

1. The Resurs Bank E-Commerce Platform	4
1.1 Concepts and Domain	6
1.2 URLs for prod/live with important checklist	14
1.3 Resurs Checkout Web	16
1.3.1 API for retrieving masked card number	31
1.3.2 Checkout Integration, initial notes / tips / FAQ	32
1.3.3 Iframe communication (v2API)	33
1.3.3.1 ResursCheckoutJS Legacy	36
1.3.4 Pre-filling customer data in Resurs Checkout	41
1.3.5 updatePaymentReference notices	42
1.4 Resurs Checkout POS	43
1.4.1 Pre-filling customer data in Resurs Checkout POS	53
1.5 Resurs Checkout PUSH	54
1.5.1 Pre-filling customer data	62
1.6 Simplified Flow API	63
1.6.1 getAddress (SE)	71
1.6.2 getPaymentMethods	73
1.6.3 bookPayment	76
1.6.4 bookSignedPayment	79
1.6.5 getAnnuityFactors	80
1.6.6 getCostOfPurchaseHtml	82
1.6.7 Widget for information on installment amounts.	85
1.6.8 getAddressByPhone (NO)	86
1.7 Merchant API 2.0	88
1.7.1 Broker Application Flow Denmark	89
1.7.2 Merchant API E-Commerce	90
1.7.3 Merchant API POS	94
1.7.4 Payment Management	98
1.7.5 Physical Agreement Finland	100
1.8 Platform Plugins	101
1.8.1 Magento modules	105
1.8.1.1 Magento Plugin - OldFlow version	107
1.8.1.1.1 Magento Plugin 1.x - Changelog	111
1.8.1.1.2 OldFlow Plugin - Coding examples for getAddress	117
1.8.1.1.3 Plugin Hotfixes	120
1.8.1.1.4 Release- and installation notes, error logging and development for Magento OldFlow	122
1.8.1.2 Magento Plugin Releases (Download)	134
1.8.1.3 Resurs Bank Magento 2.4+ Installation Instruction	135
1.8.1.4 Resurs Bank Magento 2 CE 2.4+ payment gateway documentation	139
1.8.2 OpenCart	158
1.8.2.1 Changelog	162
1.8.3 PrestaShop Payment Gateways	163
1.8.3.1 Prestashop Resurs Checkout	164
1.8.3.2 PrestaShop SimplifiedShopFlow	168
1.8.3.2.1 Developer Notes	170
1.8.3.2.2 PrestaShop Simplified: Changes and updates	171
1.8.3.2.3 PrestaShop Simplified Described	172
1.8.3.2.4 Presta-Simplified: Frequently Discussed Issues and Important Notes	174
1.8.3.2.5 Presta-Simplified and translations	176
1.8.3.2.6 Status updates and callbacks	177
1.8.4 WooCommerce	178
1.8.4.1 0 decimals in WooCommerce	179
1.8.4.2 Figuring out remote ip for whitelisting	181
1.8.4.3 Resurs Merchant API 2.0 for WooCommerce	183
1.8.4.3.1 Plugin basics and information	184
1.8.4.3.2 Trouble shooting and error handling	186
1.8.4.3.3 Installation from WordPress plugin repository	187
1.8.4.3.4 Manually installing plugin	188
1.8.4.3.5 Store configuration requirements	190
1.8.4.3.6 Plugin configuration	191
1.8.4.3.7 Order management	194
1.8.4.3.8 Part payment widget	196
1.8.4.3.9 MAPI Checkout Flow	198
1.8.4.3.10 MAPI-WC - Customizations	199
1.8.4.4 Resurs Bank Payment Gateway for WooCommerce (v2.2) Resurs Checkout & Simplified Flow	200
1.8.4.4.1 Customized part payment information on product pages (WooCommerce)	203
1.8.4.4.2 Hooks/filters v2.2 (and core tweaks)	205
1.8.4.4.3 The problems with WPMU (Wordpress Network) and Resurs webservices	211
1.8.4.4.4 Using sku as article numbers in old plugin	212
1.9 After Shop Service API	213
1.9.1 Additional Debit of Payment	216
1.9.2 Annulling	218
1.9.3 Calculate Searchresult Size	221
1.9.4 Crediting / Refunding	223
1.9.5 Finalize Payment	227
1.9.6 Find Payments	232
1.9.7 Get Payment	234
1.9.8 Get Payment Document (PDF)	237
1.9.9 Get Payment Document Names	239

