**int\_wirecard**

***Version <19.1.0>***



Inhaltsverzeichnis

[1. Summary 3](#_Toc10727839)

[2. Component Overview 3](#_Toc10727840)

[2.1 Functional Overview 3](#_Toc10727841)

[2.2 Compatibility 3](#_Toc10727842)

[3 Implementation guide 3](#_Toc10727843)

[3.1 Setup 3](#_Toc10727844)

[3.2 Configuration 4](#_Toc10727845)

[3.2.1 Add cartridges to site(s) 4](#_Toc10727846)

[3.2.2 Import metadata 4](#_Toc10727847)

[3.3 Custom code 9](#_Toc10727848)

[3.3.1 forms/default/billing.xml 9](#_Toc10727849)

[3.3.2 js/pages/checkout/index.js 9](#_Toc10727850)

[3.3.3 js/pages/checkout/billing.js 10](#_Toc10727851)

[3.3.4 js/checkout/billing/paymentgateway.js 10](#_Toc10727852)

[3.3.5 scripts/util/Resource.ds 11](#_Toc10727853)

[3.3.6 template/default/checkout/billing/paymentmethods.isml 11](#_Toc10727854)

[3.3.7 template/default/checkout/summary/summary.isml 12](#_Toc10727855)

[3.3.8 template/default/checkout/pt\_checkout\_UI.isml 12](#_Toc10727856)

[3.3.9 controllers/COBilling.js 13](#_Toc10727857)

[3.3.10 controllers/COPlaceOrder.js 13](#_Toc10727858)

[3.3.11 js/pages/checkout/billing.js 15](#_Toc10727859)

[4 Maintenance 15](#_Toc10727860)

[4.1 Custom site preferences 15](#_Toc10727861)

[4.2 Jobs 15](#_Toc10727862)

[4.3 Services 15](#_Toc10727863)

[5 Support 15](#_Toc10727864)

[6 Release History 17](#_Toc10727865)

# Summary

Wirecard AG is one of the world’s leading independent providers of outsourcing and white label solutions for electronic payment transactions. The Wirecard Group has been supporting companies in accepting electronic payments from all sales channels. A global multi-channel platform bundles international payment acceptances and methods, supplemented by fraud prevention solutions. When it comes to issuing their own payment instruments in the form of cards or mobile payment solutions, Wirecard provides companies with an end-to-end infrastructure, including the requisite licenses for card and account products.

# Component Overview

## Functional Overview

The wirecard sales force commerce cloud cartridge provides the integration of the following payment methods

* Credit Card
* PayPal
* SEPA Direct Debit
* Sofort.

Depending on the selected payment method and the custom site preferences the cartridge allows several post-order processes: cancelling, refunding and capturing transactions.

## Compatibility

Compatible to Salesforce commerce cloud 19.6 and higher.

# Implementation guide

## Setup

For the integration of wirecard in a sitegenesis storefront you have to import the cartridges int\_wirecard and int\_wirecard\_controllers.

In order to deploy the cartridges, import the cartridges into UX Studio by following these steps:

1. In UX Studio select from the menu File 🡪 Import and choose General 🡪 Existing Projects into Workspace then click next.
2. In the next window make sure the checkbox Select archive file: is selected and click on the Browse button on the right.
3. Select the archive file containing the integration and click Open then Finish.
4. If you have an active commerce cloud server connection the Studio will ask you to link the cartridge to it. Click on Yes.
5. To manually assign and deploy the cartridge to a server connection right click on it and click on Properties. Then select Project References and check the project containing the integration. The cartridge should now be automatically uploaded.

## Configuration

### Add cartridges to site(s)

To activate the integration in SiteGenesis and for the business manager funcionality, the cartridge has to be assigned to the storefront site and the bm site in the Business Manager. In order to do this, follow these steps:

1. Login to Business Manager and click on Administration.
2. Click on Sites and then Manage Sites.
3. Click on the site name and then on the top tab menu on Settings.
   1. Sitegenesis: In the text field Cartridges prepend this text: “**int\_wirecard\_controllers:int\_wirecard\_core**”
   2. bm site: in the text field Cartridges prepend this text: “**bm\_wirecard:int\_wirecard\_core**”
4. Repeat steps 3 to 4 for each Instance Type by selecting the appropriate type from the drop-down menu (Production, Staging, Sandbox / Development).

### Import metadata

For a manual deployment of the metadata just open a terminal and run zip -rq metadata.zip metadata/ from within the module’s main directory. After the creation of the metadata zip file open the business manager and upload / import the metadata.zip in Administration > Site Development > Site Import & Export.

