issued
- Time and Date when the invoice was paid
- Seller's (client system) invoice identifier
- SEQR service reference to the paid invoice
- Reference to the payment transaction
- Buyer's terminal ID

The receipt document should, preferably, use the ARTS Receipt XML format.

4.5. The terminal

Table 4.3. The terminal

Parameter	Description	Related method	interface	Choice of parameter value
seqrId	SEQR ID assigned to a terminal (but not used internally within the SEQR service)	assignSeqrId		To minimize the probability of conflicts within our solution, Seamless controls the SEQR service identities assigned and sent to a Seller. Each brand or shop is assigned a range of SEQR IDs which can be given to the Seller if needed.

Parameter	Description	Related interface method	Choice of parameter value
terminalId	SEQR ID for this terminal (used in communications between the SEQR service and client) after registerTerminal	registerTerminal response	
externalTerminalId	The identifier for this terminal in the external system and for future cross checks between the system and the SEQR service)	registerTerminal	e.g. "Store 111/Till 4"
password	Password for future communications with the SEQR service.	registerTerminal	TIPS generate a random password at registration time and save it on the terminal for future communications.
name	The name to appear on the buyer's mobile device	registerTerminal	e.g. "My Restaurant, cash register 2"

4.6. Handling getPaymentStatus

The SEQR service does not send information concerning the change in status of an invoice (invoiceStatus); the client needs to repeatedly query the status of the payment with getPaymentStatus". TIPS to minimise "buyer-seller transaction time" but not negatively affect client-SEQR response time, getPaymentStatus should be sent once per second. Note! If getPaymentStatus is not queried, payment done by the client will be refunded.

4.7. Supported conventions and standards

Amounts are presented as a value and an ISO 4217 currency code currency string. For information about currencies and number of digits see: http://www.iso.org/iso/support/faqs/faqs_widely_used_standards/widely_used_standards_other/currency_codes/currency_codes_list-1.htm.

ARTS

- How is ARTS supported / followed within SEQR
- What version of ART is supported
- Plan for migration to next version of ARTS and compatibility issues
- Compatibility with earlier version of ARTS (currently implemented by the Seller)
- Constraints placed upon a client implementation
- Reference to ARTS website: <http://www.nrf-arts.org/content/arts-xml-0>

4.8. customer token

A customer token is used to communicate meta data about an invoice/payment.

When fetching the payment status, SEQR may communicate a set of customer tokens to the merchant that are applicable for the payment. The merchant will then have to decide which tokens are applied and send them back with the updateInvoice call.

Table 4.4. customer token

Type	Description/Use	mandatory
type	A string identifier for the type of token	Yes
value	The value of the token, eg the card number, coupon code etc.	Yes
status	Status of this token: <ul style="list-style-type: none"> • 0 - Pending usage (when sent from SEQR) • 1 - Used in this payment (when updated from merchant) • 90 - Blocked, cannot be used • 99 - Invalid token, unknown by merchant 	Yes
description	Textual description of this token	Yes

Customer tokens can be used for different purposes, loyalty memberships, coupons etc

4.9. Loyalty membership

SEQR has the capability of informing the merchant that the customer has a membership program that is relevant for the merchant so it can be applied to the purchase. Normally, the merchant will only receive one loyalty program token for a specific payment.

When the merchant receives a token that is identified as a loyalty program token, it should apply the loyalty locally and then call `updateInvoice()` to confirm to SEQR the updated amount and other information along with the token that was actually used.

When the merchant receives a token that is identified as a loyalty program token, it should apply the loyalty locally and then call `updateInvoice()` to confirm to SEQR the updated amount and other information along with the token that was actually used.

Sample loyalty flow where SEQR informs merchant:

1. Customer finishes purchase at merchant (ie all items scanned)
2. Merchant creates a new invoice with `sendInvoice()`
3. Customer scans the SEQR code and is assigned to the payment
4. Merchant gets response from `getPaymentStatus()` with invoice status `PENDING_ISSUER_ACKNOWLEDGE`. Also the response contains customer token with type "X_CARD" and the value as the card number and status 0 (PENDING)
5. Merchant applies the loyalty card rules (eg adding a % discount or registering points etc) and then calls `updateInvoice` to update the amount and to send the used customer token: "X_CARD", card number and status 1 (used)
6. Customer receives information about the updated amount etc to confirm the payment
7. Customer confirms payment by entering the PIN
8. Merchant and customer are informed of the successful payment

Sample loyalty flow where merchant informs SEQR:

1. Customer finishes purchase at merchant (ie all items scanned) including swiping loyalty card
2. Merchant creates a new invoice with `sendInvoice()` attaching the loyalty card info as a token (eg "X_CARD", the card number, status 1 and the user friendly name of the loyalty card as description)
3. Customer scans the SEQR code and is assigned to the payment
4. Customer confirms payment by entering the PIN
5. Merchant and customer are informed of the successful payment
6. After the payment if a loyalty card has been swiped, the customer (depending on card rules) might get the question whether this loyalty card should be added to the users SEQR profile so that the user doesn't have to swipe it the next time

4.10. Receipt document

One or more receipt document can be uploaded to a payment with `submitPaymentReceipt`.

Table 4.5. receipt document

Type	Description/Use	mandatory
contentType	MIME type string indicating the MIME type of the receipt data. Usually text/html but other formats can be supported if agreed with SEQR.	Yes
receiptData	The actual receipt data in the format indicated by the contentType	Yes
receiptType	Type of data the receipt contains <ul style="list-style-type: none">• PAYMENT_RECEIPT - The actual payment receipt/slip specifying what was paid for etc• WARRANTY - A warranty receipt for one or more of the items paid for• VOUCHER - A voucher paid for with the purchase• OTHER - Misc receipt document	Yes

4.11. Reconciliation

Reconciliation is the process used to report on the total number of invoices from an individual cash register handled by the SEQR service during a specified period.

Reconciliation includes the marking of the end of one and the beginning of a new transaction period (`markTransactionPeriod`), and the generation (execution) of the reconciliation report for the transaction period (`executeReport`)

4.12. Webshops

Although focusing on the development of external cash register clients, this manual also supports the development of clients for webshops.

The handling of webshops and cash registers differs in the some aspects covered below.

Terminal registration: whereas cash registers are registered by the seller, webshop registration is performed by Seamless during the creation of the web-site..

QR Codes: cash registers use QR Codes included within a QR Sticker; these QR Codes are unique for the terminal and do not change over time. For webshops QR Codes are produced by the SEQR service for each transaction and sent to the webshop in the sendInvoice response.

When running on a mobile, the webshop should not generate a QR code. Instead the webshop should show the a link to the QR code but replace the http: header in the url with seqr: this will ensure that the link will launch the SEQR app properly.

So if sendInvoice returns HTTP://SEQR.SE/R12345 the webshop could show a link like: ``

When testing against a SEQR test server, the app will typically replace with SEQR-DEBUG: instead of SEQR: as a URL header.

Additionally, the mobile webshop should supply the backURL in the sendInvoice to be the link that the SEQR app should return to after the payment is completed.

NOTE: It is up to the webshop to check the payment status using `getPaymentStatus` after returning, the link will be the same regardless of the payment status.

4.13. Payments inside apps

In-app payments can be made very convenient with SEQR payments.

All apps wanting to offer payments typically have a backend that handles the actual service that is sold. It is this backend that will have to connect to SEQR using this interface and the communication between the app and the backend is up to the app developer.

Terminal registration is done as for webshops, a single terminal is registered for the app backend which is then used for all transactions.

In the same way as webshops running on the mobile, the app should not show a QR code, instead it should show a button that is used to pay with SEQR. Once the button is pressed the app should launch the SEQR app using the QR code URL returned from sendInvoice but replacing the HTTP: header with SEQR: just as for mobile webshops.

So if sendInvoice returns HTTP://SEQR.SE/R12345 the button should launch the SEQR app using the URL SEQR://SEQR.SE/R12345.