1.9.10 MetaData AfterShop	241
1.10 Payment administration GUI	243
1.10.1 Merchant Portal	244
1.10.1.1 Configuration Merchant Portal	245
1.10.1.2 Manipulate Payments in Merchant Portal	246
1.10.1.3 Search Payments in Merchant Portal	248
1.11 Configuration Service	249
1.11.1 Get registered callback	250
1.11.2 Peek Invoice Sequence	251
1.11.3 Register Event Callback	252
1.11.4 Set Invoice Data	253
1.11.5 Set Invoice Sequence	254
1.11.6 Unregister Event Callback	256
1.12 Developer Service	257
1.12.1 getLimitApplicationTemplate	258
1.12.2 isSigned	259
1.12.3 triggerEvent	260
1.13 Callbacks	261
1.13.1 ANNULMENT	268
1.13.2 AUTOMATIC_FRAUD_CONTROL	269
1.13.3 BOOKED	270
1.13.4 FINALIZATION	272
1.13.5 Parameters and Callbacks	273
1.13.6 TEST	275
1.13.7 UNFREEZE	277
1.13.8 UPDATE	278
1.14 Development	280
1.14.1 Errors, problem solving and corner cases	281
1.14.1.1 EComPHP custom errorcodes	282
1.14.1.2 Error handling (Resurs error codes)	284
1.14.2 General: Integration development	286
1.14.2.1 Building plugins (settings)	291
1.14.2.2 Handling checkout-flows bottle necks	293
1.14.3 PHP and development libraries	296
1.14.3.1 Version 1.x (EComPHP)	297
1.14.3.1.1 EComPHP: ChangeLog	300
1.14.3.1.2 EComPHP: Contribute	304
1.14.3.1.3 EComPHP: createPayment [hostedFlow]	305
1.14.3.1.4 EComPHP: createPayment [RCO]	310
1.14.3.1.5 EComPHP: createPayment [simplifiedShopFlow]	315
1.14.3.1.6 EComPHP: DEBIT, CREDIT, ANNUL [afterShopFlow]	320
1.14.3.1.7 EComPHP: getCostOfPurchaseHtml (Automated)	327
1.14.3.1.8 EComPHP: setAdditionalDebitOfPayment() - Update payments with additional data	328
1.14.3.1.9 EComPHP and localization	329
1.14.3.1.10 EComPHP features and tips	331
1.14.3.1.11 Important notes and troubleshooting/exceptions (EComPHP)	350
1.14.3.1.12 Testing (EComPHP)	356
1.14.3.2 Version 2.x (EComPHP)	358
1.14.4 API Types	362
1.14.4.1 address	363
1.14.4.2 basicPayment	364
1.14.4.3 bookingResult	365
1.14.4.4 bookPaymentResult	366
1.14.4.5 bookPaymentStatus	367
1.14.4.6 cardData	368
1.14.4.7 countryCode	369
1.14.4.8 customer	370
1.14.4.8.1 extendedCustomer	371
1.14.4.9 customerCard	372
1.14.4.10 customerIdentification	373
1.14.4.11 customerIdentificationResponse	374
1.14.4.12 customerType	375
1.14.4.13 digestAlgorithm	376
1.14.4.14 digestConfiguration	377
1.14.4.15 ECommerceError	378
1.14.4.16 formElement	379
1.14.4.17 fraudControlStatus	380
1.14.4.18 invoiceData	381
1.14.4.19 invoiceDeliveryType	382
1.14.4.20 limit	383
1.14.4.21 limitApplicationFormAsCompiledForm	384
1.14.4.22 limitApplicationFormAsObjectGraph	385
1.14.4.23 limitDecision	386
1.14.4.24 mapEntry	387
1.14.4.25 option	388
1.14.4.26 payment	389
1.14.4.27 paymentData	390
1.14.4.28 paymentDiff	391
1.14.4.29 paymentDiffType	392