* + - 1. Service wirecard.paymentgateway

If not imported as described in 3.2.2 check the package’s metadata/service.xml for the service configuration / profile / credentials and create a new service.

* + - 1. Jobs

There are 2 jobs coming with the package:

* PaymentGatewayHandleNotifications
* PaymentGatewayDeleteProcessedNotifications

When notifications are received from the paygate by the dedicated controller actions (e.g. PaymentGateway-Notify) a new custom object “PaymentGatewayNotification” gets created with the notification data. Depending on the job configuration these custom objects will be handled on a regular basis (by default every 5min). After the appropriate action has been performed for each notification (custom object), the notification will be archived – that is another custom object will be created (PaymentGatewayProcessedNotification) which then holds the data from its predecessor.

The 2nd job will then delete all processed notifications (custom object).

Check metadata/jobs.xml about job parameters etc. when you decide to create them manually.

* + - 1. Custom Objects

There are 2 types of custom objects already mentioned in the section above together with its particular purpose:

* PaymentGatewayNotification
* PaymentGatewayProcessedNotification

Please check metadata/meta/wirecard-object-extensions.xml for the objects’ properties.

* + - 1. System object custom attributes

**dw.order.Basket**

|  |  |  |
| --- | --- | --- |
| Property name | Type | Description |
| paymentGatewayReservedOrderNo | String | When generating the request data before rendering the seamless / embedded credit card form, an order number will be created / reserved for the current basket and later be used to create an order when the customer finalizes the checkout. |

**dw.order.Order**

|  |  |  |
| --- | --- | --- |
| Property name | Type | Description |
| paymentGatewayTransactions | Set-of-string | Property where payment gateway transaction data (received as response / notification) is saved |
| paymentGatewayCapturedAmount | String | Sum of amount that has already been captured by (subsequent) post-order processes |
| paymentGatewayRefundedAmount | String | Sum of amount that has been refunded |
| paymentGatewayOrderState | Enum-of-string | The value of this property will be set according to the transaction type of the last successful transaction |

**dw.order.OrderPaymentInstrument**

|  |  |  |
| --- | --- | --- |
| Property name | Type | Description |
| paymentGatewayFingerprint | String | Holds hash calculated out of order / basket data / url salt to check incoming notifications for their integrity |
| paymentGatewaySEPADebtorName | String | Debtor name for SEPA direct debit |
| paymentGatewaySEPAIBAN | String | Iban used for SEPA direct debit |
| paymentGatewaySEPABIC | String | BIC used for SEPA direct debit |