When testing against a SEQR test server, the app will typically replace with SEQR-DEBUG: instead of SEQR: as a URL header.

Additionally, the app should supply the backURL in the sendInvoice to be the link that the SEQR app should return to after the payment is completed. Typically this will be a URL with a URL scheme that will launch the calling app again.

NOTE: It is up to the webshop to check the payment status using `getPaymentStatus` after returning, the link will be the same regardless of the payment status.

4.14. Testing

The SEQR CR start up instructions guide contains a link to the CR testing checklist and other relevant information (e.g. information on connecting to test servers). After performing an integration the seller tests using this CR testing checklist. Seamless and the seller will then meet and jointly go through the CR testing checklist.

Chapter 5. Interface methods

This section describes each of the following interface methods; the used Result Codes are listed at the end of the section.

Table 5.1. Interface methods

Method	Request	Response
sendInvoice	Sends an invoice to the SEQR service	SEQR invoice reference
updateInvoice	Updates an already sent invoice with new set of invoice rows or attributes (e.g. total invoice amount); used to support loyalty programmes etc	
commitReservation	Commits a payment that	
getPaymentStatus	Obtains status of a previously submitted invoice	invoiceStatus
cancelInvoice	Cancels an unpaid invoice	
registerTerminal	Registers a new terminal in the SEQR service	
unregisterTerminal	Unregisters an already registered terminal	
assignSeqrId	Assigns a SEQR ID to a terminal	
submitPaymentReceipt	Sends the receipt document of a payment or refund	
refundPayment	Refunds a previous payment	
markTransactionPeriod	Marks the end of one and the beginning of a new transaction period; used in reporting	
executeReport	Executes a report on the SEQR service.	Executed report

5.1. sendInvoice

5.1.1. sendInvoice Description

Sends an invoice to the SEQR service.

5.1.2. sendInvoice request fields

Table 5.2. sendInvoice request fields

Name	Description
Context	The ClientContext object (see Implementation Notes).
Invoice	Invoice data, which contains the amount and other invoice information.

5.1.3. sendInvoice response fields

Table 5.3. sendInvoice response fields

Name	Description
ersReference	Not used by this method (will be null after this method).
resultCode	Request result code (see Result Codes).
invoiceQRCode	SEQR generated QR Code (used for webshops; not relevant for cash registers)
resultDescription	A textual description of resultCode (see Result Codes).
invoiceReference	The SEQR service reference to the registered invoice.

5.1.4. sendInvoice SOAP examples

Example 5.1. An example of a sendInvoice SOAP request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:ext="http://external.interfaces.ers.seamless.com/">
  <soapenv:Header/>
  <soapenv:Body>
    <ext:sendInvoice>
      <context>
        <channel>extWS</channel>
        <clientComment>comment</clientComment>
        <clientId>testClient</clientId>
        <clientReference>12345</clientReference>
        <clientRequestTimeout>0</clientRequestTimeout>
        <initiatorPrincipalId>
          <id>87e791f9e24148a6892c52aa85bb0331</id>
          <type>TERMINALID</type>
        </initiatorPrincipalId>
        <password>1234</password>
      </context>
      <invoice>
        <title>Some Invoice</title>
        <cashierId>Bob</cashierId>
        <totalAmount>
          <currency>SEK</currency>
          <value>10.22</value>
        </totalAmount>
      </invoice>
    </ext:sendInvoice>
  </soapenv:Body>
</soapenv:Envelope>
```

Example 5.2. An example of a sendInvoice SOAP response

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns2:sendInvoiceResponse xmlns:ns2="http://
external.interfaces.ers.seamless.com/">
      <return>
        <resultCode>0</resultCode>
        <resultDescription>SUCCESS</resultDescription>
        <invoiceQRCode>http://seqr.se/R1328543027208</invoiceQRCode>
        <invoiceReference>1328543027208</invoiceReference>
      </return>
    </ns2:sendInvoiceResponse>
  </soap:Body>
</soap:Envelope>
```

5.2. updateInvoice

5.2.1. updateInvoice Description

Updates an already sent invoice with new set of invoice rows or attributes.

5.2.2. updateInvoice Description

Table 5.4. updateInvoice request fields

Name	Description
context	The ClientContext object (see Implementation Notes).
invoice	Invoice data, which contains the amount and other invoice information
invoiceReference	The SEQR service reference to the registered invoice.
tokens	The customer tokens applied to this invoice.

5.2.3. updateInvoice response fields

Table 5.5. updateInvoice response fields

Name	Description
ersReference	Not used by this method (will be null after this method).
resultCode	Request result code (see Result Codes).
resultDescription	A textual description of resultCode (see Result Codes).

5.2.4. updateInvoice SOAP examples

Example 5.3. An example of a updateInvoice SOAP request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:ext="http://external.interfaces.ers.seamless.com/">
  <soapenv:Header/>
  <soapenv:Body>
    <ext:updateInvoice>
      <context>
        <channel>extWS</channel>
        <clientComment>comment</clientComment>
        <clientId>testClient</clientId>
        <clientReference>12345</clientReference>
        <clientRequestTimeout>0</clientRequestTimeout>
        <initiatorPrincipalId>
          <id>87e791f9e24148a6892c52aa85bb0331</id>
          <type>TERMINALID</type>
        </initiatorPrincipalId>
        <password>1234</password>
      </context>
      <invoice>
        <title>Some Invoice</title>
        <cashierId>Bob</cashierId>
        <totalAmount>
          <currency>SEK</currency>
          <value>10.22</value>
        </totalAmount>
      </invoice>
    </ext:updateInvoice>
  </soapenv:Body>
</soapenv:Envelope>
```

Example 5.4. An example of a updateInvoice SOAP response

```

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns2:updateInvoiceResponse xmlns:ns2="http://
external.interfaces.ers.seamless.com/">
      <return>
        <resultCode>0</resultCode>
        <resultDescription>SUCCESS</resultDescription>
      </return>
    </ns2:updateInvoiceResponse>
  </soap:Body>
</soap:Envelope>

```

5.3. getPaymentStatus

5.3.1. getPaymentStatus Description

Obtains status of a previously submitted invoice.

5.3.2. getPaymentStatus request fields

Table 5.6. getPaymentstatus request fields

Name	Description
context	The ClientContext object (see Implementation Notes).
invoiceReference	The SEQR service reference to the registered invoice.
invoiceVersion	<p>Version of the invoice.</p> <p>The first time that it uses getPaymentStatus method the client sets the invoiceVersion to zero.</p> <p>The SEQR service increments the invoiceVersion in response message when:</p> <ul style="list-style-type: none"> the state of the payment (invoiceStatus) changes, or, a new buyer token is provided to be considered in the invoice. <p>In subsequent uses of the getPaymentStatus method, the client must use the latest value of invoiceVersion as an acknowledgement that it has received the latest change.</p>

5.3.3. getPaymentStatus response fields

Table 5.7. getPaymentstatus response fields

Name	Description
ersReference	The unique reference generated by the SEQR service once the invoice has been paid (null for all other invoiceStatus than PAID invoice has been paid).
resultCode	Request result code (see Result Codes).
resultDescription	A textual description of resultCode (see Result Codes).

Name	Description
invoiceStatus	<p>Status of the invoice:</p> <ul style="list-style-type: none"> • 0 - Pending usage (when sent from SEQR) • ISSUED - Invoice is issued, and waiting for payment • PAID Invoice is paid • PARTIALLY_PAID - Invoice is partially paid • PENDING_ISSUER_ACKNOWLEDGE - Payment is updated and waiting for issuer acknowledgement • CANCELED - Invoice is canceled • FAILED - Invoice payment has failed • RESERVED - The invoice amount is reserved
customerTokens	List of customer tokens relevant for this payment (see separate chapter)
deliveryAddress	If the payment should be delivered automatically, this contains the delivery address to deliver to
receipt	Receipt of the payment, if the status is PAID

5.3.4. getPaymentStatus SOAP examples

Example 5.5. An example of a getPaymentStatus SOAP request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:ext="http://external.interfaces.ers.seamless.com/">
  <soapenv:Header/>
  <soapenv:Body>
    <ext:getPaymentStatus>
      <context>
        <channel>extWS</channel>
        <clientComment>comment</clientComment>
        <clientId>testClient</clientId>
        <clientReference>12345</clientReference>
        <clientRequestTimeout>0</clientRequestTimeout>
        <initiatorPrincipalId>
          <id>87e791f9e24148a6892c52aa85bb0331</id>
          <type>TERMINALID</type>
        </initiatorPrincipalId>
        <password>123456</password>
      </context>
      <invoiceVersion>0</invoiceVersion>
      <invoiceReference>123123</invoiceReference>
    </ext:getPaymentStatus>
  </soapenv:Body>
</soapenv:Envelope>
```

