Introduction

This page describes the protocol implemented for Westpay's Pay@Table component. This is the protocol that operates between the terminal and the restaurant's server that manages tables and orders.

Note that SEQR support is now obsolete, and will be removed from this protocol at some future stage. It is kept here for reference only to support existing installations.

Message wrapper

The protocol is implemented as XML messages in a simpler wrapper. The wrapper consists of a four byte header that gives the length of the data to follow. If the data is 171 bytes long (0xAB) then the first four bytes of the message will be:

```
00 00 00 AB
```

The rest of the message is the XML data encoded using UTF-8.

Fundamentals

- Every request that is sent from the terminal should have a response from the server.
- The server never initiates a message. All requests originate at the terminal.

Communications parameters

- Pay@Table only works over a network connection using regular TCP/IP sockets.
- The terminal takes the role of the client and it creates a new connection to the server for each message pair.
- The convention is to use port 45017 for connections, but this is configurable.

Protocol Details

Sequence of operations

After the terminal has connected to the Pay@Table server it generally follows this sequence:

1. If necessary, **fetch a list of the available tables** from the server. The Pay@Table software on the terminal will then display the list for table selection and the required table can be selected from the list. This functionality can be disabled in the Pay@Table software on the terminal, which means the server does not have to maintain a list of available tables.
2. **Open a table** in order to retrieve the order for the table, the amount to pay, and details of any payments already made.
3. Optionally retrieve data for a **pro-forma receipt** that simplifies splitting the bill among the customers.
4. **Make a payment** on an open table. Multiple payments can be made if the bill is to be split.
5. Optionally **get receipt data** to print an accounting receipt, e.g. for expenses.
6. **Release the table** if there will be more payments to make, or **close the table** once finished.

Closing a table means that everything has been paid for. The Pay@Table server may choose to lock a table while it is open, and therefore prevent another terminal from opening the table.

Special cases

There are some special cases / operations that are reflected in protocol usage and they are covered here.

SEQR

If configured to do so, the terminal can run SEQR transactions. The procedure for this is slightly complex because the SEQR payment is run by the Pay@Table server / ECR so, unlike card payments, the terminal software has to have a dialogue with the server to start, monitor and cancel (if needed) the SEQR transaction.

The procedure here is that the customer selects SEQR as a payment method and then the terminal sends a request to the server to initiate a SEQR purchase. The server responds with an invoice reference and the value for a QR code. The terminal displays the QR code on screen and then asks the server for the transaction status. This status request only happens once, so the server waits for the SEQR transaction to finish or to timeout, and then sends the status response to provide the outcome.

The terminal can cancel the SEQR purchase while waiting. If this happens then the server needs to clean up the SEQR transaction properly, so if the transaction was successful but the terminal cancelled (i.e. the cancellation happened at the same time as the transaction was approved) then the server must deal with reversing the SEQR purchase.

XML Request and Response Contents

Example messages

OpenTable request example

```
<?xml version="1.0" encoding="utf-8"?>
<rtmp>
  <request sequenceNumber="8" serialNumber="D0029" type="OpenTable"
terminalId="52400004" tableId="2" waiterId="1" />
</rtmp>
```

OpenTable response example

```
<?xml version="1.0" encoding="utf-8"?>
<rtmp>
 <response type="OpenTable" sequenceNumber="8" serialNumber="D0029"
responseCode="0">
 <table tableId="2" totalAmount="205250" />
 </response>
</rtmp>
```

UpdateTable request example

```
<?xml version="1.0" encoding="utf-8"?>
<rtmp>
  <request sequenceNumber="11" serialNumber="D0029" type="UpdateTable"
terminalId="52400004" tableId="2" waiterId="1" amount="55200" extra="0"
paymentType="card" referenceNumber="524000042748"
financialInstitution="SWE" />
</rtmp>
```

UpdateTable response example

```
<?xml version="1.0" encoding="utf-8"?>
<rtmp>
  <response type="UpdateTable" sequenceNumber="11" serialNumber="D0029"
responseCode="0">
    <table tableId="2" totalAmount="205250">
      <payment type="card" amount="150050" />
      <payment type="card" amount="55200" />
    </table>
  </response>
</rtmp>
```

