Wirecard CEE Integration Documentation

() -

Created: 2019-09-20 16:12

Code Examples for Toolkit (PHP), JAPI and Wirecard PHP Payment SDK

These code examples provide guidance during the switching process from the Wirecard Checkout Enterprise legacy stack to the new Wirecard Payment Gateway stack.

Wirecard Checkout Enterprise

These code examples for credit card transactions show how the core features are implemented, using Toolkit and JAPI.

Toolkit (PHP)

	Toolkit (PHP) Code Sample

Configuration Setup	`$JAVA_HOME = "<PUT YOUR JAVA HOME DIRECTORY HERE>";`
	
	`$VOY_ROOTPATH = "<PUT YOUR QTILL INSTALLATION DIRECTORY HERE>";`
	
	`$VOY_CLASSPATH = "$VOY_ROOTPATH/voyager.zip -Dqtill.properties=$VOY_ROOTPATH/qtill.properties -Dfile.encoding=ISO-8859-1";`
	
	`$VOY_MERCHANTKEY = "<PUT YOUR MERCHANTKEY HERE>";`
	
	`$VOY_ADMINPASSWORD = "<PUT YOUR ADMINPASSWORD HERE>";`
	
	`$VOY_LANGUAGE = "de";`

Initiate Payment	`// Initiated in Frontend via WCS/WCP or JAPI/Serial API call.`
3-D Secure	`// Initiated in Frontend via WCS/WCP or JAPI/Serial API call.`

Deposit	`//Deposit refers to previously created order and initialization with specific order number.`
	
	`$orderNumber = 123123;`
	`$paymentNumber = $orderNumber;`
	`$amount = "1,23";`
	`$isocc = "EUR";`
	
	`$status = deposit($orderNumber, $paymentNumber, $amount, $isocc, $message);`

Recurring	`$oldOrderNumber = 123123;`
	`$newOrderNumber = -1;`
	
	`$amount = "1,23";`
	`$isocc = "EUR";`
	
	`$orderDescription = "Test transaction Wirecard CEE.";`
	
	`// qtill-function createOrder to obtain a not yet used order number`
	`$orderStatus = createOrder($newOrderNumber, $message);`
	
	`// if order created successfully, recurring payment can be initiated`
	`$recurStatus = recurPayment($oldOrderNumber, $newOrderNumber, $amount, $isocc, $orderDescription,`

Refund	`$oldOrderNumber = 123123;`
	
	`$newOrderNumber = -1;`
	
	`$amount = "1.23";`
	`$isocc = "EUR";`
	
	`$orderDescription = "Test transaction Wirecard CEE.";`
	
	`// qtill-function createOrder to obtain a not yet used order number`
	`$orderStatus = createOrder($newOrderNumber, $message);`
	
	`// if order created successfully, recurring payment can be initiated`
	`$recurStatus = recurPayment($oldOrderNumber, $newOrderNumber, $amount, $isocc, $orderDescription, $message);`

Approve Reversal	`// Ordernumber indicates which approval is reversed`
	`$orderNumber = 123123;`
	`$paymentNumber = $orderNumber;`
	
	`// approve reversal transaction initiated`
	`$status = approveReversal($orderNumber, $paymentNumber, $message);`

Deposit Reversal	`// Ordernumber indicates which deposit is reversed`
	`$orderNumber = 123123;`
	`$paymentNumber = $orderNumber;`
	
	`// Deposit reversal transaction initiated`
	`$status = depositReversal($orderNumber, $paymentNumber, $message);`

JAPI

	JAPI Code Sample

Configuration Setup	`QTillSettings settings = new QTillSettings(new java.io.File("/path/to/qtill.properties"));`
	
	`QTillInstance qtill = new QTillInstance("[your Merchant Key]", "[your Toolkit-Password]", settings);`
	
	`QTillInstance qtill = new QTillInstance();`

Initiate Payment	`QTillAmount amount = new QTillAmount("1.23", new QTillCurrency("EUR"));`
	
	`// initiation of credit card`
	`QTillDate expiry = new QTillDate(1, 2019);`
	`String pan = "9400000000000003";`
	
	`QTillCreditcardInitiation initiation = new QTillCreditcardInitiation(pan, expiry);`
	`initiation.setCardVerifyCode(new QTillNumericToken("003"));`
	`initiation.setCardholderName("John Doe");`
	`initiation.setClientIPAddress("1.2.3.4");`
	
	`int orderNumber = qtill.generateOrderNumber();`
	
	`// create order reference`
	`QTillOrderReference orderReference = new QTillOrderReference(orderNumber, "Test transaction Wirecard CEE.");`
	