Example 5.6. An example of a getPaymentStatus SOAP response

```

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns2:getPaymentStatusResponse xmlns:ns2="http://
external.interfaces.ers.seamless.com/">
      <return>
        <resultCode>0</resultCode>
        <resultDescription>SUCCESS</resultDescription>
        <status>ISSUED</status>
      </return>
    </ns2:getPaymentStatusResponse>
  </soap:Body>
</soap:Envelope>

```

5.4. cancelInvoice

5.4.1. cancelInvoice Description

Cancels an unpaid invoice.

5.4.2. cancelInvoice request fields

Table 5.8. cancelInvoice request fields

Name	Description
context	The ClientContext object (see Implementation Notes).
invoiceReference	Reference of the invoice to be cancelled.

5.4.3. cancelInvoice response fields

Table 5.9. cancelInvoice response fields

Name	Description
ersReference	Not used by this method (will be null after this method).
resultCode	Request result code (see Result Codes).
resultDescription	A textual description of resultCode (see Result Codes).

5.4.4. cancelInvoice SOAP examples

Example 5.7. An example of a cancelInvoice SOAP request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:ext="http://external.interfaces.ers.seamless.com/">
  <soapenv:Header/>
  <soapenv:Body>
    <ext:cancelInvoice>
      <context>
        <channel>extWS</channel>
        <clientComment>comment</clientComment>
        <clientId>testClient</clientId>
        <clientReference>12345</clientReference>
        <clientRequestTimeout>0</clientRequestTimeout>
        <initiatorPrincipalId>
          <id>87e791f9e24148a6892c52aa85bb0331</id>
          <type>TERMINALID</type>
        </initiatorPrincipalId>
        <password>123456</password>
      </context>
      <invoiceReference>123123</invoiceReference>
    </ext:cancelInvoice>
  </soapenv:Body>
</soapenv:Envelope>
```

Example 5.8. An example of a cancelInvoice SOAP response

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns2:cancelInvoiceResponse xmlns:ns2="http://
external.interfaces.ers.seamless.com/">
      <return>
        <resultCode>0</resultCode>
        <resultDescription>SUCCESS</resultDescription>
      </return>
    </ns2:cancelInvoiceResponse>
  </soap:Body>
</soap:Envelope>
```

5.5. registerTerminal

5.5.1. registerTerminal Description

Registers a new terminal in the SEQR service.

5.5.2. registerTerminal request fields

Table 5.10. registerTerminal request fields

Name	Description
context	The ClientContext object (see Implementation Notes).
externalTerminalId	The identifier of the terminal in the client system, e.g. "Store 111/Till 4"
password	Password for future communications with the SEQR service.
name	The name to appear on the buyer's mobile device, e.g. "My Restaurant, cash register 2"

5.5.3. registerTerminal response fields

Table 5.11. registerTerminal response fields

Name	Description
ersReference	Not used by this method (will be null after this method).
resultCode	The result code for this request.
resultDescription	Request result code (see Result Codes)
terminalId	The newly generated unique identifier for this terminal. This identifier should be used in future communications of this terminal towards the SEQR service.

5.5.4. registerTerminal SOAP examples

Example 5.9. An example of a registerTerminal SOAP request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:ext="http://external.interfaces.ers.seamless.com/">
  <soapenv:Header/>
  <soapenv:Body>
    <ext:registerTerminal>
      <context>
        <channel>WS</channel>
        <clientComment>comment</clientComment>
        <clientId>testClient</clientId>
        <clientReference>12345</clientReference>
        <clientRequestTimeout>0</clientRequestTimeout>
        <initiatorPrincipalId>
          <id>fredellsfisk</id>
          <type>RESELLERUSER</type>
          <userId>9900</userId>
        </initiatorPrincipalId>
        <password>2009</password>
      </context>
      <externalTerminalId>Shop 1/POS 2</externalTerminalId>
      <password>secret</password>
      <name>My Shop's Name</name>
    </ext:registerTerminal>
  </soapenv:Body>
</soapenv:Envelope>
```

Example 5.10. An example of a registerTerminal SOAP response

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns2:registerTerminalResponse xmlns:ns2="http://
external.interfaces.ers.seamless.com/">
      <return>
        <resultCode>0</resultCode>
        <resultDescription>SUCCESS</resultDescription>
        <terminalId>87e791f9e24148a6892c52aa85bb0331</terminalId>
      </return>
    </ns2:registerTerminalResponse>
  </soap:Body>
</soap:Envelope>
```

5.6. unregisterTerminal

5.6.1. unregisterTerminal Description

Unregisters an already registered terminal.

5.6.2. unregisterTerminal request fields

Table 5.12. unregisterTerminal request fields

Name	Description
context	The ClientContext object (see Implementation Notes).
TerminalId	The SEQR ID of the terminal to be unregistered.

5.6.3. unregisterTerminal response fields

Table 5.13. unregisterTerminal response fields

Name	Description
ersReference	Not used by this method (will be null after this method).
resultCode	The result code for this request.
resultDescription	Request result code (see Result Codes)

5.6.4. unregisterTerminal SOAP examples

Example 5.11. An example of a unregisterTerminal SOAP request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:ext="http://external.interfaces.ers.seamless.com/">
  <soapenv:Header/>
  <soapenv:Body>
    <ext:unregisterTerminal>
      <context>
        <channel>WS</channel>
        <clientComment>comment</clientComment>
        <clientId>testClient</clientId>
        <clientReference>12345</clientReference>
        <clientRequestTimeout>0</clientRequestTimeout>
        <initiatorPrincipalId>
          <id>87e791f9e24148a6892c52aa85bb0331</id>
          <type>TERMINALID</type>
        </initiatorPrincipalId>
        <password>secret</password>
      </context>
    </ext:unregisterTerminal>
  </soapenv:Body>
</soapenv:Envelope>
```

Example 5.12. An example of a unregisterTerminal SOAP response

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns2:unregisterTerminalResponse xmlns:ns2="http://
external.interfaces.ers.seamless.com/">
      <return>
        <resultCode>0</resultCode>
        <resultDescription>SUCCESS</resultDescription>
      </return>
    </ns2:unregisterTerminalResponse>
  </soap:Body>
</soap:Envelope>
```

5.7. assignSeqrId

5.7.1. assignSeqrId Description

Assigns a SEQR ID to a terminal.

5.7.2. assignSeqrId request fields

Table 5.14. assignSeqrId request fields

Name	Description
context	The ClientContext object (see Implementation Notes).
seqrId	The SEQR ID of the terminal.

