```
const async = require('async');
const config = require('../../config');
const mcache = require('memory-cache');
const request = require('requestretry');
const moment = require('moment');
const { clear, show, remove } = require('../../services/mcache');
const parser = require('xml2js').parseString;
const { Order } = require('../orders/model');
const logger = require('../../services/logger');
const responseHandler = require('../../services/response');
const _ = require('lodash');
const { toggleEnv, voidSACTSales } = require('./utility');
const { confirmEmail, smtpMail } = require('./emailer');
var mongoose = require('mongoose');
require('mongoose-long')(mongoose);
var Long = mongoose.Types.Long;

exports.confirmWOT = (data, callback) => {
  let res = data.res
  let req = data.req
  let deviceId = '02310';
  if(!(data.receipt_number.indexOf('OSPM') > -1)){
    deviceId = '02313';
  }
  let ticketing_url = config.ticketing.wot.url + "/OnlineWebService.asmx?
WSDL="
  let wot_confirm_xml = '<?xml version="1.0" encoding="utf-8"?
><soap12:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap12="http://
www.w3.org/2003/05/soap-envelope"><soap12:Body><ConfirmSales
xmlns="http://SACT.WOTWebService.Webservices/"><ConfirmIn><Password>'
+ config.ticketing.wot.password + '</Password><TransactionID>' +
data.receipt_number + '</TransactionID><Pincode>' + data.wot_pin + '</
Pincode><Name>' + data.name + '</Name><Email>' + data.email + '</
Email><TelNo>' + data.contact + '</TelNo><BookingFee><ItemCode>0</
ItemCode><Quantity>0</Quantity></BookingFee><PaymentID>' +
data.transaction_id + '</PaymentID><PaymentType>0</
PaymentType><CreditCard>' + data.card_digits + '</CreditCard><DeviceID>'
+deviceId+ '</DeviceID></ConfirmIn></ConfirmSales></soap12:Body></
soap12:Envelope>'
  request.post({
    url: ticketing_url,
    body: wot_confirm_xml,
    agentOptions: {
```

```
      rejectUnauthorized: false
    },
    headers: { 'Content-Type': 'text/xml' }
  }, function(err, response, body) {
    if(!err) {
      let sact_req = response.request
      parser(body, { explicitArray: false }, function(err, results) {
        if(!err)  {
          let confirm_response = results['soap:Envelope']['soap:Body']
['ConfirmSalesResponse']['ConfirmSalesResult']
          if(confirm_response['ResponseCode']['Status'] === 'Success' ||
confirm_response['ResponseCode']['Status'] === 'LastCallSuccessful') {
            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)
            callback(null, confirm_response)
          } else { // failed response code from sact
            logger.log('sact', req.headers.correlation_id, null, 'error', { url:
sact_req.uri.href, method: sact_req.method, headers: sact_req.headers, body:
sact_req.body }, response.statusCode, response.body)
            console.log(err)
            // responseHandler.serverError(res)
            callback(err)
          }
        } else { // parsing of xml failed
          logger.log('sact', req.headers.correlation_id, null, 'error', { url:
sact_req.uri.href, method: sact_req.method, headers: sact_req.headers, body:
sact_req.body }, response.statusCode, err)
          // responseHandler.partnerErrors(res, err)
          callback(err)
        }
      })
    } else { // sact request errors
      logger.log('sact', req.headers.correlation_id, null, 'error', { url: ticketing_url,
method: 'POST', headers: {}, body: {} }, 500, err)
      // responseHandler.requestErrors(res, err)
      callback(err)
    }
  })
}

exports.confirmTickets = (data, callback) => {
  let res = data.res
  let req = data.req
  let deviceId = '02310';
  if(!(data.receipt_number.indexOf('OSPM') > -1)){
```

```
    deviceId = '02313';
  }

  let xml = "<?xml version='1.0' encoding='utf-8'?><soap12:Envelope
xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance' xmlns:xsd='http://
www.w3.org/2001/XMLSchema' xmlns:soap12='http://www.w3.org/2003/05/
soap-envelope'><soap12:Body><confirmSales xmlns='http://
SACT.SentosaOnlineWebService.Webservices/'><cfmIn>" + data.tickets +
"<PaymentID>" + data.receipt_number + "</PaymentID><CreditCard>" +
data.card_digits + "</CreditCard><DeviceID>" + deviceId + "</
DeviceID><ReservationID>" + data.reservation_id + "</
ReservationID><Name>" + data.name + "</Name><Email>" + data.email + "</
Email><TelNo>" + data.contact + "</TelNo><NRIC></NRIC><Password>" +
config.ticketing.dry.password + "</Password><TransactionID>" +
data.receipt_number + "</
TransactionID><RequestStatus><Status>Successful</Status><Message></
Message></RequestStatus></cfmIn></confirmSales></soap12:Body></
soap12:Envelope>";
  let ticketing_url = config.ticketing.dry.url + "/OnlineWebService.asmx?WSDL="
  request.post({
    url: ticketing_url,
    body: xml,
    agentOptions: {
      rejectUnauthorized: false
    },
    headers: { 'Content-Type': 'text/xml' }
  }, function(err, response, body) {
    if(!err)  {
      let sact_req = response.request
