271 lines (237 sloc) | 9.64 KB

```php
<?php
/**
 * Copyright (c) 2018 CardGate B.V.
 * All rights reserved.
 * See LICENSE for license details.
 */
namespace Cardgate\Payment\Controller\Payment;

use Cardgate\Payment\Model\GatewayClient;
use Cardgate\Payment\Model\Config\Master;
use Magento\Framework\App\ObjectManager;
use Magento\Sales\Api\Data\TransactionInterface;

/**
 * Callback handler action
 *
 * @author DBS B.V.
 * @package Magento2
 */
class Callback extends \Magento\Framework\App\Action\Action {

	/**
	 *
	 * @var \Magento\Sales\Model\Order\Email\Sender\OrderSender
	 */
	protected $orderSender;

	/**
	 *
	 * @var \Magento\Sales\Model\Order\Email\Sender\InvoiceSender
	 */
	protected $invoiceSender;

	/**
	 *
	 * @var \Magento\Framework\App\Config\ScopeConfigInterface
	 */
	protected $scopeConfig;

	/**
	 *
	 * @var GatewayClient
	 */
	private $_cardgateClient;

	/**
	 *
	 * @var Master
	 */
	private $_cardgateConfig;

	/**
	 *
	 * @var \Magento\Framework\Encryption\Encryptor
	 */
	private $encryptor;

	public function __construct ( \Magento\Framework\App\Action\Context $context, \Magento\Sales\Model\Order\Email\Sender\OrderSender $orderSender, \Magento\Sales\Model\Order\
			\Magento\Framework\App\Config\ScopeConfigInterface $scopeConfig, GatewayClient $client, \Cardgate\Payment\Model\Config $config) {
		$encryptor = ObjectManager::getInstance()->get( \Magento\Framework\Encryption\Encryptor::class );
		parent::__construct( $context );
		$this->invoiceSender = $invoiceSender;
		$this->orderSender = $orderSender;
		$this->scopeConfig = $scopeConfig;
		$this->_cardgateConfig = $config;
		$this->_cardgateClient = $client;
		$this->encryptor = $encryptor;
	}

	/**
	 *
	 * {@inheritdoc}
	 *
	 * @see \Magento\Framework\App\ActionInterface::execute()
	 */
	public function execute () {
		$result = $this->resultFactory->create( \Magento\Framework\Controller\ResultFactory::TYPE_RAW );
		$order = $payment = NULL;
```

```
		$post = $this->getRequest()->getPostValue();
		if ( ! is_array( $post ) ) {
			$post = [];
		}
		$get = $this->getRequest()->getParams();
		if ( ! is_array( $get ) ) {
			$get = [];
		}

		if (!empty($get['cgp_sitesetup']) && !empty($get['token'])) {

			try {
				$bIsTest = ($get['testmode'] == 1 ? true : false);
				$aResult = $this->_cardgateClient->pullConfig($get['token'], $bIsTest);
				$aConfigData = $aResult['pullconfig']['content'];
				$this->_cardgateConfig->setGlobal( 'testmode', $aConfigData['testmode'] );
				$this->_cardgateConfig->setGlobal( 'site_id', $aConfigData['site_id'] );
				$this->_cardgateConfig->setGlobal( 'site_key', $aConfigData['site_key'] );
				$this->_cardgateConfig->setGlobal( 'api_username', $aConfigData['merchant_id'] );
				$this->_cardgateConfig->setGlobal('api_password', $this->encryptor->encrypt($aConfigData['api_key'] ));
				$typeListInterface = ObjectManager::getInstance()->get( \Magento\Framework\App\Cache\TypeListInterface::class );
				$typeListInterface->cleanType('config');
				$sResponse = $this->_cardgateConfig->getGlobal('api_username') . '.' . $this->_cardgateConfig->getGlobal('site_id') . '.200';
				return $this->getResponse()->setBody($sResponse);

			} catch (\Exception $e) {
				return $this->getResponse()->setBody($e->getMessage());
			}
		}

		$transactionId = empty( $post['transaction'] ) ? $this->getRequest()->getParam( 'transaction' ) : $post['transaction'];
		$reference = empty( $post['reference'] ) ? $this->getRequest()->getParam( 'reference' ) : $post['reference'];
		$code = (int)( empty( $post['code'] ) ? $this->getRequest()->getParam( 'code' ) : $post['code'] );
		$currency = empty( $post['currency'] ) ? $this->getRequest()->getParam( 'currency' ) : $post['currency'];
		$amount = (int)( empty( $post['amount'] ) ? $this->getRequest()->getParam( 'amount' ) : $post['amount'] );
		$pt = empty( $post['pt'] ) ? $this->getRequest()->getParam( 'pt' ) : $post['pt'];
		$pmId = ( ! empty( $pt ) ? $pt : 'unknown' );

		$manualProcessing = !!$this->_cardgateConfig->getGlobal( 'manually_process_order' );
		$updateCardgateData = false;
		$payment = null;
		try {
			if ( FALSE == $this->_cardgateClient->transactions()->verifyCallback( empty( $post ) ? $get : $post, $this->_cardgateClient->getSiteKey() ) ) {
				throw new \Exception( 'hash verification failure' );
			}

			$order = ObjectManager::getInstance()->create( \Magento\Sales\Model\Order::class )->loadByIncrementId( $reference );
			$order->addStatusHistoryComment( __( "Update for transaction %1. Received status code %2.", $transactionId, $code ) );


			if ( !$manualProcessing ) {
				$payment = $order->getPayment();
				$updateCardgateData = ! (
					$payment->getCardgateStatus() >= 200
					&& $payment->getCardgateStatus() < 300
				);

				// If the gateway is using a different payment method than us, update the payment method of our order to
				// match the one from the gateway.