5.7.3. assignSeqrId response fields

Table 5.15. assignSeqrId response fields

Name	Description
ersReference	Not used by this method (will be null after this method).
resultCode	The result code for this request.
resultDescription	Request result code (see Result Codes)

5.7.4. assignSeqrId SOAP examples

Example 5.13. An example of a assignSeqrId SOAP request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:ext="http://external.interfaces.ers.seamless.com/">
  <soapenv:Header/>
  <soapenv:Body>
    <ext:assignSeqrId>
      <context>
        <channel>WS</channel>
        <clientComment>comment</clientComment>
        <clientId>testClient</clientId>
        <clientReference>12345</clientReference>
        <clientRequestTimeout>0</clientRequestTimeout>
        <initiatorPrincipalId>
          <id>87e791f9e24148a6892c52aa85bb0331</id>
          <type>TERMINALID</type>
        </initiatorPrincipalId>
        <password>secret</password>
      </context>
      <seqrId>ABC123456</seqrId>
    </ext:assignSeqrId>
  </soapenv:Body>
</soapenv:Envelope>
```

Example 5.14. An example of a assignSeqrId SOAP response

```

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns2:assignSeqrIdResponse xmlns:ns2="http://
external.interfaces.ers.seamless.com/">
      <return>
        <resultCode>0</resultCode>
        <resultDescription>SUCCESS</resultDescription>
      </return>
    </ns2:assignSeqrIdResponse>
  </soap:Body>
</soap:Envelope>

```

5.8. commitReservation

5.8.1. commitReservation Description

Commits the payment for an invoice that is in RESERVED state.

If getPaymentStatus returns an invoice state of RESERVED, the merchant will have to call commitReservation to complete the payments.

For in-store payments this is usually done in real-time when closing the receipt/customer, eg within a few seconds. For on-line and shipping payments, this can often be called several days later.

If the merchant does not call commitReservation withing a certain time, the reservation will time out and no payment will be made to the merchant. The exact timeout depends on the type of merchant and payment scenarion. For in-store payments this is usually a short timeout (minutes).

5.8.2. commitReservation request fields

Table 5.16. commitReservation request fields

Name	Description
context	The ClientContext object (see Implementation Notes).
invoiceReference	Reference of the invoice that is reserved.

5.8.3. commitReservation response fields

Table 5.17. commitReservation response fields

Name	Description
resultCode	The result code for this request.
resultDescription	Request result code (see Result Codes)

5.9. submitPaymentReceipt

5.9.1. submitPaymentReceipt Description

Sends a receipt for a payment or refund which is shown on the buyer's mobile device.

NOTE: submitPaymentReceipt should be sent as soon as possible after the payment is completed/committed.

submitPaymentReceipt can be called multiple times for the same payment to add multiple types of receipt document (indicated by receiptType), eg Payment receipt (slip), warranty, vouchers etc

5.9.2. submitPaymentReceipt request fields

Table 5.18. submitPaymentReceipt request fields

Name	Description
context	The ClientContext object (see Implementation Notes).
ersReference	Reference of the payment for which the receipt is applicable.
receiptDocument	Receipt document, containing the full details of the receipt. NOTES Preferably in ARTS Receipt XML format (see Implementation Notes).

5.9.3. submitPaymentReceipt response fields

Table 5.19. submitPaymentReceipt response fields

Name	Description
ersReference	Not used by this method (will be null after this method).
resultCode	The result code for this request.
resultDescription	Request result code (see Result Codes)

5.9.4. submitPaymentReceipt SOAP examples

Example 5.15. An example of a submitPaymentReceipt SOAP request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:ext="http://external.interfaces.ers.seamless.com/">
  <soapenv:Header/>
  <soapenv:Body>
    <ext:submitPaymentReceipt>
      <context>
        <channel>WS</channel>
        <clientComment>comment</clientComment>
        <clientId>testClient</clientId>
        <clientReference>12345</clientReference>
        <clientRequestTimeout>0</clientRequestTimeout>
        <initiatorPrincipalId>
          <id>87e791f9e24148a6892c52aa85bb0331</id>
          <type>TERMINALID</type>
        </initiatorPrincipalId>
        <password>secret</password>
      </context>
      <ersReference>20120501000000000000000001</ersReference>
      <receiptDocument>
        <mimeType>plain/xml</mimeType>
        <receiptData>Cjw/eGlsIHZlcnNpb249IjEuMCIgZW5jb2Rpbmc9IlVURi04Ij8+Cjx</
receiptData>
      </receiptDocument>
    </ext:submitPaymentReceipt>
  </soapenv:Body>
</soapenv:Envelope>
```

Example 5.16. An example of a submitPaymentReceipt SOAP response

```

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns2:submitPaymentReceiptResponse xmlns:ns2="http://
external.interfaces.ers.seamless.com/">
      <return>
        <ersReference>20120501000000000000000002</ersReference>
        <resultCode>0</resultCode>
        <resultDescription>SUCCESS</resultDescription>
      </return>
    </ns2:submitPaymentReceiptResponse>
  </soap:Body>
</soap:Envelope>

```

5.10. refundPayment

5.10.1. refundPayment Description

Refunds a previous payment. After this method, submitPaymentReceipt is used to upload a receipt.

5.10.2. refundPayment request fields

Table 5.20. refundPayment request fields

Name	Description
context	The ClientContext object (see Implementation Notes).
ersReference	Reference of the payment to be refunded
invoice	Invoice data, which contains the amount and other invoice information after products has been removed from the original invoice.

5.10.3. refundPayment response fields

Table 5.21. refundPayment response fields

Name	Description
ersReference	The reference of the refund.
resultCode	The result code for this request.
resultDescription	Request result code (see Result Codes)

5.10.4. refundPayment SOAP examples

Example 5.17. An example of a refundPayment SOAP request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:ext="http://external.interfaces.ers.seamless.com/">
  <soapenv:Header/>
  <soapenv:Body>
    <ext:refundPayment>
      <context>
        <channel>WS</channel>
        <clientComment>comment</clientComment>
        <clientId>testClient</clientId>
        <clientReference>12345</clientReference>
        <clientRequestTimeout>0</clientRequestTimeout>
        <initiatorPrincipalId>
          <id>87e791f9e24148a6892c52aa85bb0331</id>
          <type>TERMINALID</type>
        </initiatorPrincipalId>
        <password>secret</password>
      </context>
      <ersReference>20120501000000000000000001</ersReference>
      <invoice>
        <title>Refund</title>
        <cashierId>Bob</cashierId>
        <totalAmount>
          <currency>SEK</currency>
          <value>10.22</value>
        </totalAmount>
      </invoice>
    </ext:refundPayment>
  </soapenv:Body>
</soapenv:Envelope>
```

Example 5.18. An example of a refundPayment SOAP response

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns2:refundPaymentResponse xmlns:ns2="http://
external.interfaces.ers.seamless.com/">
      <return>
        <ersReference>20120501000000000000000002</ersReference>
        <resultCode>0</resultCode>
        <resultDescription>SUCCESS</resultDescription>
      </return>
    </ns2:refundPaymentResponse>
  </soap:Body>
</soap:Envelope>
```

5.11. markTransactionPeriod

5.11.1. markTransactionPeriod Description

Marks the end of one and the beginning of a new transaction period; used in reporting.

5.11.2. markTransactionPeriod request fields

Table 5.22. markTransactionPeriod request fields

Name	Description
context	The ClientContext object (see Implementation Notes).
parameters	Optional parameters that can be used in processing the request.