**dw.system.SitePreferences**

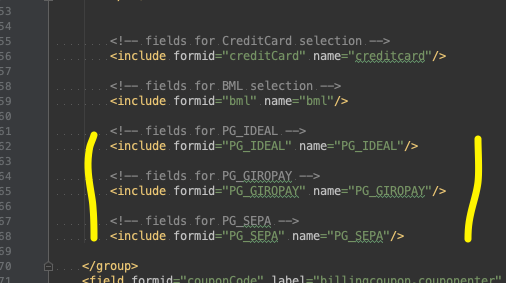
|  |  |  |
| --- | --- | --- |
| Property name | Type | Description |
| paymentGatewayDebugMode | Boolean | If set to “Yes” api communication will be logged to a custom info log. |
| paymentGatewayShopName | String | Used for creating the “descriptor” parameter |
| paymentGatewayNotificationEmail | String | E-Mail address where notification emails will be sent to |
| paymentGatewayModuleVersion | String | The module’s current version |
| paymentGatewaySupportEmail | String | E-Mail address used for sending support requests from the support form in the business manager |
| paymentGatewayUrlSalt | String | Used for calculating basket fingerprint |
| *Preferences for PG\_SOFORT all preferences are prefixed with “paymentGateway”* | | |
| SofortBaseUrl | String | Base url for api communication |
| SofortMerchantAccountID | String | MAID for Sofort. |
| SofortHttpUser | String | Username for basic auth |
| SofortHttpPassword | String | Password for basic auth |
| SofortSendAdditionalData | Boolean | If “yes” order data, billing / shipping address will be included in request sent to payment gateway |
| SofortSecret | String | Used for checking request integrity |
| *Preferences for PG\_PAYPAL all preferences are prefixed with “paymentGateway”*  *\* for description compare with analogue preference for Sofort.* | | |
| PayPalBaseUrl | String | \* |
| PayPalMerchantAccountID | String | \* |
| PayPalSecret | String | \* |
| PayPalHttpUser | String | \* |
| PayPalHttpPassword | String | \* |
| PayPalAddDescriptorToRequest | Boolean | If “yes” parameter “descriptor” will be included in request to payment gateway |
| PayPalSendAdditionalData | Boolean | \* |
| PayPalSendBasketData | Boolean | If “yes” request will contain information about order items |
| PayPalInitialTransactionType | Enum-of-string | Payment Action: either “authorization” or “purchase” |
| *Preferences for PG\_SEPA (Direct debit) all preferences are prefixed with “paymentGateway”*  *\* for description compare with analogue preference for Sofort. / PayPal* | | |
| SEPADebitMerchantAccountID | String | \* |
| SEPADebitHttpUser | String | \* |
| SEPADebitHttpPassword | String | \* |
| SEPADebitBaseUrl | String | \* |
| SEPADebitSecret | String | \* |
| SEPADebitInitialTransactionType | Enum-of-string | \* |
| SEPADebitShowBicInCheckout | Boolean | If “yes” form in checkout payment step will hold an input for the BIC |
| SEPADebitMandateCreditorID | String | For the merchant’s creditor id |
| SEPADebitMandateName | String | Pls refer to wirecard doc center for further information |
| SEPADebitMandateStreet | String |
| SEPADebitMandateZIP | String |
| SEPADebitMandateCity | String |
| SEPADebitMandateCountry | String |
| SEPADebitMandateText | Text |
| SEPADebitMandateTextAccept | String |
| *Preferences for SEPA Credit all preferences are prefixed with “paymentGateway”*  *\* for description compare with analogue preference for Sofort. / PayPal* | | |
| SEPACreditMerchantAccountID | String | \* |
| SEPACreditHttpUser | String | \* |
| SEPACreditHttpPassword | String | \* |
| SEPACreditBaseUrl | String | \* |
| SEPACreditSecret | String | \* |
| *Preferences for PG\_CREDITCARD*  *all preferences are prefixed with “paymentGateway”*  *\* for description compare with analogue preference for Sofort. / PayPal* | | |
| CreditCardMerchantAccountID | String | \* |
| CreditCardHttpUser | String | \* |
| CreditCardHttpPassword | String | \* |
| CreditCardBaseUrl | String | \* |
| CreditCardSecret | String | \* |
| CreditCardMerchantAccountID3DS | String | MAID for 3ds transactions |
| CreditCardMerchantSecret3DS | String | Secret used for 3ds transactions |
| CreditCardUse3DS | Boolean | If “yes”, 3ds transactions will be made if other pre-conditions are met: |
| CreditCard3DSMinLimit | String | Amount in current site’s default currency. If exceeded and 3ds is enabled, request will hold 3ds parameters |
| CreditCardConversionRates | Set-of-string | Used to calculated 3ds-Min-Limit for currency other than the site’s default one |
| CreditCardInitialTransactionType | Enum-of-string | \* |
| *Preferences for iDEAL all preferences are prefixed with “paymentGateway”*  *\* for description compare with analogue preference for Sofort. / PayPal* | | |
| IdealMerchantAccountID | String | \* |
| IdealHttpUser | String | \* |
| IdealHttpPassword | String | \* |
| IdealBaseUrl | String | \* |
| IdealSendAdditionalData | Boolean | \* |
| IdealSecret | String | \* |
| *Preferences for Giropay all preferences are prefixed with “paymentGateway”*  *\* for description compare with analogue preference for Sofort. / PayPal* | | |
| GiropayMerchantAccountID | String | \* |
| GiropayHttpUser | String | \* |
| GiropayHttpPassword | String | \* |
| GiropayBaseUrl | String | \* |
| GiropaySecret | String | \* |
| GiropaySendAccitionalData | Boolean | \* |

## Custom code

All necessary customizations for the integration in a sitegenesis frontend are provided with the package’s app\_storefront\_core and app\_storefront\_controllers cartridges.