TableReceipt response example

```
<?xml version="1.0" encoding="utf-8"?>
<rtmp>
  <response type="TableReceipt" sequenceNumber="22" serialNumber="D0029"
responseCode="0" controlBox="1" footer="Receipt footer text&#xD;&#xA;End
of receipt" header="WestInt restaurant" receiptNumber="2">
    <table tableId="2" totalAmount="205250">
      <payment type="card" amount="120050" />
      <payment type="cash" amount="25000" />
      <payment type="card" amount="60200" />
    </table>
    <food name="Pizza" price="50000" count="3.5" />
    <food name="Beer" price="4500" count="6" />
    <food name="Cider" price="6500" count="0.5" />
    <vat percent="12" amount="6000" />
    <vat percent="25" amount="2750" />
  </response>
</rtmp>
```

XML elements

<rtmp> (root node)

The root node of all the messages is

`<rtmp>`

This is an abbreviation of Restaurant Table Management Protocol.

<request>

Requests from the terminal are enclosed in a request element. The request element has the following attributes:

Attribute	Required	Data type and limits	Description
sequenceNumber	yes	number 0 .. 99	The message sequence number is included in every message that is sent. It is set to 0 for the first request after start up, then incremented after each request is sent. After message number 99 the value is set back to 1. The Pay@Table server's response should echo the sequence number sent in the request. If the sequence number in the response does not match the request then it will be treated as an error.
serialNumber	yes	text	The serial number attribute is set by the terminal in the request to the server. It is a unique value that identifies the terminal, and is typically set to the terminal's hardware serial number. The server can use this to identify an individual terminal. The server should echo the serial number in the response. If the terminal receives a different value for the serial number then it will be treated as an error.
type	yes	text	This is taken from the list of message types above, e.g. "UpdateTable", "Loyalty", etc. These are covered in detail below.
terminalId	yes	text max 16 characters	The terminal ID is used by the payment systems to identify the terminal. The Pay@Table server can use it to identify the terminal in a similar way to the serial number.
waiterId	no	number max 12 digits	The waiter ID identifies the waiter. A waiter ID can be pre-defined for a terminal if desired, and this avoid the need for a waiter ID to be selected.
tableId	no	text max 12 characters	The table ID identifies the table being handled. Consists of digits 0 - 9 and can contain the characters '*', '/' and '#'.
amount	no	number	The amount attribute indicates the amount of a transaction. Amounts are always expressed in minor currency units.

extra	no	number	The extra attributes indicates the tip amount for a transaction.
paymentType	no	text	The payment type identifies how an amount was paid. There are some pre-defined values for this attribute but it can also have a custom value depending on configuration: • "card" indicates a card payment was made using the terminal. • "cash" indicates a cash payment was made. • "seqr" indicates that SEQR was used for a payment.
referenceNumber	no	text max 12 characters	Unique transaction identifier.
financialInstitution	no	text max 3 characters	The three-letter financial institution code of a card used for payment.
invoiceReference	no	text	SEQR invoice reference.
loyaltyCardNumber	no	text max 19 characters	Loyalty card number.
checkSequenceNr	no	number max 10 digits	Receipt sequence number, used in a Micros environment.
ipAddress	no	text normal <addr>:<port> format	IP address of the terminal, provided so that an EPAS ECR can connect to the terminal.
epasModeOn	no	text	Indicates that the terminal is now in EPAS mode. The only value used here is "true".

<response>

Responses from the Pay@Table server are always enclosed in a response element.