				if ( $payment->getCardgatePaymentmethod() != $pmId ) {
					$payment->setCardgatePaymentmethod( $pmId );
					$order->addStatusHistoryComment( __( "Callback received for transaction %1 with payment method '%2' but payment method should be '%3'. Proce
				}
			}

			if ( $code < 100 ) {
				// 0xx pending
				if ( $order->getState() != \Magento\Sales\Model\Order::STATE_NEW ) {
					$order->addStatusHistoryComment( __( 'Transaction already processed.' ) );
				}
			} elseif ( $code < 200 ) {
				// 1xx auth phase
				if ( $order->getState() != \Magento\Sales\Model\Order::STATE_NEW ) {
					$order->addStatusHistoryComment( __( 'Transaction already processed.' ) );
				}
			} elseif ( $code < 300 ) {
				// 2xx success
				if ( $order->getState() == \Magento\Sales\Model\Order::STATE_NEW ) {
					$order->setState(\Magento\Sales\Model\Order::STATE_PROCESSING);
				}
				$order->setStatus( "cardgate_payment_success" );
				$order->addStatusHistoryComment( __( "Transaction success." ) );

				if ( !$manualProcessing ) {
					// Uncancel if needed.
					if ( $order->isCanceled() ) {
						$stockRegistry = ObjectManager::getInstance()->get( \Magento\CatalogInventory\Model\Spi\StockRegistryProviderInterface::class );
						foreach ( $order->getItems() as $item ) {
							$stockItem = $stockRegistry->getStockItem( $item->getProductId(), $order->getStore()->getWebsiteId() );
							$stockItem->setQty( $stockItem->getQty() - $item->getQtyCanceled() );
							$stockItem->save();

							$item->setQtyCanceled( 0 );
							$item->setTaxCanceled( 0 );
							$item->setDiscountTaxCompensationCanceled( 0 );
							$item->save();
						}
						$order->addStatusHistoryComment( __( 'Transaction rebooked. Product stock reclaimed from inventory.' ) );
```

```
                                        }

                                        // Test if transaction has been processed already.
                                        $paymentRepository = ObjectManager::getInstance()->get( \Magento\Sales\Model\Order\Payment\Transaction\Repository::class );
                                        $currentTransaction = $paymentRepository->getByTransactionId( $transactionId, $payment->getId(), $order->getId() );
                                        if (
                                                ! empty( $currentTransaction )
                                                && $currentTransaction->getTxnType() == TransactionInterface::TYPE_CAPTURE
                                        ) {
                                                $order->addStatusHistoryComment( __( 'Transaction already processed.' ) );
                                                $updateCardgateData = FALSE;
                                                throw new \Exception( 'transaction already processed.' );
                                        }

                                        // Test if payment has been processed already.
                                        if (
                                                $payment->getCardgateStatus() >= 200
                                                && $payment->getCardgateStatus() < 300
                                        ) {
                                                $order->addStatusHistoryComment( __( 'Payment already processed in another transaction.' ) );
                                                $updateCardgateData = FALSE;
                                                throw new \Exception( 'payment already processed in another transaction.' );
                                        }

                                        // Do capture.
                                        $payment->setTransactionId( $transactionId );
                                        $payment->setCurrencyCode( $currency );
                                        $payment->registerCaptureNotification( $amount / 100 );
                                        $payment->setMethod( 'cardgate_' . $pt );

                                        if ( ! $order->getEmailSent() ) {
                                                $this->orderSender->send( $order );
                                        }

                                        $invoice = $payment->getCreatedInvoice();
                                        if ( ! empty( $invoice ) ) {
                                                $invoice->save(); // makes sure there's an invoice id generated
                                                $this->invoiceSender->send( $invoice );
                                        } else {
                                                $order->addStatusHistoryComment( __( 'Failed to create invoice.' ) );
                                                throw new \Exception( 'failed to create invoice.' );
                                        }
                                }
                        } elseif ( $code < 400 ) {
                                // 3xx error
                                if ( !$manualProcessing ) {
                                        try {
                                                $order->registerCancellation( __( 'Transaction canceled.' ), false );
                                                $order->setStatus( "cardgate_payment_failure" );
                                                $order->addStatusHistoryComment( __( "Transaction failure." ) );
                                        } catch ( \Exception $e ) {
                                                $order->addStatusHistoryComment( __( "Failed to cancel order. Order state was : %1.", $order->getState() . '/' . $order->getStatus(
                                                throw new \Exception( 'failed to cancel order.' );
                                        }
                                }
                        } elseif ( $code < 500 ) {
                                // 4xx refund
                                if ( !$manualProcessing ) {
                                        $order->registerCancellation( __( "Transaction refund received. Amount %1.", $currency . ' ' . round( $amount / 100, 2 ) ) );
                                }
                        } elseif (
                                $code >= 600
                                && $code < 700
                        ) {
                                // 6xx notification from bank
                        } elseif ( $code < 800 ) {
                                // 7xx waiting for confirmation
                        }

                        // Set the output to a string that the gateway expects.
                        $result->setContents( $transactionId . '.' . $code );

                } catch ( \Exception $e ) {

                        // Add the exception message to the output.
                        $result->setContents( $e->getMessage() );
                }

                if (
                        $payment != NULL
                        && $updateCardgateData
                ) {
                        $payment->setCardgateStatus( $code );
                        $payment->setCardgateTransaction( $transactionId );
                        $payment->save();
                }

                if ( $order != NULL ) {
                        $order->save();
                }

                return $result;
        }

}
```