      parser(body, { explicitArray: false }, function(err, results) {
        if(!err)  {
          let confirm_response = results['soap:Envelope']['soap:Body']
['confirmSalesResponse']['confirmSalesResult']
          if(confirm_response['RequestStatus']['Status'] === 'Successful' ||
confirm_response['RequestStatus']['Status'] === 'LastCallSuccessful') {
            // console.log(results['soap:Envelope']['soap:Body'])
            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)
            callback(null, confirm_response)
          } else { // sact confirmation failed
            console.log(results['soap:Envelope']['soap:Body']
['confirmSalesResponse'])
            console.log(' ----- ' + data.reservation_id)
            logger.log('sact', req.headers.correlation_id, null, 'error', { url:
sact_req.uri.href, method: sact_req.method, headers: sact_req.headers, body:
```

```
sact_req.body }, response.statusCode, response.body)
          responseHandler.serverError(res)
          callback(err)
        }
      } else { // error parsing response xml or invalid response
        logger.log('sact', req.headers.correlation_id, null, 'error', { url:
sact_req.uri.href, method: sact_req.method, headers: sact_req.headers, body:
sact_req.body }, response.statusCode, err)
        responseHandler.partnerErrors(res, err)
        callback(err)
      }
    })
  } else { // sact request error
    logger.log('sact', req.headers.correlation_id, null, 'error', { url: ticketing_url,
method: 'POST', headers: {}, body: {} }, 500, err)
    responseHandler.requestErrors(res, err)
    callback(err)
  }
 })
}

exports.processPartialConfirm = (res, req, reservation_id, wirecard, wot_query)
=> {
 let query = {$set:{'status': 'confirmed', 'transaction_id':
req.body.transaction_id, 'issues': 'Partial' }}
 if(wirecard) {
  query = {$set:{'status': 'confirmed', 'transaction_id': req.body.transaction_id,
'card_digits': req.body.digits, 'issues': 'Partial' }}
 }
 Order.findOneAndUpdate(wot_query, query, { __v: 0, _id: 0, reservation_id: 0 },
function(err, updated_order) {
  let attraction = updated_order.attractions[0]
  request.get({
    url: toggleEnv(config.env) + attraction.id
  }, function(err, response, attr_body) {
    if(!err)  {
      attr_body = JSON.parse(attr_body)
      attraction.title = attr_body.data.directory.title
      attraction.thumbnail = attr_body.data.directory.thumbnail
      attraction.redemption_info = attr_body.data.directory.redemption_info
      updated_order.attractions[0] = attraction
      updated_order.issues = 'Partial'
      updated_order = updated_order.toObject()
      updated_order['date_time'] =
moment(updated_order.date_time).format("MMM Do YY, h:mm a")
      Order.findOneAndUpdate(wot_query, {$set: {'attractions': attraction}},
```

```
{ __v: 0, _id: 0, confirmed: 0}, function(err, latest_order) {
        if(!err)  {
          if(updated_order.user_id !== null) {
            // re cache user's orders
            let key = '/ticketing/orders/user/' + updated_order.user_id
            let cachedBody = mcache.get(key)
            if(cachedBody)  { // exists, clear
              remove(key)
            }
          }
          responseHandler.partialConfirmation(res, updated_order)
        } else {
          // send emailer for failed confirmation

          voidWOTSales(req.body.transaction_id, updated_order.pincode.wot,
(err, response) => {
            responseHandler.confirmationUnsuccessful(res, { order_id:
updated_order.order_id })
          })
        }
       })
     } else {
       // send emailer for failed confirmation

       voidWOTSales(req.body.transaction_id, updated_order.pincode.wot, (err,
response) => {
         responseHandler.confirmationUnsuccessful(res, { order_id:
updated_order.order_id })
       })
     }
   })
  })
}

exports.processConfirmedTickets = (res, req, reservation_id, wirecard, order,
confirm_tickets_response, email_type, callback) => {
  if(typeof confirm_tickets_response !== 'undefined') {
    let pincode = confirm_tickets_response['PinCode']
    let pin_expiry = moment().add({months:3}).format("MMM Do YY")
    req.body.pincode = pincode
    const query = { 'reservation_id': Long.fromString(reservation_id.toString()) }
    let update = {$set:{'pincode.pin':pincode, 'pincode.expiry': pin_expiry,
'status': 'reserved', 'transaction_id': req.body.transaction_id, 'card_digits':
String(req.body.digits), 'bank_approval_code': req.body.bank_approval_code,
'payment_type': req.body.payment_type }}
    if(wirecard)  {
```

```
    update = {$set:{'pincode.pin':pincode, 'pincode.expiry': pin_expiry, 'status':
'reserved', 'transaction_id': req.body.transaction_id, 'card_digits':
String(req.body.digits), 'bank_approval_code': req.body.bank_approval_code,
'payment_type': req.body.payment_type }}
  }
  // update order with latest confirmed details
  Order.findOneAndUpdate(query, update, { __v: 0, _id: 0, reservation_id: 0,