Attribute	Required	Data type and limits	Description
sequenceNumber	yes	number 0 .. 99	The message sequence number is included in every message that is sent. It is set to 0 for the first request after start up, then incremented after each request is sent. After message number 99 the value is set back to 1. The Pay@Table server's response should echo the sequence number sent in the request. If the sequence number in the response does not match the request then it will be treated as an error.
serialNumber	yes	text	The serial number attribute is set by the terminal in the request to the server. It is a unique value that identifies the terminal, and is typically set to the terminal's hardware serial number. The server can use this to identify an individual terminal. The server should echo the serial number in the response. If the terminal receives a different value for the serial number then it will be treated as an error.
type	yes	text	This is taken from the list of message types above, e.g. "UpdateTable", "Loyalty", etc. These are covered in detail below.
responseCode	yes	number 0 .. 99	The response code is included in every response. Zero means success, other outcomes are documented later.
errorString	no	text	If the responseCode indicates an error then the text here can be supplied for more detailed reporting. The Pay@Table terminal application will display this on screen.
invoiceQrCode	no	text	If the customer chooses to pay using SEQR then the QR code details from SEQR are supplied here so that the terminal can display it on screen.
invoiceReference	no	text	If the customer chooses to pay using SEQR then the SEQR invoice reference is given here.
header	no	text	Receipt header for the terminal to print at the start of a receipt.
footer	no	text	Receipt footer for the terminal to print at the end of a receipt.
controlBox	no	text max 17 characters	Control box ID to be included on the terminal receipt
receiptNumber	no	text max 12 characters	Receipt number set by the ECR to be included on the terminal receipt
receiptContent	no	text	A pre-formatted table receipt can be provided here. Each receipt line is separated with a line feed (ASCII 10 / 0x0A) and should be formatted to include 32 characters per line.

<table>

Responses can include the table element to provide information on a table's status. Multiple table elements can be used to provide a choice of available tables to a waiter.

Attribute	Required	Data type and limits	Description
totalAmount	no	number	The total amount of the bill for the table. This is not the balance to pay, i.e. it does not decrease as payments are made.
extra	no	number	The total of tips paid on this table.
tableId	no	text max 12 characters	The ID of this table. Consists of digits 0 - 9 and can contain the characters '*', '/' and '#'.
departmentId	no	text max 12 characters	The department ID indicates which department / section this table is in. When the terminal is processing the response to the FetchTables request then if the terminal has a department ID set, and if this attribute has a value, it will compare the attribute value against the terminal's department ID and will exclude tables that do not match.
checkSequenceNr	no	number max 10 digits	See the documentation for the request element for details.
checkNumber	no	number	When the terminal is processing the response to the FetchTables request then If the check number is defined the check number will be shown alongside the table number in the list of available tables, rather than showing the balance to pay.

<payment>

The payment element is only found as a child of a table element. Each payment element represents a single payment on a table, and a table can have multiple payments.

Attribute	Required	Data type and limits	Description
type	yes	text	The payment type. Refer to paymentType in the request element.
amount	yes	number	The amount, in minor currency units, of the payment.

<food>

The food element is used in pro-forma and table receipt data sent from the Pay@Table server / ECR to itemise the contents of the bill. While this is typically an item of food or drink, in can be anything.

Attribute	Required	Data type and limits	Description
name	yes	text max 30 characters	The text of the bill item to be shown on the receipt.
price	yes	number	The cost, in minor currency units, of one of these items.
count	yes	decimal	The number of these items on the bill.

<vat>

The vat element is used in the table receipt data sent from the Pay@Table server / ECR. If multiple VAT rates / bands apply then multiple VAT elements can be included.

Attribute	Required	Data type and limits	Description
percent	yes	decimal	The VAT rate as a percentage.
amount	yes	number	The amount, in minor currency units, of VAT charged at this rate.

Response code (attribute name: "responseCode")

This is a number that indicates the outcome of the request.

If the value is zero then it indicates success, and the rest of the response is expected to follow the protocol. A non-zero value is a failure, and the message does not need to contain any further elements.

Possible values are:

Code	Meaning
0	Success
1	Invalid waiter ID
2	Invalid table ID
3	Table is closed
4	Table is locked
5	Wrong key
6	Communications failure between the Pay@Table server and the ECR
7	Refund amount too high
8	No open tables found
9	Loyalty / bonus card number registration failed
Any other value	Generic error

Message types

The 'type' attribute indicates the message type that is being sent. Each message type represents an operation or request, and these are detailed below. Note that 'attributes used' indicates which of the optional attributes can be used with that message, and the mandatory attributes are not listed here in order to save space.