5.11.3. markTransactionPeriod response fields

Table 5.23. markTransactionPeriod response fields

Name	Description
ersReference	The reference for this operation
resultCode	The result code for this request.
resultDescription	Request result code (see Result Codes)

5.11.4. markTransactionPeriod SOAP examples

Example 5.19. An example of a markTransactionPeriod SOAP request, for **per shop reconciliation**

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:ext="http://external.interfaces.ers.seamless.com/">
  <soapenv:Header/>
  <soapenv:Body>
    <ext:markTransactionPeriod>
      <context>
        <channel>WS</channel>
        <clientComment>comment</clientComment>
        <clientId>testClient</clientId>
        <clientReference>12345</clientReference>
        <clientRequestTimeout>0</clientRequestTimeout>
        <initiatorPrincipalId>
          <id>fredellsfisk</id>
          <type>RESELERUSER</type>
          <userId>9900</id>
        </initiatorPrincipalId>
        <password>secret</password>
      </context>
    </ext:markTransactionPeriod>
  </soapenv:Body>
</soapenv:Envelope>
```

Example 5.20. An example of a markTransactionPeriod SOAP response, for **per shop reconciliation**

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns2:markTransactionPeriodResponse xmlns:ns2="http://
external.interfaces.ers.seamless.com/">
      <return>
        <resultCode>0</resultCode>
        <resultDescription>SUCCESS</resultDescription>
        <transactionPeriodId>2012053119463656301000002</transactionPeriodId>
      </return>
    </ns2:markTransactionPeriodResponse>
  </soap:Body>
</soap:Envelope>
```

Example 5.21. An example of a markTransactionPeriod SOAP request, for **per terminal reconciliation**

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:ext="http://external.interfaces.ers.seamless.com/">
  <soapenv:Header/>
  <soapenv:Body>
    <ext:markTransactionPeriod>
      <context>
        <channel>WS</channel>
        <clientComment>comment</clientComment>
        <clientId>testClient</clientId>
        <clientReference>12345</clientReference>
        <clientRequestTimeout>0</clientRequestTimeout>
        <initiatorPrincipalId>
          <id>fredellsfisk</id>
          <type>RESELERUSER</type>
          <userId>9900</id>
        </initiatorPrincipalId>
        <password>secret</password>
      </context>
      <entry>
        <key>TERMINALID</key>
        <value>2469e0bf14214797880cafb0edalb535</value>
      </entry>
    </ext:markTransactionPeriod>
  </soapenv:Body>
</soapenv:Envelope>
```

Example 5.22. An example of a markTransactionPeriod SOAP response, for **per terminal reconciliation**

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns2:markTransactionPeriodResponse xmlns:ns2="http://
external.interfaces.ers.seamless.com/">
      <return>
        <resultCode>0</resultCode>
        <resultDescription>SUCCESS</resultDescription>
        <transactionPeriodId>2012053119463656301000002</transactionPeriodId>
      </return>
    </ns2:markTransactionPeriodResponse>
  </soap:Body>
</soap:Envelope>
```

5.12. executeReport

5.12.1. executeReport Description

Executes a report on the SEQR service.

5.12.2. executeReport request fields

Table 5.24. executeReport request fields

Name	Description
context	The ClientContext object (see Implementation Notes).
reportId	The identifier of the report that should be executed / produced.
language	The report language (null if the default language is to be used).
parameters	Optional parameters that can be used in processing the request.

5.12.3. executeReport response fields

Table 5.25. executeReport response fields

Name	Description
ersReference	The reference for this operation
resultCode	The result code for this request.
resultDescription	Request result code (see Result Codes)
report	The produced / executed report, in binary and plain text form, if available.

5.12.4. executeReport SOAP examples

Example 5.23. An example of a executeReport SOAP request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:ext="http://external.interfaces.ers.seamless.com/">
  <soapenv:Header/>
  <soapenv:Body>
    <ext:executeReport>
      <context>
        <channel>WS</channel>
        <clientComment>comment</clientComment>
        <clientId>testClient</clientId>
        <clientReference>12345</clientReference>
        <clientRequestTimeout>0</clientRequestTimeout>
        <initiatorPrincipalId>
          <id>fredellsfisk</id>
          <type>RESELLERUSER</type>
          <user>9900</user>
        </initiatorPrincipalId>
        <password>secret</password>
      </context>
      <reportId>SOME_REPORT</reportId>
    </ext:executeReport>
  </soapenv:Body>
</soapenv:Envelope>
```

Example 5.24. An example of a executeReport SOAP response

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns2:executeReportResponse xmlns:ns2="http://
external.interfaces.ers.seamless.com/">
      <return>
        <resultCode>0</resultCode>
        <resultDescription>SUCCESS</resultDescription>
        <report>
          <content>MTE1Nzky</content>
          <contentString>115792</contentString>
          <mimeType>text/plain</mimeType>
          <title>Number of transfers today</title>
        </report>
      </return>
    </ns2:executeReportResponse>
  </soap:Body>
</soap:Envelope>
```

5.13. Result codes

Result codes returned by each interface methods.

Table 5.26. Result codes

code	Result Description	Relevant responses		updateInvoice	getPaymentStatus	cancelInvoice	confirmCustomerRegistration	registerTerminal
0	SUCCESS	Y	sendInvoice	Y	Y	Y	Y	Y
11	REJECTED_AMOUNT	Y		Y				
20	AUTHENTICATION_FAILED	Y		Y	Y	Y	Y	Y
29	INVALID_INITIATOR_PRINCIPAL_ID	Y		Y	Y	Y	Y	Y
33	INVALID_INITIATOR_PRINCIPAL_STATE	Y		Y	Y	Y	Y	Y
37	INITIATOR_PRINCIPAL_NOT_FOUND	Y		Y	Y	Y	Y	Y
48	PAYMENT_IN_PROGRESS			Y		Y		
49	INVALID_INVOICE_DATA	Y		Y				
50	CANNOT_CANCEL_PAID_INVOICE					Y		
51	CANNOT_CANCEL_INVOICE_IN_PROGRESS					Y		
53	INVALID_SEQR_ID							
90	SYSTEM_ERROR	Y		Y	Y	Y	Y	Y
91	UNSUPPORTED_OPERATION	Y		Y	Y	Y	Y	Y
93	SYSTEM_BUSY	Y		Y	Y	Y	Y	Y
94	SERVICE_UNAVAILABLE	Y		Y	Y	Y	Y	Y

Table 5.27. Result codes (continued)

Result code	Result Description	unregisterTerminal	assignSeqId	submitPaymentReceipt	refundPayment
0	SUCCESS	Y	Y	Y	Y
11	REJECTED_AMOUNT				Y
20	AUTHENTICATION_FAILED	Y	Y	Y	Y
29	INVALID_INITIATOR_PRINCIPAL_ID	Y	Y	Y	Y
33	INVALID_INITIATOR_PRINCIPAL_STATE	Y	Y	Y	Y
37	INITIATOR_PRINCIPAL_NOT_FOUND	Y	Y	Y	Y
48	PAYMENT_IN_PROGRESS				Y
49	INVALID_INVOICE_DATA				
50	CANNOT_CANCEL_PAID_INVOICE				
51	CANNOT_CANCEL_INVOICE_IN_PROGRESS				
53	INVALID_SEQR_ID		Y		
90	SYSTEM_ERROR	Y	Y	Y	Y
91	UNSUPPORTED_OPERATION	Y	Y	Y	Y
93	SYSTEM_BUSY	Y	Y	Y	Y
94	SERVICE_UNAVAILABLE	Y	Y	Y	Y

Chapter 6. Developing clients in Java

6.1. Overview

The simplest way to develop a Java client is to use the JAR files provided in the API bundle. They contain the annotated WebService interface that can be implemented in a Java program.

An alternative is to compile the WSDL files included in the API bundle, which might be preferable in some environments.

The complete source code of the Java interface is included in the API bundle. There are also a few code samples in `com.seamless.ers.interfaces.external.samples` package. All samples uses the Apache CXF (<http://cxf.apache.org/>) webservice library