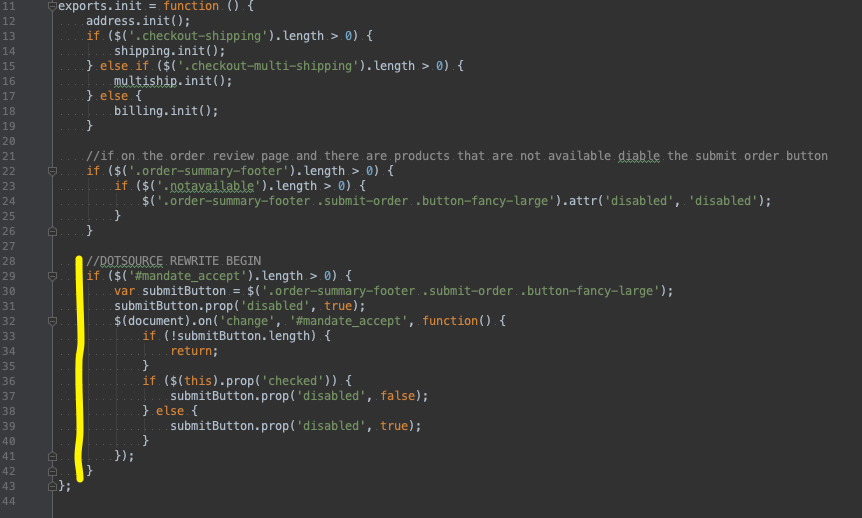
Please check these cartridges for the below described modifications.

### forms/default/billing.xml

To integration the necessary form fields for the Wirecard payment methods you will need to modify your storefront’s billing.xml  


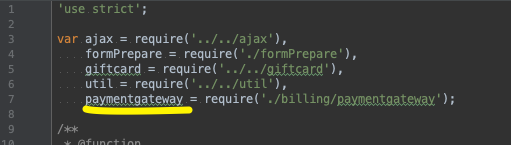
### js/pages/checkout/index.js

For the payment method PG\_SEPA you have to extend the script’s init method:



### js/pages/checkout/billing.js

This frontend js will be extended to require the paymentgateway.js in the checkout:



And within the init method:

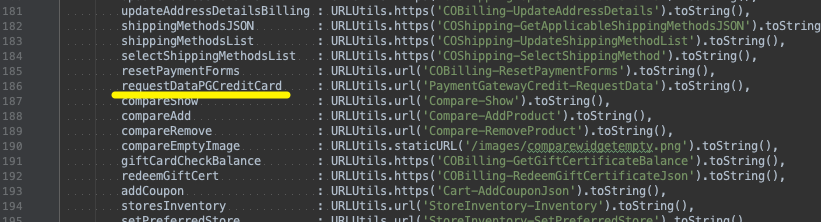


### js/checkout/billing/paymentgateway.js

This frontend js has to be copied to your storefront cartridge. Mind the correct script path!

### scripts/util/Resource.ds

Here you need to add a new property to the getUrls method:

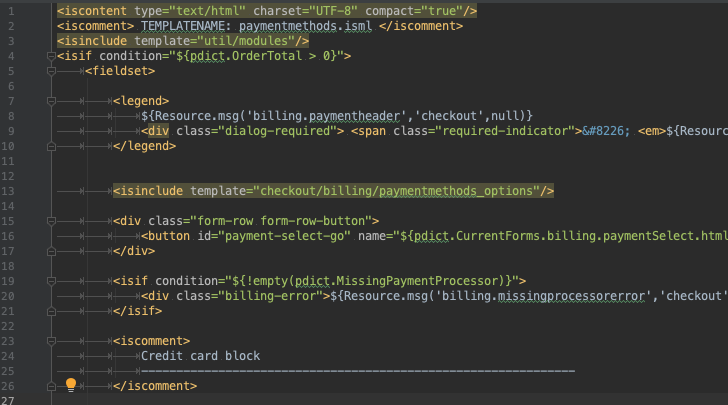


This is the url later used by the embedded payment form when requesting the payment gateway request data via ajax before rendering the credit card form on the order overview page.

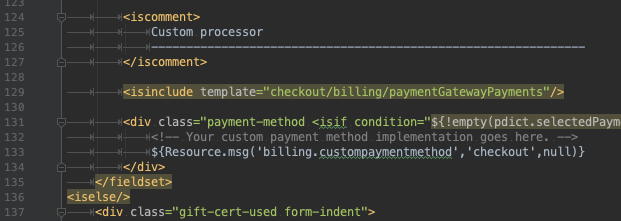
### template/default/checkout/billing/paymentmethods.isml

For having the wirecard payment options available in the billing step, you will have to replace the complete <div class=”payment-method-options form-indent”>..</div> with just one line: <isinclude template=”checkout/billing/paymentmethods\_options”/>

The result should look like this:



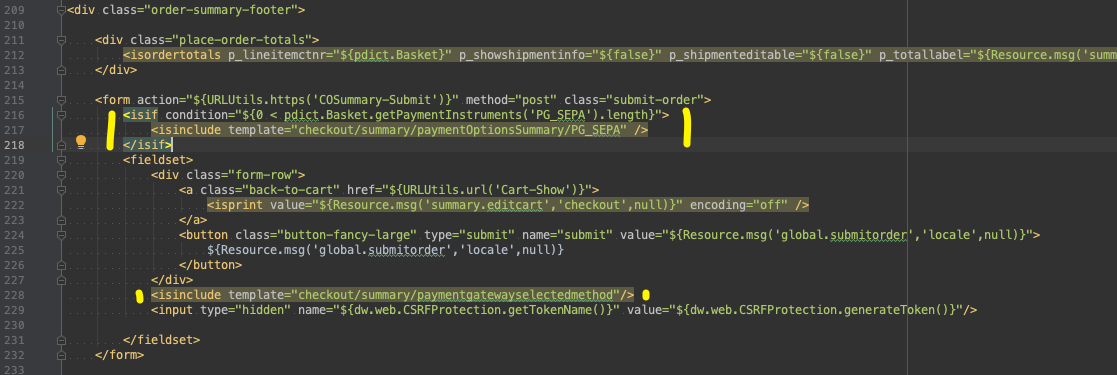
At the end of the same file you will have to include another template:



<isinclude template=”checkout/billing/paymentGatewayPayments”/>

### template/default/checkout/summary/summary.isml

Add 2 isincludes to the form close to the end of that template:



The 1st is for including a checkbox for the payment method PG\_SEPA:

*<isif condition=”${ 0 < pdict.Basket.getPaymentInstruments(‘PG\_SEPA’).length}”>  
 <isinclude template=”${ checkout/summary/paymentOptionsSummary/PG\_SEPA”/>  
</isif>*

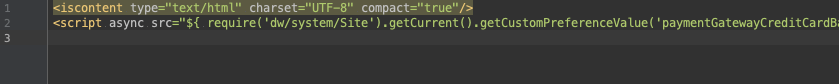
The 2nd is to add a hidden input for the payment method PG\_CREDITCARD:

*<isinclude template=”checkout/summary/paymentgatewayselectedmethod”/>*

Please refer also to the sample file provided with the package in app\_storefront\_core for a correct integration.

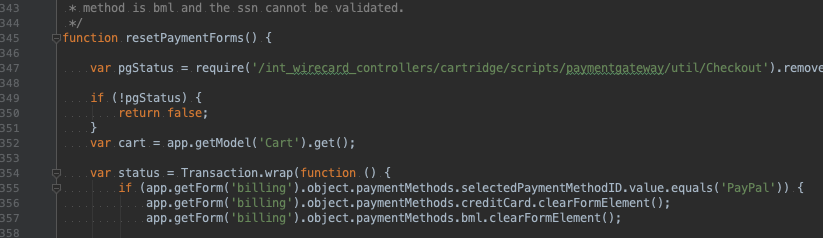
### template/default/checkout/pt\_checkout\_UI.isml

As well as with the summary template, you will have to add a script tag as shown below to the pt\_checkout\_UI template if you plan to use the payment method PG\_CREDITCARD



### controllers/COBilling.js

Within the method “resetPaymentForms” append the following at the beginning:

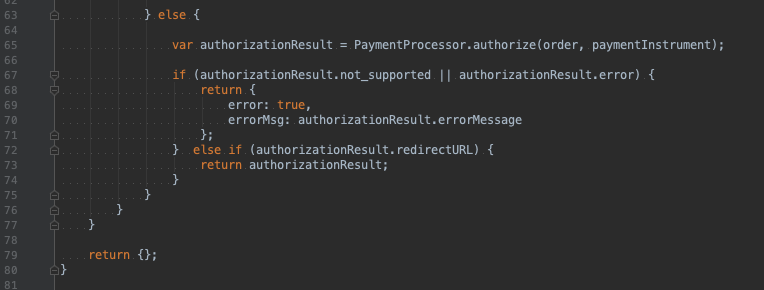


*var pgStatus = require('/int\_wirecard\_controllers/cartridge/scripts/paymentgateway/util/Checkout').removePaymentInstrumentsFromBasket();  
  
if (!pgStatus) {  
 return false;  
}*

This will remove the payment gateway payment methods from the cart when initializing the billing step.