status: 0, transaction_id: 0 }, function(err, confirmed_order)  {
    if(!err)  {
      // retrieve rest of attraction details via API calls
      confirmed_order.pincode.pin = pincode
      confirmed_order.pincode.expiry = pin_expiry
      let order_attractions = confirmed_order.attractions
      async.forEachOf(order_attractions, function(attraction, i, callback) {
        let attraction_id = attraction.id
        request.get({
          url: toggleEnv(config.env) + attraction_id
        }, function(err, response, attr_body) {
          if(!err)  {
            attr_body = JSON.parse(attr_body)
            // check item code mappping if != null
            attraction.title = attr_body.data.directory.title
            attraction.thumbnail = attr_body.data.directory.thumbnail
            if(attr_body.data.directory.item_code_mapping !== null)  {
              // fun-pass case; check for item-code-specific redemption_info and
important_notes
              let code = attraction.tickets[0].types[0].code
              // need to check for undefined - item code does not exist in
directories
              if(typeof attr_body['data']['directory']['item_code_mapping'][code] !
== 'undefined') {
                attraction.redemption_info = attr_body['data']['directory']
['item_code_mapping'][code]['redemption_info']
                attraction.important_notes = attr_body['data']['directory']
['item_code_mapping'][code]['important_notes']
              } else {
                attraction.redemption_info = ''
                attraction.important_notes = ''
              }
            } else {
              attraction.redemption_info = attr_body.data.directory.redemption_info
              attraction.important_notes =
attr_body.data.directory.important_notes
            }
          } else {
            voidSACTSales(reservation_id, (sact_err, response) => {
```

```
            logger.log('http', res.req.headers.correlation_id, 'async foreach
attraction details error', 'error', res.req, 200, err)
            callback(err)
           })
          }
          callback()
         })
        }, function(err) { // after looping
         if(!err)  {
          confirmed_order.attractions = order_attractions
          confirmed_order = confirmed_order.toObject()
          confirmed_order['date_time'] =
moment(confirmed_order.date_time).format("MMM Do YY, h:mm a")
          // re-update db with attraction details
          Order.findOneAndUpdate(query, {$set: {'attractions':
confirmed_order.attractions, 'status': 'confirmed' }}, { __v: 0, _id: 0, status: 0},
function(err, latest_order) {
           if(!err)  {
            // re-cache user's orders
            let key = '/ticketing/orders/user/' + order.user_id
            let cachedBody = mcache.get(key)
            if(cachedBody)  { // exists, clear
             remove(key)
            }
            logger.log('cosmodb', res.req.headers.correlation_id, null, 'info',
query, null, order);
            //confirmEmail(latest_order.email, confirmed_order, latest_order,
email_type)
            smtpMail(latest_order.email, confirmed_order, latest_order,
email_type)
            // logger.sales(latest_order.email,
JSON.stringify(latest_order.attractions), latest_order.contact,
JSON.stringify(latest_order.pincode), 'confirmed', latest_order.payment_mode)
            callback(null, confirmed_order)
           } else { // error updating latest orders
            voidSACTSales(reservation_id, (err, response) => {
             logger.log('cosmodb', res.req.headers.correlation_id, null, 'error',
query, null, err);
             responseHandler.confirmationUnsuccessful(res, { order_id:
confirmed_order.order_id })
            })
            console.log("error updating latest orders")
            console.log(err)
           }
          })
         } else {
```

```
            console.log("async for each issue")
            logger.log('http', res.req.headers.correlation_id, 'async foreach
attraction details error', 'error', res.req, 200, err)
            responseHandler.confirmationUnsuccessful(res, { order_id:
order.order_id })
          }
        })
      } else { // updating of latest confirmed details failed
        voidSACTSales(reservation_id, (err, response) => {
          logger.log('cosmodb', res.req.headers.correlation_id, null, 'error', query,
null, err);
          responseHandler.confirmationUnsuccessful(res, { order_id:
confirmed_order.order_id })
        })
        console.log("error updating confirmed details")
        console.log(err)
      }
    })
  } else {
    console.log("error")
  }
}

async function ticketRetryStrategy(err, response, body) {
  let sact_req = response.request
  if(err) {
    return err
  } else {
    await parser(body, { explicitArray: false }, function(err2, results) {
      if(err2) {
        return err2
      } else {
        return false
      }
    })
  }
}
```