6.2. Samples

6.2.1. MPaymentFlowSample.java

Sample code for processing a simple Mobile Payment by cashregister.

```
package com.seamless.ers.interfaces.external.samples;

import com.seamless.ers.interfaces.external.data.Amount;
import com.seamless.ers.interfaces.external.data.ClientContext;
import com.seamless.ers.interfaces.external.data.Invoice;
import com.seamless.ers.interfaces.external.data.PrincipalId;
import com.seamless.ers.interfaces.external.response.ERSWSPaymentStatusResponse;
import
    com.seamless.ers.interfaces.external.response.ERSWSPaymentStatusResponse.InvoiceStatus;
import com.seamless.ers.interfaces.external.response.ERSWSResultCodes;
import com.seamless.ers.interfaces.external.response.ERSWSSendInvoiceResponse;

/**
 * A sample ERS client that receives a SEQR payment.
 *
 * If support for loyalty programs/coupons are needed, use MPaymentFlowSample2
 */
public class MPaymentFlowSample
{
    /**
     * @param args
     * @throws Exception
     */
    public static void main(String[] args) throws Exception
    {
        // Connect to ERS.
        SampleWSClient ersClient = new SampleWSClient(
            "http://localhost:8913/extclientproxy/service");

        // Define context.
        ClientContext context = new ClientContext();
        // from which component
        context.setClientId("ers-extclientproxy");
        context.setClientComment("test purchase");
        context.setClientReference(System.currentTimeMillis() + "");
        context.setPassword("2009");
        context.setChannel("WEBSERVICE");

        // Define sender principal.
        PrincipalId paymentPrincipalId = new PrincipalId();
        paymentPrincipalId.setId("0460000300002010");
        paymentPrincipalId.setType("TERMINALID");
        context.setInitiatorPrincipalId(paymentPrincipalId);
        Amount totalPrice = new Amount(10, "SEK");

        Invoice invoice = new Invoice();
```

```

invoice.setCashierId("Bob");
invoice.setTotalAmount(totalPrice);
invoice.setTitle("sample title");

// issue the invoice
ERSWSSendInvoiceResponse response = ersClient.getExternalClientService().
    sendInvoice(context, invoice, null);

if (response.getResultCode() != ERSWSResultCodes.SUCCESS)
{
    System.out.println(String.format("Error issuing an invoice: %s (%d)",
        response.getResultDescription(), response.getResultCode()));
    return;
}
System.out.println("Invoice issued, reference: " + response.getInvoiceReference());

ERSWSPaymentStatusResponse paymentStatus = null;

// wait until the invoice is paid or canceled/failed
while (true)
{
    Thread.sleep(1000);
    System.out.println("Waiting for invoice payment...");

    paymentStatus = ersClient.getExternalClientService().getPaymentStatus(
        context,
        response.getInvoiceReference(),
        paymentStatus == null ? 0 : paymentStatus.getVersion());
    if (paymentStatus.getResultCode() != ERSWSResultCodes.SUCCESS)
    {
        System.out.println(String.format(
            "Error while waiting for invoice payment: %s (%d)",
            paymentStatus.getResultDescription(),
            paymentStatus.getResultCode()));
        return;
    }
    if (paymentStatus.getStatus() != InvoiceStatus.ISSUED)
        break;
}

// check what happened to the invoice
switch (paymentStatus.getStatus())
{
    case PAID:
        System.out.println("Invoice was paid.");
        break;

    default:
        System.out.println("Invoice was not paid, state is : "
            + paymentStatus.getStatus().name());
}
}
}

```

6.2.2. MPaymentFlowWithLoyaltySample.java

Sample code for processing a Mobile Payment by cashregister that supports loyalty programs.

```

package com.seamless.ers.interfaces.external.samples;

import java.math.BigDecimal;
import java.util.ArrayList;
import java.util.List;

import com.seamless.ers.interfaces.external.data.Amount;
import com.seamless.ers.interfaces.external.data.ClientContext;
import com.seamless.ers.interfaces.external.data.CustomerToken;
import com.seamless.ers.interfaces.external.data.Invoice;
import com.seamless.ers.interfaces.external.data.PrincipalId;
import com.seamless.ers.interfaces.external.response.ERSWSPaymentStatusResponse;

```

```
import
    com.seamless.ers.interfaces.external.response.ERSWSPaymentStatusResponse.InvoiceStatus;
import com.seamless.ers.interfaces.external.response.ERSWSResponse;
import com.seamless.ers.interfaces.external.response.ERSWSResultCodes;
import com.seamless.ers.interfaces.external.response.ERSWSSendInvoiceResponse;

/**
 * A sample ERS client that receives a SEQR payment.
 *
 * This version supports coupons/membership discounts/loyalty programs.
 */
public class MPaymentFlowWithLoyaltySample
{
    /**
     * @param args
     * @throws Exception
     */
    public static void main(String[] args) throws Exception
    {
        // Connect to ERS.
        SampleWSCClient ersClient = new SampleWSCClient(
            "http://localhost:8913/extclientproxy/service");

        // Define context.
        ClientContext context = new ClientContext();
        // from which component
        context.setClientId("ers-extclientproxy");
        context.setClientComment("test purchase");
        context.setClientReference(System.currentTimeMillis() + "");
        context.setPassword("2009");
        context.setChannel("WEBSERVICE");

        // Define sender principal.
        PrincipalId paymentPrincipalId = new PrincipalId();
        paymentPrincipalId.setId("0460000100001002");
        paymentPrincipalId.setType("TERMINALID");
        context.setInitiatorPrincipalId(paymentPrincipalId);

        Invoice invoice = createInvoice();

        // issue the invoice
        ERSWSSendInvoiceResponse response = ersClient.getExternalClientService().
            sendInvoice(context, invoice, null);

        if (response.getResultCode() != ERSWSResultCodes.SUCCESS)
        {
            System.out.println(String.format("Error issuing an invoice: %s (%d)",
                response.getResultDescription(), response.getResultCode()));
            return;
        }
        System.out.println("Invoice issued, reference: " + response.getInvoiceReference());

        ERSWSPaymentStatusResponse paymentStatus = null;
        List<CustomerToken> loyaltyPrograms = new ArrayList<CustomerToken>();

        // wait until the invoice is paid or canceled/failed
        while (true)
        {
            Thread.sleep(1000);
            System.out.println("Waiting for invoice payment...");

            paymentStatus = ersClient.getExternalClientService().getPaymentStatus(
                context,
                response.getInvoiceReference(),
                paymentStatus == null ? 0 : paymentStatus.getVersion());
            if (paymentStatus.getResultCode() != ERSWSResultCodes.SUCCESS)
            {
                System.out.println(String.format(
                    "Error while waiting for invoice payment: %s (%d)",
                    paymentStatus.getResultDescription(),
                    paymentStatus.getResultCode()));
                return;
            }
            if (paymentStatus.getStatus() != InvoiceStatus.ISSUED
                && paymentStatus.getStatus() != InvoiceStatus.PENDING_ISSUER_ACKNOWLEDGE)
        }
    }
}
```



```

        break;

        // check if there are any new loyalty programs to be applied to the
        // current purchase
        for (CustomerToken token : paymentStatus.getCustomerTokens())
            // is the loyalty program already applied?
            if (!loyaltyPrograms.contains(token))
            {
                // add the program to the applied programs
                loyaltyPrograms.add(token);

                // created the new invoice after applying the program
                Invoice discountedInvoice = applyLoyaltyProgramToInvoice(token, invoice);

                // send the updated invoice to ERS
                ERSWSResponse updateInvoiceResponse = ersClient.getExternalClientService().
                    updateInvoice(
                        context,
                        response.getInvoiceReference(),
                        discountedInvoice,
                        null);

                // make sure the update of invoice was successful
                if (updateInvoiceResponse.getResultCode() != ERSWSResultCodes.SUCCESS)
                {
                    System.out.println(String.format(
                        "Error updating the invoice: %s (%d)",
                        paymentStatus.getResultDescription(),
                        paymentStatus.getResultCode()));
                    return;
                }
            }
    }

    // check what happened to the invoice
    switch (paymentStatus.getStatus())
    {
        case PAID:
            System.out.println("Invoice was paid.");
            break;

        default:
            System.out.println("Invoice was not paid, state is : "
                + paymentStatus.getStatus().name());
    }
}

/**
 * Applies the specified loyalty program to the invoice, possibly changing
 * the amount to pay
 *
 * @param program
 * @param invoice
 * @return
 */
private static Invoice applyLoyaltyProgramToInvoice(CustomerToken program, Invoice
invoice)
{
    // re-calculate total invoice price
    invoice.setTotalAmount(new Amount(new BigDecimal("9.00"), "SEK"));

    return invoice;
}

/**
 * Creates the invoice to be paid
 *
 * @return
 */
private static Invoice createInvoice()
{
    Amount totalPrice = new Amount(new BigDecimal("10.00"), "SEK");
    Invoice invoice = new Invoice();
    invoice.setCashierId("Bob");
    invoice.setTotalAmount(totalPrice);
    invoice.setTitle("sample title");
}

```

```
return invoice;  
}  
}
```

Chapter 7. Developing clients in C/C++

7.1. Overview

There are many possibilities when it comes to developing clients in C or C++. A simple and flexible solution is to use a SOAP library to compile the WSDL files and generate C code that can then simply be called as functions from within the client.

