```
const _ = require('lodash');
const request = require('request');
const json2xml = require('json2xml');
const async = require('async');
const { Order } = require('../orders/model');
const logger = require('../../services/logger');
const responseHandler = require('../../services/response');
const { wirecardProcessing, voidSACTSales, voidWirecardPayment,
flagCriticalError } = require('./utility');
const { confirmEmail } = require('./emailer');
const { confirmTickets, processConfirmedTickets } = require('./
process_confirmations');
var mongoose = require('mongoose');
require('mongoose-long')(mongoose);
var Long = mongoose.Types.Long;

exports.processAttractionsConfirm = (res, req, wirecard, reservation_id,
transaction_id, email_type, batch) => {
  const query = {reservation_id : Long.fromString(reservation_id.toString()) }
  Order.findOne(query, { __v: 0, _id: 0 })
    .exec((err, order) => {
      if(!err)  {
        if(order !== null)  {
          if(wirecard)  {
            req.headers.uid = order.user_id
          }
          let attractions = order.attractions
          // formulate xml
          let tickets_xml = ""
          _.filter(attractions, (attraction) => {
            _.filter(attraction.tickets, (ticket) => {
              _.filter(ticket.types, (type) => {
                let ticket = {
                  "BookingFee":  {
                    "itemcode": 0,
                    "BookingFee": 0,
                    "Quantity": 0
                  }
                }
                let ticket_xml = json2xml(ticket)
                tickets_xml += ticket_xml
              })
            })
          })
```

```
        let tickets_confirm_in = {
          tickets: tickets_xml,
          transaction_id: req.body.transaction_id,
          card_digits: req.body.digits,
          reservation_id: order.reservation_id,
          name: order.name,
          email: order.email,
          contact: order.contact,
          order_id: order.order_id,
          receipt_number: order.receipt_number,
          res: res,
          req: req
        }
        confirmTickets(tickets_confirm_in, function(err,
confirm_tickets_response) {
          if(!err && typeof confirm_tickets_response !== 'undefined')  {
            //logger.log('sact', req.headers.correlation_id, null, 'info', { url:
sact_req.uri.href, method: sact_req.method, headers: sact_req.headers, body:
sact_req.body }, response.statusCode, response.body)
            processConfirmedTickets(res, req, reservation_id, wirecard, order,
confirm_tickets_response, email_type, function(err, confirmed_order) {
              if(!err)  {
                batch ? console.log('batch (attr)') : responseHandler.success(res,
confirmed_order)
              } else { // err
                var tasks = [];
                tasks.push(async.apply(voidSACTSales, reservation_id));

                if (wirecard) {
                  tasks.push(async.apply(voidWirecardPayment, res,
req.body.payment_timestamp, req.body.request_id, req.body.transaction_id));
                }

                async.parallel(tasks, (err, response) => {
                  batch ? console.log('batch (attr)') :
responseHandler.confirmationUnsuccessful(res, { order_id:
confirmed_order.order_id });
                });
              }
            })
          } else { // error confirming tickets
            // delay and retry
            setTimeout(function() {
              processConfirmedTickets(res, req, reservation_id, wirecard, order,
confirm_tickets_response, function(err, confirmed_order) {
                if(!err)  {
```

```
                batch ? console.log('batch (attr)') : responseHandler.success(res,
confirmed_order)
              } else {
                var tasks = [];
                tasks.push(async.apply(voidSACTSales, reservation_id));

                if (wirecard) {
                  tasks.push(async.apply(voidWirecardPayment, res,
req.body.payment_timestamp, req.body.request_id, req.body.transaction_id));
                }

                async.parallel(tasks, (err, response) => {
                  batch ? console.log('batch (attr)') :
responseHandler.confirmationUnsuccessful(res, { order_id:
confirmed_order.order_id });
                });
                console.log("error confirming tickets")
                console.log(err)
              }
            })
          }, 3000)
        }
      })
    } else { // order not found
      flagCriticalError(res, reservation_id, transaction_id, 'order not found
error')
      batch ? console.log('batch (attr)') : responseHandler.notFound(res)
      console.log("no order found")
      console.log(err)
    }
  } else { // db query for order error
    var tasks = [];
    tasks.push(async.apply(voidSACTSales, reservation_id));

    if (wirecard) {
      tasks.push(async.apply(voidWirecardPayment, res,
req.body.payment_timestamp, req.body.request_id, req.body.transaction_id));
    }

    async.parallel(tasks, (err, response) => {
      flagCriticalError(res, reservation_id, transaction_id, 'db query error')
      logger.log('cosmodb', res.req.headers.correlation_id, null, 'error', query,
null, err);
      batch ? console.log('batch (attr)') : responseHandler.serverError(res)
    });
    console.log("db query error")
```

```
        console.log(err)
      }
    })
}
```