	`// initiate payment`
	`qtill.initiatePayment(new QTillInitiateDto(orderReference, amount, initiation));`

3-D Secure	`// generate order number and reference`
	`int orderNumber = qtill.generateOrderNumber();`
	`orderReference = new QTillOrderReference(orderNumber, orderDescription);`
	
	`// initiate 3D payment`
	`initiation = new QTill3DSecureInitiation(returnUrl, serviceUrl, userAgent, acceptHeaders, MCSC_PAN, expiry);`
	`qtill.initiatePayment(new QTillInitiateDto(this.orderReference, this.amount, this.initiation));`
	
	`// Customer is redirected to 3D-Secure-Page`
	`// PaRes and MD are returned values from access control server`
	`String wumPaRes = "[from ACS returned parameter PaRes]";`
	`String wumMD = "[from ACS returned parameter MD]";`
	
	`// take the orderNumber from the initiation for validation`
	`QTill3DSecureValidationData validationData = new QTill3DSecureValidationData(wumPaRes, wumMD);`
	`validationData.setSupposedOrderNumber(orderNumber);`
	
	`QTillPayment payment = qtill.verifyPayment(validationData);`

Deposit	`// generate new order`
	`int orderNumber = qtill.generateOrderNumber();`
	`orderReference = new QTillOrderReference(orderNumber, orderDescription);`
	`QTillAmount amount = new QTillAmount("1.23", QTillCurrency.EUR);`
	
	`// verify payment`
	`QTillPayment payment = qtill.verifyPayment(orderNumber);`
	`int paymentNumber = payment.getPaymentNumber();`
	
	`// Deposit`
	`qtill.deposit(new QTillDepositDto(orderNumber, paymentNumber, amount));`

Recurring	`int orderNumber = qtill.generateOrderNumber();`
	
	`// set the orderNumber of an order already processed and`
	`// to be recurred now`
	`int oldOrderNumber = 123123;`
	
	`// create the amount`
	`QTillAmount amount = new QTillAmount("1.23", QTillCurrency.EUR);`
	
	`// create reference for recurring order`
	`QTillOrderReference orderRef = new QTillOrderReference(orderNumber, ORDER_DESCRIPTION);`
	
	`// recur the payment`
	`qtill.recurPayment(new QTillRecurDto(oldOrderNumber, orderRef, amount, false, null, null, null, null));`

Refund	`// get order number of order to be refunded`
	`int orderNumber = 123123;`
	
	`// Set refund amount`
	`QTillAmount refundAmount = new QTillAmount("1.23", QTillCurrency.EUR);`
	
	`// Refund amount`
	`qtill.refund(new QTillRefundDto(refundAmount).setOrderNumber(orderNumber). setMerchantReference("MerchRefNo"));`

Approve Reversal	`// Generate new order`
	`int orderNumber = this.qtill.generateOrderNumber();`
	`QTillAmount amount = new QTillAmount("1.23", QTillCurrency.EUR);`
	`orderReference = new QTillOrderReference(orderNumber, this.orderDescription);`
	
	`// Verify payment`
	`QTillPayment payment = qtill.verifyPayment(orderNumber);`
	
	`// Reverse Approval`
	`qtill.approveReversal(orderNumber,payment.getPaymentNumber());`

Deposit Reversal	`// Generate new order`
	`int orderNumber = qtill.generateOrderNumber();`
	`orderReference = new QTillOrderReference(orderNumber, orderDescription);`
	`QTillAmount amount = new QTillAmount("1.23", QTillCurrency.EUR);`
	
	`// Verify payment`
	`QTillPayment payment = qtill.verifyPayment(orderNumber);`
	`int paymentNumber = payment.getPaymentNumber();`
	
	`// Deposit`
	`qtill.deposit(orderNumber, paymentNumber, amount);`
	
	`// Reverse Deposit`
	`qtill.depositReversal(orderNumber, paymentNumber);`

Wirecard Payment Gateway

Wirecard PHP Payment SDK

Wirecard PHP Payment SDK simplifies the REST API usage and facilitates its implementation. However, it offers less functionality than addressing the REST API directly. Its original purpose was to simplify the development of Wirecard Shop Extensions.

The Wirecard PHP Payment SDK functions cannot be used as a replacement for the Wirecard Checkout Enterprise solutions, but may be helpful during the transition phase. They show how Wirecard developers implement certain features.
This table shows an overview of the available functions for credit card transactions.
Click here for the source code and detailed examples on GitHub.

