POST request url : <http://localhost:8088/statementProcessor>

1. Sample Input JSON : {"path":"/Users/738575/records.csv"}

Sample Output :

{

"duplicateTransactionReference": [

{

"transactionRefrence": 163590,

"description": "Tickets from Rik Bakker"

},

{

"transactionRefrence": 158338,

"description": "Tickets for Vincent King"

},

{

"transactionRefrence": 112806,

"description": "Subscription for Jan Theu\ufffd"

}

],

"invalidEndBalance": []

}

1. Sample Input JSON : {"path":"/Users/738575/records.xml"}

Sample Output:

{

"duplicateTransactionReference": [],

"invalidEndBalance": [

{

"transactionRefrence": 154270,

"description": "Candy for Peter de Vries"

},

{

"transactionRefrence": 140269,

"description": "Tickets for Vincent Dekker"

}

]

} {

"transactionRefrence": 112806,

"description": "Subscription for Jan Theu\ufffd"

}

],

"invalidEndBalance": []

}

The following Lamda expression is used to find failed end balance.

customerStatementRecordList.stream().filter(p -> !p.getStartingBalance().add(p.getMutation()).equals(p.getEndBalance()))

.collect(Collectors.*toList*());

The following Lamda expression is used to find duplicate Transaction reference

customerStatementRecordList.stream().collect(Collectors.*groupingBy*(Function.*identity*(), Collectors.*counting*())).entrySet().stream()

.filter(e -> e.getValue() > 1L).map(e -> e.getKey()).collect(Collectors.*toList*());