### controllers/COPlaceOrder.js

Within the method **handlePayments** append the following:



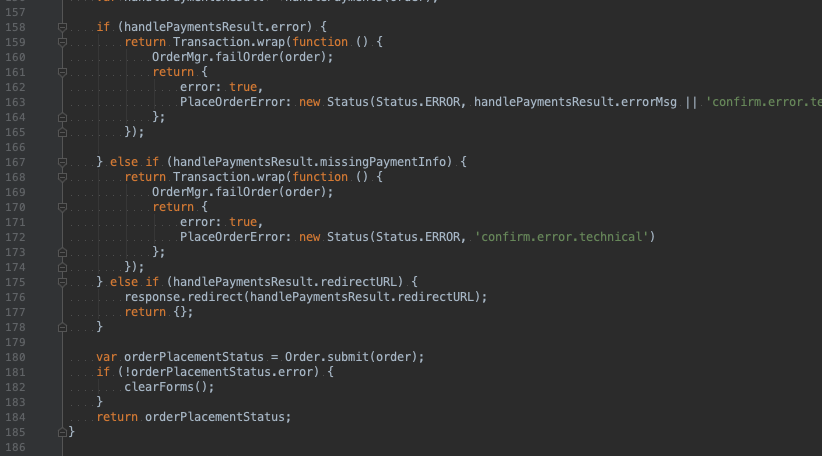
In the case when the initial request to the wpg triggers a redirect to a payment page, the authorization result object will contain a url which will be used in the frontend as redirect target for the customer.

*} else if (authorizationResult.redirectURL) {  
 return authorizationResult;  
}*

In the method start you will have to make the following changes in the control structure if (handlePaymentsResult.error):

*if (handlePaymentsResult.error) {  
 return Transaction.wrap(function () {  
 OrderMgr.failOrder(order);  
 return {  
 error: true,  
 PlaceOrderError: new Status(Status.ERROR, handlePaymentsResult.errorMsg || 'confirm.error.technical')  
 };  
 });  
  
} else if (handlePaymentsResult.missingPaymentInfo) {  
 return Transaction.wrap(function () {  
 OrderMgr.failOrder(order);  
 return {  
 error: true,  
 PlaceOrderError: new Status(Status.ERROR, 'confirm.error.technical')  
 };  
 });  
} else if (handlePaymentsResult.redirectURL) {  
 response.redirect(handlePaymentsResult.redirectURL);  
 return {};  
}*

The result should look like this:



### js/pages/checkout/billing.js

Copy the frontend js from with the package’s app\_storefront\_core/cartridge/js/pages/checkout/billing to its corresponding directory of your sitegenesis storefront cartridge.

Afterwards run gulp build.

# Maintenance

## Custom site preferences

There are separate preference groups: a more general one and one for each payment method.

## Jobs

Please refer to section 3.2.2.1 of this document

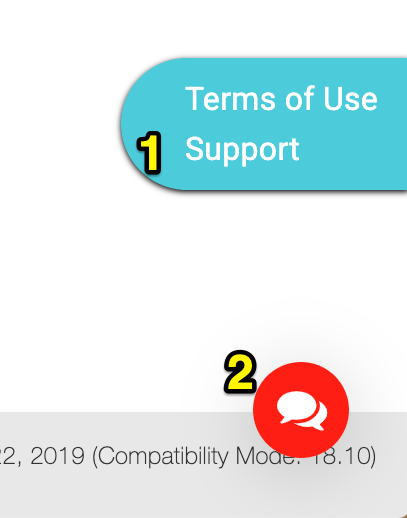
## Services

Please refer to section 3.2.2.2 of this document

# Support

There are several ways of contacting our support. You can do so by sending your request to the email address: [shop-systems-support@wirecard.com](mailto:shop-systems-support@wirecard.com)

From within the business manager you have to options:



1. Send a request to customer support

Navigate to PaymentGateway > Contact Support from within the site’s merchant tools. When sending the request, your current site preferences will be added to the request (no sensible data such as passwords will be transmitted!) – this is for helping our support trouble-shooting your request.

1. Use the livechat

Our livechat is available from any business manager page of our module. That is one of:

* Transaction overview
* Transaction detail view
* Test http user / password
* Contact support (form)
* Terms of Use

# Release History

|  |  |  |
| --- | --- | --- |
| **Version** | **Date** | **Changes** |
| 1.0.0 | Jun 7, 2019 | Initial version containing payment methods:   * Credit Card * PayPal * SEPA Direct Debit * Sofort. |
| 1.1.0 | Jun 16, 2019 | Payment methods:   * Giropay * iDEAL |