The following notes apply to some of the attributes, and these notes should be considered alongside the details of which attributes are used in the various requests and responses.

Attribute	Note
errorString	If the response indicates an error then the errorString may be used to provide additional details. This can be done in any response to the terminal.
checkSequenceNr	If the Pay@Table server / ECR provides a check sequence number in response to an OpenTable request then it will be included in every request sent until the table is closed.

type when the operation is successful. If it not successful then the errorString attribute can be used in any response. To save space the mandatory attributes (see above) are not included.

Message type	Request details	Response details
FetchTables	Requests a list of available tables from the Pay@Table server / ECR. The tableId attribute is defined on the terminal, and can be interpreted in whatever way the Pay@Table server / ECR wishes. If the tableId attribute is not given, or if it is empty, then the Pay@Table server / ECR should give a list of all the available tables for the waiter. Optional attributes used: tableId (optional), waiterId (required)	The Pay@Table server / ECR should reply with zero or more table elements containing information on tables that can be opened by the waiter. Optional attributes used: *none* Elements used: table
OpenTable	Requests that the indicated table should be opened and its details returned to the terminal. The Pay@Table server / ECR may choose to lock the table to the terminal that opened it, preventing other terminals from performing any operations on the table until the first terminal releases or closes the table. Optional attributes used: tableId (required), waiterId (required)	If the table can be opened then the Pay@Table server / ECR should reply with one table element containing information on selected table. Optional attributes used: *none* Elements used: table

UpdateTable	The terminal sends this to notify the Pay@Table server / ECR of a payment for the table. The amount of the payment may not be the full amount required for the table. Optional attributes used: tableId(required), waiterId (required), amount (required), extra (required), paymentType (required), referenceNumber (required), financialInstitutaion (required if paymentType = "card")	The Pay@Table server / ECR should response with the updated table element that reflects the payment that has been made.
ReverseTable	The terminal sends this if a payment, previously notified via UpdateTable, is to be reversed. Optional attributes used: tableId(required), waiterId (required), amount (required), extra (required), paymentType (required), referenceNumber (required), financialInstitutaion (required if paymentType = "card")	The Pay@Table server / ECR should response with the updated table element that reflects the payment that has been reversed.
CloseTable	The terminal sends this to indcate that all amounts due on the table have been paid and that the selected table is available for a new customer. Optional attributes used: tableId (required)	A successful response indicates that the table has been closed. Optional attributes used: *none* Elements used: *none*
ReleaseTable	The terminal sends this to indicate that that it is no longer handling this table, but that the table cannot be closed because there is still something to pay. An example of this is where the customer asks for a receipt but is not yet ready to pay (i.e. table is opened, receipt is printed, table is released.) If a table is locked in the ECR then it can be unlocked when this is received. ReleaseTable is a configurable option in the terminal, and will only be used if enabled. If ReleaseTable is not enabled then it may be best if the ECR does not implement strict table locking, since that would mean a second terminal would be unable to access the table until the first terminal closes it once everything has been paid. Optional attributes used: tableId (required), waiterId (required)	A successful response indicates that the table has been released. Optional attributes used: *none* Elements used: *none*
Loyalty	The terminal sends this when a loyalty card has been presented. Payment card number are never provided. Optional attributes used: tableId (required), loyaltyCardNumber (required), amount (required)	A successful response indicates that the loyalty card details have been accepted. The Pay@Table server / ECR can return an error code of 9 with appropriate error text if the loyalty card is not acceptable. Optional attributes used: *none* Elements used: *none*
TableReceipt	The terminal sends this to retrieve data for an accountable receipt. An accountable receipt shows the breakdown of payments, what was ordered, and VAT amounts. The intention is that it can be used as a record of business expenses. A TableReceipt is only allowed by the terminal when the whole bill for the table has been paid. Optional attributes used: tableId (required)	There are two forms of response to this request.
		Response 1 is to provide the elements of the receipt data and then leave the terminal to format and print it. Optional attributes used: controlBoxId (required), receiptNumber (required), header (optional), footer (optional) Elements used: table (required), vat (required), food (optional)
		Response 2 is to provide a pre-formatted receipt that the terminal will print without reference to the contents. The receipt contents are formatted as described in the attribute documentation above. Optional attributes used: receiptContent (required) Elements used: *none*

