# In case of the Receipt:

**Where does it get handled?** It gets handled completely in the **Frontend** by passing the value of this field to the Receipt screen.

**Does it appear on production?** For Android **YES** for iOS **NO.**

**How to make it appear?** Both will be handled from the front-end side.

# In case of the Transaction History:

**Where does it get handled?** It gets handled completely in the **Frontend, but we get it from the Backend,** in case of Android this happens at DefaultTransactionNameResolver.kt, this gets used in ApolloTransactionHistoryRepository.kt file, the GetSavingsAccountTransactionListUseCase uses a TransactionHistoryRepository repository, the implementation in the ApolloTransactionHistoryRepository and the GetSavingsAccountTransactionListUseCase is being used on our view model: TransactionHistoryListViewModel**.**

**Does it appear on production?** For Android **NOT SURE** for iOS **NO.**

**How to make it appear?** Both will be handled from the frontend side.

**Where does it get handled in the Backed?** Apollo calls transaction-service which queries all the transaction data from its database called retail\_transaction in transaction database, when we investigated the other payment details for the pay 2 proxy we couldn’t find it, modifying the value gave us the correct outcome.

* The only source the transaction-service gets it’s dta from is from a Kafka topic pfm.transaction-service.create-transaction-command.v1 where we send the trasnaction details the very last step after it is completed.
* Payment-service sends messages on this Kafka topic.
* We reached the conclusion that the kafka topic basically quries the payment data from the payment-service database itself, that when inspected, was found to not contain payment details field in the database for the pay to proxy transaction, from backwards to forward: publishCreateTransactionCommandEvent in CreateTransactionCommandProducer.kt in payment-service > PostingInstructionBatch.publishCreateTransactionCommand in PostingHistoryService > handleIncomingPostingInstructionBatch in PostingHistoryService > consumeTMPostingEvent in PaymentVaultPostingConsumer.kt.
* We query the payment transaction data from createPaymentTransaction in PostingHistoryService.kt.
* Whenever there is an update to the payement, it is posted on this topic: payment.payment.snapshot.v1, for the pay2prozy case this was called 7 times only for the pay2proxy case.
* handleOutboundPaymentCompletedEvent function in CreditTransferCompletedService.kt either changes it to the wrong value or get the wrong value.
* The value gets assigned in the PaymentEntity.enrichOutboundPaymentEntity

Function, it get its value from RequestedPayment object, which if traced back, comes from fis-gateway.

* The wrong value comes from doPayment in fis-gatewway, there is logs for this called “Payment request:”, this is most likely due to wrong data origninally sent to it.

# In case of the eStatments:

**Where does it get handled?** It gets handled completely in the **Backend in statement-service:**

* The flow starts by a schedular that runs exactly at 10 AM every 5th day of the month that sends a message to an internal Kafka topic that calls a function called CreateCustomerStatementCommandProcessor.
* In RetailStatmentService the function generateStatement gets called collects data and generates a pdf.
* In the step where it generates this data it generates the transactions, fast forward to CustomerTransaction service in the function getCustomerTransactionsDetails.
* In line 50 transactionApiModelMapper.mapToDto function gets called to map each transaction data depending on the transaction type, if it’s an SME transaction it just prints the SME transaction name without any other thing, **but in case of deposit account (saving or current supposedly) it calls prepareTransactionDetails function**.
* The last point is on TransactionApiModel calling TransactionApiModel.prepareTransactionDetails function that in the case of the transaction type not being saving pot transfer or pot transfer, it just returns in the details field the: description, source name, destination name, recipientReference and most importantly **other details field** each separated by a slash.

**Does it appear on production?** For Android **NOT SURE** for iOS **NOT SURE.**

**How to make it appear?** Depending on if they show on production or not, everything will be handled on the backend.