```
const Joi = require('joi');
const _ = require('lodash');
const async = require('async');
const parser = require('xml2js').parseString;
const config = require('../../config');
const responseHandler = require('../../services/response');
const { confirmRequestSchema } = require('./model');
const { confirmWOT, confirmTickets, confirmEmail, wirecardProcessing } =
require('./utility');
const { getWOTCapacity, toSACTDateTimeFormat, voidSACTSales,
flagCriticalError } = require('../reserve/utility');
const { processWOTAndAttractions } = require('./process_wotattractions');
const { processWOTConfirm } = require('./process_wot')
const { processAttractionsConfirm } = require('./process_attractions');

exports.index = (req, res, next) => {
  let wirecard = false;
  if (req.body.eppresponse) {
    wirecard = true;
  }
  // delay of random between 0 to 11 seconds
  // let delay = Math.floor(Math.random() * 10000) + 1000;
  let delay = 0;
  // set delay timer to stagger load onto third party APIs
  setTimeout(function() {
    async.waterfall([
      async.apply(wirecardProcessing, {req: req, res: res })
    ], function(err) {
      if(!err)  {
        const result = Joi.validate(req.body,
Joi.object().keys(confirmRequestSchema), { presence: "required",
stripUnknown: true })
        const validate_id = Joi.validate(req.params.id, Joi.number().min(0),
{ stripUnknown: true })

        if(result.error === null) {
          req.body = result.value
          req.params.id = validate_id.value
          let reservation_id = req.params.id
          let transaction_id = req.body.transaction_id
          let name = req.body.name || ''
          let wot_pin = req.query.pin || null
          // if has wot
          if(wot_pin !== null && (parseInt(reservation_id) === 0)) { // only wot
```

```
            processWOTConfirm(res, req, wirecard, wot_pin, reservation_id,
transaction_id, 1, false)
          } else if(wot_pin !== null && (parseInt(reservation_id) !== 0)) { // both wot
and attractions
            processWOTAndAttractions(res, req, wirecard, wot_pin, reservation_id,
transaction_id, 0, false)
          } else { // only normal
            processAttractionsConfirm(res, req, wirecard, reservation_id,
transaction_id, -1, false)
          }
        } else { // request schema invalid error
          // *** void sales
          voidSACTSales(reservation_id, (err, response) => {
            flagCriticalError(res, reservation_id, transaction_id, 'invalid schema
error')
            let detailed_error = result.error.details[0].message // detailed error msg
            responseHandler.invalidFields(res, detailed_error)
          })
        }
      } else { // waterfall error - wirecard payload processing error
        voidSACTSales(reservation_id, (err, response) => {
          responseHandler.waterfallErrors(res, err)
        })
      }
    });
  }, delay)
}
```