When testing this interface we have used gSOAP which is an open source (GNU) library for using SOAP from C/C++.

Chapter 8. Reconciliation reports

Reconciliation of cash register system against SEQR service is done automatically as part of cash register integration. List of reports are provided to the integrators to support their current models of reconciliation. The reports are generate an XML format document. This section describes reconciliation reports

8.1. Reconciliation reports

There are list of reports for reconciliation

Table 8.1. Reconciliation reports

Report ID	Report name	Description
STD_RECON_001	Merchant Transactions Summary	Transaction summary for a shop representing number of transactions and summary amount done for the period
STD_RECON_002	Merchant Transactions Summary, Terminal by Terminal	Transaction summary for a shop group by cash register showing number of transactions and summary amount done from each cash register for the period
STD_RECON_003	Merchant Transactions Details	Transaction details for a shop showing ersReference, cashier, cash register and amount for the period
STD_RECON_004	Merchant Transactions Summary, Cashier by Cashier	Transaction summary for a shop group by cashier showing number of transactions and summary amount done from each cashier for the period
STD_RECON_006	Terminal Transactions Summary	Similar to "STD_RECON_002", but the report is used in per terminal reconciliation process.
STD_RECON_007	Terminal Transactions Details	Similar to "STD_RECON_003", but the report is used in per terminal reconciliation process

8.2. Samples

8.2.1. STD_RECON_001 SOAP examples

Example 8.1. An example of executing report STD_RECON_001 SOAP request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:ext="http://external.interfaces.ers.seamless.com/">
  <soapenv:Header/>
  <soapenv:Body>
    <ext:executeReport>
      <context>
        <channel>WS</channel>
        <clientComment>comment</clientComment>
        <clientId>testClient</clientId>
        <clientReference>12345</clientReference>
        <clientRequestTimeout>0</clientRequestTimeout>
        <initiatorPrincipalId>
          <id>hml</id>
          <type>RESELLERUSER</type>
          <userId>9900</userId>
        </initiatorPrincipalId>
        <password>2009</password>
      </context>
      <reportId>Reconciliation/Shop.xml</reportId>
      <language>en</language>
      <parameters>
        <parameter>
          <entry>
            <key>transactionPeriodId</key>
            <value>2012071711291546101000028</value>
          </entry>
        </parameter>
      </parameters>
    </ext:executeReport>
  </soapenv:Body>
</soapenv:Envelope>
```

Example 8.2. An example of executing report STD_RECON_001 SOAP response

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns2:executeReportResponse xmlns:ns2="http://
external.interfaces.ers.seamless.com/">
      <return>
        <resultCode>0</resultCode>
        <resultDescription>SUCCESS</resultDescription>
        <report>
          <title/>
          <mimeType>text/html</mimeType>
          <content>Cjw/...=</content>
          <contentString><![CDATA[<?xml version="1.0" encoding="UTF-8"?>
<ResellerTransactionSummary>
<Meta>
<ResellerId>hml</ResellerId>
<TransactionPeriodId>2012071711291546101000028</TransactionPeriodId>
<GeneratedAt>2012-07-18 15:19</GeneratedAt>
</Meta>
<Row>
<SalesCount>202</SalesCount>
<SalesTotal>5476.52000</SalesTotal>
</Row>
<ResellerTransactionSummary>]]</contentString>
        </report>
      </return>
    </ns2:executeReportResponse>
  </soap:Body>
</soap:Envelope>
```

8.2.2. STD_RECON_002 SOAP examples

Example 8.3. An example of executing report STD_RECON_002 SOAP request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:ext="http://external.interfaces.ers.seamless.com/">
  <soapenv:Header/>
  <soapenv:Body>
    <ext:executeReport>
      <context>
        <channel>WS</channel>
        <clientComment>comment</clientComment>
        <clientId>testClient</clientId>
        <clientReference>12345</clientReference>
        <clientRequestTimeout>0</clientRequestTimeout>
        <initiatorPrincipalId>
          <id>hml</id>
          <type>RESELLERUSER</type>
          <userId>9900</userId>
        </initiatorPrincipalId>
        <password>2009</password>
      </context>
      <reportId>Reconciliation/STD_RECON_002.xml</reportId>
      <language>en</language>
      <parameters>
        <parameter>
          <entry>
            <key>transactionPeriodId</key>
            <value>2012071711291546101000028</value>
          </entry>
        </parameter>
      </parameters>
    </ext:executeReport>
  </soapenv:Body>
</soapenv:Envelope>
```

Example 8.4. An example of executing report STD_RECON_002 SOAP response

```

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns2:executeReportResponse xmlns:ns2="http://
external.interfaces.ers.seamless.com/">
      <return>
        <resultCode>0</resultCode>
        <resultDescription>SUCCESS</resultDescription>
        <report>
          <title>STD_RECON_002_2012071711291546101000028.xml</title>
          <mimeType>text/html</mimeType>
          <content>PD94bWwgdmVyc2...==</content>
          <contentString><![CDATA[<?xml version="1.0" encoding="UTF-8"?>
<TerminalTransactionSummary>
<Meta>
<ResellerId>hml</ResellerId>
<TransactionPeriodId>2012071711291546101000028</TransactionPeriodId>
<GeneratedAt>2012-07-18 15:20</GeneratedAt>
</Meta>

<Row>
<TerminalId>0232bad16bbd46d7a64e850da2d739ac</TerminalId>
<SalesCount>1</SalesCount>
<SalesTotal>20.00000</SalesTotal>
</Row>

<Row>
<TerminalId>06d9a15fc6d04a89b4a6ea7744e5dfed</TerminalId>
<SalesCount>1</SalesCount>
<SalesTotal>41.21000</SalesTotal>
</Row>

</TerminalTransactionSummary>]]</contentString>
        </report>
      </return>
    </ns2:executeReportResponse>
  </soap:Body>
</soap:Envelope>

```

8.2.3. STD_RECON_003 SOAP examples

Example 8.5. An example of executing report STD_RECON_003 SOAP request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:ext="http://external.interfaces.ers.seamless.com/">
  <soapenv:Header/>
  <soapenv:Body>
    <ext:executeReport>
      <context>
        <channel>WS</channel>
        <clientComment>comment</clientComment>
        <clientId>testClient</clientId>
        <clientReference>12345</clientReference>
        <clientRequestTimeout>0</clientRequestTimeout>
        <initiatorPrincipalId>
          <id>hml</id>
          <type>RESELLERUSER</type>
          <userId>9900</userId>
        </initiatorPrincipalId>
        <password>2009</password>
      </context>
      <reportId>Reconciliation/STD_RECON_003.xml</reportId>
      <language>en</language>
      <parameters>
        <parameter>
          <entry>
            <key>transactionPeriodId</key>
            <value>2012071711291546101000028</value>
          </entry>
        </parameter>
      </parameters>
    </ext:executeReport>
  </soapenv:Body>
</soapenv:Envelope>
```