ProFormaReceipt	The terminal sends this request to retrieve data for a pro-forma receipt. This is where a summary of the table is needed so that the customers can work out how to split the bill. A pro-forma receipt is available when the bill for the table has not been paid. Optional attributes used: tableId (required)	In the same way as for the TableReceipt response, there are two forms of response and the respone details for TableReceipt also apply here, with the single exception that there is no vat element in the response here.
SeqrInvoice	The terminal sends this request if the customer has chosen SEQR as a payment method. SEQR has to be enabled in the terminal and also in fixed terminal configuration before it can be selected. Optional attributes used: tableId (required), amount (required)	The Pay@Table server / ECR should initiate a SEQR transaction and should respond with both the invoice reference and the QR code value. The terminal will display the QR code on screen ready for the payment process to begin. Optional attributes used: invoiceReference (required), invoiceQrCode (required) Elements used: *none*
SeqrPaymentStatus	When a SEQR payment is in progress, i.e. the terminal has displayed the SEQR QR code, the terminal will send this request to the Pay@Table server / ECR. The server should not reply until the SEQR transaction has ended. Note that if the SEQR payment process is successful then the terminal will send the associated UpdateTable request immediately afterwards. Optional attributes used: invoiceReference (required)	When the SEQR transaction has ended then the Pay@Table server / ECR should response to indicate the status of the transaction, i.e. success or failure. The errorString attribute will be useful to indicate the reason for failure. If payment was successful then the response should indicate success in the normal, way, i.e. with the response code set to zero. Optional attributes used: *none* Elements used: *none*
CancelSeqrInvoice	While waiting for a SEQR payment to complete the waiter has the option to cancel the SEQR payment. If this is selected then the network socket that is used for the waiting SeqrPaymentStatus request is closed and then this cancellation request is sent. Note: From the Pay@Table server / ECR perspective there are two ways to finish a SEQR payment. Either the terminal sends an UpdateTable request with the payment type set to "seqr", in which case all is well and the payment can be logged, or it sends a CancelSeqrInvoice request, in which case the payment can be cancelled or, if already made, reversed. If the terminal is shut down or there is a communications failure then it is possible that there will be no response. Manual intervention is likely to be required there to establish if the transaction was completed or not.	When the Pay@Table server / ECR receives this cancellation request then it should abort the SEQR payment. It is possible that the SEQR payment was made while the cancellation was happening, in which case the SEQR payment should be reversed. Optional attributes used: *none* Elements used: *none*
SeqrRefund	This requests asks the Pay@Table server / ECR to issue a refund using SEQR. There are two ways this can be requested: • The waiter requests a reversal of a SEQR payment made on a table, in which case a ReverseTable request will follow the successful completion of the SEQR refund, or • The waiter has started an individual (non-table-related) refund request of a SEQR transaction. In this case there will be no table ID and no ReverseTable request. The terminal does not use SeqrPaymentStatus in the refund case, it simply waits for a response. Optional attributes used: invoiceReference (required), amount (required)	The Pay@Table server / ECR should run a SEQR refund transaction and the report success or failure in the normal way. Optional attributes used: *none* Elements used: *none*

TerminalRegistration	There is a function in the terminal to switch from Pay@Table mode to Pay@Counter. Pay@Counter is a mode where the terminal operates as a server for an EPAS ECR and the details of this are very much beyond the scope of this document. If the switch is succesful then the terminal sends the TerminalRegistration request to inform the Pay@Table server / ECR of the basic operational and connection details for EPAS operation. Optional attributes used: ipAddress (required), terminalId (required), serialNumber (required), epasModeOn (required, set to "true")	The Pay@Table server / ECR should send a success response. In practice the terminal will ignore the response, since the switch to EPAS mode has already been made at this stage and it is not dependent on the agreement of the Pay@Table server / ECR. Optional attributes used: *none* Elements used: *none*