	Wirecard PHP Payment SDK Code Sample

Configuration Setup

```
// Add global config data
$baseUrl = 'https://api-test.wirecard.com';
$httpUser = '70000-APITEST-AP';
$httpPass = 'qD2wzQ_hrc!8';


$config = new Config\Config($baseUrl, $httpUser, $httpPass,
'EUR');


// add setup for each payment type
 $creditcardConfig = new CreditCardConfig();


// example config merchant ID and key for non 3d creditcard
$creditcardConfig->setNonThreeDCredentials(
    '53f2895a-e4de-4e82-a813-0d87a10e55e6',
    'dbc5a498-9a66-43b9-bf1d-a618dd399684'
);


$config->add($creditcardConfig);
```

Initiate Payment

```
// Based on already existing tokenID for credit card.
// Initiated in frontend with javascript

// The credit card transaction contains all relevant data for
the payment process.

$transactionService = new TransactionService($config);

//Initiate new credit card transaction
$transaction = new CreditCardTransaction();
$transaction->setAmount($amount);

$transaction->setTokenId($tokenId);

$transaction->setTermUrl($redirectUrl);

//Reserve transaction
$response = $transactionService->reserve($transaction);
//For payment methods without capturing pay() initiates the
payment

$transactionService = new TransactionService($config);
$transaction = new CreditCardTransaction();
$transaction->setParentTransactionId($parentTransactionId)
$transaction->setAmount($amount);

$response = $transactionService->pay($transaction);
```

3-D Secure	`// Amounts larger than threeDMinLimit and smaller or equal // nonThreeDLimit will first be tried as 3-D-Secure transaction $creditcardConfig->addNonThreeDMaxLimit(new Amount(100.0, 'EUR')); $creditcardConfig->addThreeDMinLimit(new Amount(50.0, 'EUR'));`
	
	`// The credit card transaction contains all relevant data for the payment process. $creditcardConfig->setThreeDCredentials( '508b8896-b37d-4614-845c-26bf8bf2c948', 'dbc5a498-9a66-43b9-bf1d-a618dd399684' ); $config->add($creditcardConfig);`
	
	`// The credit card transaction contains all relevant data for the payment process. $transactionService = new TransactionService($config);`
	
	`//Initiate new credit card transaction $transaction = new CreditCardTransaction(); $transaction->setAmount($amount); $transaction->setTokenId($tokenId); $transaction->setTermUrl($redirectUrl); //Reserve or pay transaction $response = $transactionService->reserve($transaction);`
	
	`//Automatic redirect to 3D-Secure-Page initiated`

Deposit	`//Initiate new transaction for capture-authorization $transactionService = new TransactionService($config);`
	
	`$transaction = new CreditCardTransaction(); $transaction->setParentTransactionId($parentTransactionId) $transaction->setAmount($amount);`
	
	`//Capture-authorization based on reserve transaction $response = $transactionService->pay($transaction);`

Recurring	`// The credit card transaction contains all relevant data for the payment process. $transaction = new CreditCardTransaction(); $transaction->setAmount($amount);`
	
	`// Token ID is necessary for recur purchase with credit card via token. $transaction->setTokenId($tokenId);`
	
	`// The service is used to execute the payment operation itself. $transactionService = new TransactionService($config);`
	
	`$response = $transactionService->pay($transaction);`

Refund	`// parent transaction id needs to be defined`
	`$transaction = new CreditCardTransaction();`
	`$transaction->setParentTransactionId($parentTransactionId]);`
	
	`// cancel leads to void or refund`
	`// depending on current status of transaction`
	`$transactionService = new TransactionService($config);`
	
	`$response = $transactionService->cancel($transaction);`
	`// credit leads to payout to customer for credit card`
	`$accountHolder = new AccountHolder();`
	`$accountHolder->setLastName('Doe');`
	
	`$transaction = new CreditCardTransaction();`
	`$transaction->setAmount($amount);`
	
	`// corresponding token ID for account holder is required`
	`$transaction->setTokenId($tokenId);`
	`$transaction->setAccountHolder($accountHolder);`
	
	`$transactionService = new TransactionService($config);`
	
	`$response = $transactionService->credit($transaction);`

Approve Reversal	`// parent transaction id needs to be defined`
	`$transaction = new CreditCardTransaction();`
	`$transaction->setParentTransactionId($parentTransactionId]);`
	
	`// cancel leads to void or refund depending on current status of transaction`
	`$transactionService = new TransactionService($config);`
	`$response = $transactionService->cancel($transaction);`

Deposit Reversal	`// parent transaction id needs to be defined`
	`$transaction = new CreditCardTransaction();`
	`$transaction->setParentTransactionId($parentTransactionId]);`
	
	`// cancel leads to void or refund depending on current status of transaction`
	`$transactionService = new TransactionService($config);`
	`$response = $transactionService->cancel($transaction);`

Back to Switching to Wirecard Payment Gateway main page.