Example 8.6. An example of executing report STD_RECON_003 SOAP response

```

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns2:executeReportResponse xmlns:ns2="http://
external.interfaces.ers.seamless.com/">
      <return>
        <resultCode>0</resultCode>
        <resultDescription>SUCCESS</resultDescription>
        <report>
          <title>STD_RECON_003_2012071711291546101000028.xml</title>
          <mimeType>text/html</mimeType>
          <content>PD94bWwgdmVyc2...=</content>
          <contentString><![CDATA[<?xml version="1.0" encoding="UTF-8"?>
<TransactionDetails>
<Meta>
<ResellerId>hml</ResellerId>
<TransactionPeriodId>2012071711291546101000028</TransactionPeriodId>
<GeneratedAt>2012-07-18 15:21</GeneratedAt>
</Meta>

<Row>
<TerminalId>d8e0ff10ea214e5282899aa9697117c1</TerminalId>
<CashierId>bob</CashierId>
<SalesAmount>93.31000</SalesAmount>
<ErsReference>2012060410485453001000101</ErsReference>
</Row>

<Row>
<TerminalId>d8e0ff10ea214e5282899aa9697117c1</TerminalId>
<CashierId>bob</CashierId>
<SalesAmount>30.00000</SalesAmount>
<ErsReference>2012060410502892201000104</ErsReference>
</Row>

<Row>
<TerminalId>d8e0ff10ea214e5282899aa9697117c1</TerminalId>
<CashierId>bob</CashierId>
<SalesAmount>340.00000</SalesAmount>
<ErsReference>2012060411033709001000145</ErsReference>
</Row>

</TransactionDetails>]]></contentString>
        </report>
      </return>
    </ns2:executeReportResponse>
  </soap:Body>
</soap:Envelope>

```

8.2.4. STD_RECON_004 SOAP examples

Example 8.7. An example of executing report STD_RECON_004 SOAP request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:ext="http://external.interfaces.ers.seamless.com/">
  <soapenv:Header/>
  <soapenv:Body>
    <ext:executeReport>
      <context>
        <channel>WS</channel>
        <clientComment>comment</clientComment>
        <clientId>testClient</clientId>
        <clientReference>12345</clientReference>
        <clientRequestTimeout>0</clientRequestTimeout>
        <initiatorPrincipalId>
          <id>hml</id>
          <type>RESELLERUSER</type>
          <userId>9900</userId>
        </initiatorPrincipalId>
        <password>2009</password>
      </context>
      <reportId>Reconciliation/STD_RECON_004.xml</reportId>
      <language>en</language>
      <parameters>
        <parameter>
          <entry>
            <key>transactionPeriodId</key>
            <value>2012071711291546101000028</value>
          </entry>
        </parameter>
      </parameters>
    </ext:executeReport>
  </soapenv:Body>
</soapenv:Envelope>
```

Example 8.8. An example of executing report STD_RECON_004 SOAP response

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns2:executeReportResponse xmlns:ns2="http://
external.interfaces.ers.seamless.com/">
      <return>
        <resultCode>0</resultCode>
        <resultDescription>SUCCESS</resultDescription>
        <report>
          <title>STD_RECON_004_2012071711291546101000028.xml</title>
          <mimeType>text/html</mimeType>
          <content>PD94...=</content>
          <contentString><![CDATA[<?xml version="1.0" encoding="UTF-8"?>
<CashierTransactionSummary>
<Meta>
<ResellerId>hml</ResellerId>
<TransactionPeriodId>2012071711291546101000028</TransactionPeriodId>
<GeneratedAt>2012-07-18 15:22</GeneratedAt>
</Meta>

<Row>
<CashierId>bob</CashierId>
<SalesCount>132</SalesCount>
<SalesTotal>4000.22000</SalesTotal>
</Row>

</CashierTransactionSummary>]]</contentString>
        </report>
      </return>
    </ns2:executeReportResponse>
  </soap:Body>
</soap:Envelope>
```

8.2.5. STD_RECON_006 SOAP examples

Example 8.9. An example of executing report STD_RECON_006 SOAP request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:ext="http://external.interfaces.ers.seamless.com/">
  <soapenv:Header/>
  <soapenv:Body>
    <ext:executeReport>
      <context>
        <channel>WS</channel>
        <clientComment>comment</clientComment>
        <clientId>testClient</clientId>
        <clientReference>12345</clientReference>
        <clientRequestTimeout>0</clientRequestTimeout>
        <initiatorPrincipalId>
          <id>hml</id>
          <type>RESELLERUSER</type>
          <userId>9900</userId>
        </initiatorPrincipalId>
        <password>2009</password>
      </context>
      <reportId>Reconciliation/STD_RECON_006.xml</reportId>
      <language>en</language>
      <parameters>
        <parameter>
          <entry>
            <key>transactionPeriodId</key>
            <value>2012102916093771801000146</value>
          </entry>

          <entry>
            <key>TERMINALID</key>
            <value>2469e0bf14214797880cafb0eda1b535</value>
          </entry>

        </parameter>
      </parameters>
    </ext:executeReport>
  </soapenv:Body>
</soapenv:Envelope>
```

Example 8.10. An example of executing report STD_RECON_006 SOAP response

```

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns2:executeReportResponse xmlns:ns2="http://
external.interfaces.ers.seamless.com/">
      <return>
        <resultCode>0</resultCode>
        <resultDescription>SUCCESS</resultDescription>
        <report>
          <title>STD_RECON_006_2012102916093771801000146.xml</title>
          <mimeType>text/html</mimeType>
          <content>PD94...==</content>
          <contentString><![CDATA[<?xml version="1.0" encoding="UTF-8"?>
<CashierTransactionSummary>
<Meta>
<ResellerId>hml</ResellerId>
<TerminalId>2469e0bf14214797880cafb0eda1b535</TerminalId>
<TransactionPeriodId>2012102916093771801000146</TransactionPeriodId>
<GeneratedAt>2012-10-30 12:18</GeneratedAt>
</Meta>

<Row>
<SalesCount>1</SalesCount>
<SalesTotal>10.00000</SalesTotal>
</Row>

</TerminalTransactionSummary>]]></contentString>
        </report>
      </return>
    </ns2:executeReportResponse>
  </soap:Body>
</soap:Envelope>

```

8.2.6. STD_RECON_007 SOAP examples

Example 8.11. An example of executing report STD_RECON_007 SOAP request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:ext="http://external.interfaces.ers.seamless.com/">
  <soapenv:Header/>
  <soapenv:Body>
    <ext:executeReport>
      <context>
        <channel>WS</channel>
        <clientComment>comment</clientComment>
        <clientId>testClient</clientId>
        <clientReference>12345</clientReference>
        <clientRequestTimeout>0</clientRequestTimeout>
        <initiatorPrincipalId>
          <id>hml</id>
          <type>RESELLERUSER</type>
          <userId>9900</userId>
        </initiatorPrincipalId>
        <password>2009</password>
      </context>
      <reportId>Reconciliation/STD_RECON_007.xml</reportId>
      <language>en</language>
      <parameters>
        <parameter>
          <entry>
            <key>transactionPeriodId</key>
            <value>2012102916093771801000146</value>
          </entry>

          <entry>
            <key>TERMINALID</key>
            <value>2469e0bf14214797880cafb0eda1b535</value>
          </entry>

        </parameter>
      </parameters>
    </ext:executeReport>
  </soapenv:Body>
</soapenv:Envelope>
```

Example 8.12. An example of executing report STD_RECON_007 SOAP response

```

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns2:executeReportResponse xmlns:ns2="http://
external.interfaces.ers.seamless.com/">
      <return>
        <resultCode>0</resultCode>
        <resultDescription>SUCCESS</resultDescription>
        <report>
          <title>STD_RECON_007_2012102916091394901000143.xml</title>
          <mimeType>text/html</mimeType>
          <content>PD94...=</content>
          <contentString><![CDATA[<?xml version="1.0" encoding="UTF-8"?>
<CashierTransactionSummary>
<Meta>
<ResellerId>hml</ResellerId>
<TerminalId>2469e0bf14214797880cafb0eda1b535</TerminalId>
<TransactionPeriodId>2012102916093771801000146</TransactionPeriodId>
<GeneratedAt>2012-10-30 12:20</GeneratedAt>
</Meta>

<Row>
<TerminalId>2469e0bf14214797880cafb0eda1b535</TerminalId>
<CashierId>bob</CashierId>
<SalesAmount>10.00000</SalesAmount>
<ErsReference>2012102916091394901000143</ErsReference>
</Row>

</TransactionDetails>]]</contentString>
        </report>
      </return>
    </ns2:executeReportResponse>
  </soap:Body>
</soap:Envelope>

```