1.14.4.30 paymentMethod	393
1.14.4.31 paymentMethodType	394
1.14.4.32 paymentSession	395
1.14.4.33 paymentSpec	396
1.14.4.34 paymentStatus	397
1.14.4.35 pdf	398
1.14.4.36 searchCriteria	399
1.14.4.37 signing object	400
1.14.4.38 Simple Types.....	401
1.14.4.39 sortAlternative	402
1.14.4.40 sortOrder	403
1.14.4.41 specLine	404
1.14.4.42 webLink	405
1.14.4.43 withMetaData	406
1.14.5 Create part payment widget	407
1.14.6 Customer data - Regular expressions	408
1.14.7 Rounding	410
1.14.8 Recognized metadata	411
1.14.9 Payment providers in simplified flow	412
1.14.10 Permissions and passwords - Platform access	413
1.15 Flow Chart Library	414
1.15.1 Annullment & Crediting Chart	420
1.15.2 Callback Overall Flow Chart	421
1.15.3 DocumentNames and Document Flow chart	422
1.15.4 Finalize Payment Chart	423
1.15.5 Fraud Screening Chart	424
1.15.6 Limit Application Form Chart	425
1.15.7 Signing Overall Flow Chart	426
1.15.8 Simplified Shop Flow Chart	427
1.16 Gift Card Service	428
1.17 Invoice Service 2.0	431
1.17.1 API Reference	433
1.17.2 Sample calls with cURL	442
1.18 Invoice Service 1.0 (deprecated)	447
1.19 Consumer Loan API	448
1.19.1 submitApplicationExt DK	450
1.19.2 submitApplicationExt FI	452
1.19.3 submitApplicationExt NO	466
1.19.4 submitApplicationExt SE	482
1.19.5 getApplicationQuote	494
1.19.6 acceptQuote	495
1.19.7 updateApplication	496
1.19.8 Callbacks and polling status	497
1.20 Reporting files example	502
1.21 Legal requirements	503
1.22 Testing	504
1.22.1 Test URLs	505
1.22.2 Verify integration	507
1.22.3 Test Data - Sweden	510
1.22.4 Test Data - Denmark	513
1.22.5 Test Data - Finland	515
1.22.6 Test Data - Norway	517
1.22.6.1 Testing Vipps	519
1.22.7 Customer Field Validation (regex)	521
1.23 FAQ	522
1.23.1 The "not a bug" list	527
1.24 Contact	528
1.25 Frequently asked questions	529
1.26 Merchant API 1	530
1.26.1 Credit Application	531
1.26.2 Credit Application and Payment	534
1.26.3 Payment with Existing Account	535
1.26.4 URLs for test & production	537
1.27 Logotypes new	538
1.28 Swish för Handel - Resurs Technical supplier	539

The Resurs Bank E-Commerce Platform



Resurs Checkout Web

Iframe based omni channel checkout for both retail stores and online



Resurs Checkout POS

Resurs Checkout POS is an iFrame browser-based payment gateway that can be used in a Point of Sale-system (POS).



Resurs Checkout PUSH

Load card, Cancel card, Get balance, Annul balance



Simplified Flow API

Integrate your checkout with
our webservice API



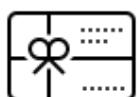
Plugins

Magento, WooCommerce,
Prestashop and OpenCart
plugins



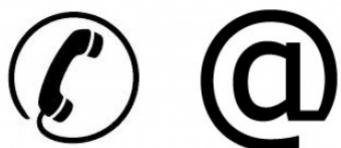
Payment administration GUI

After Shop Webservices



Gift Card Service

Load card, Cancel card, Get
balance, Annul balance



Concepts and Domain

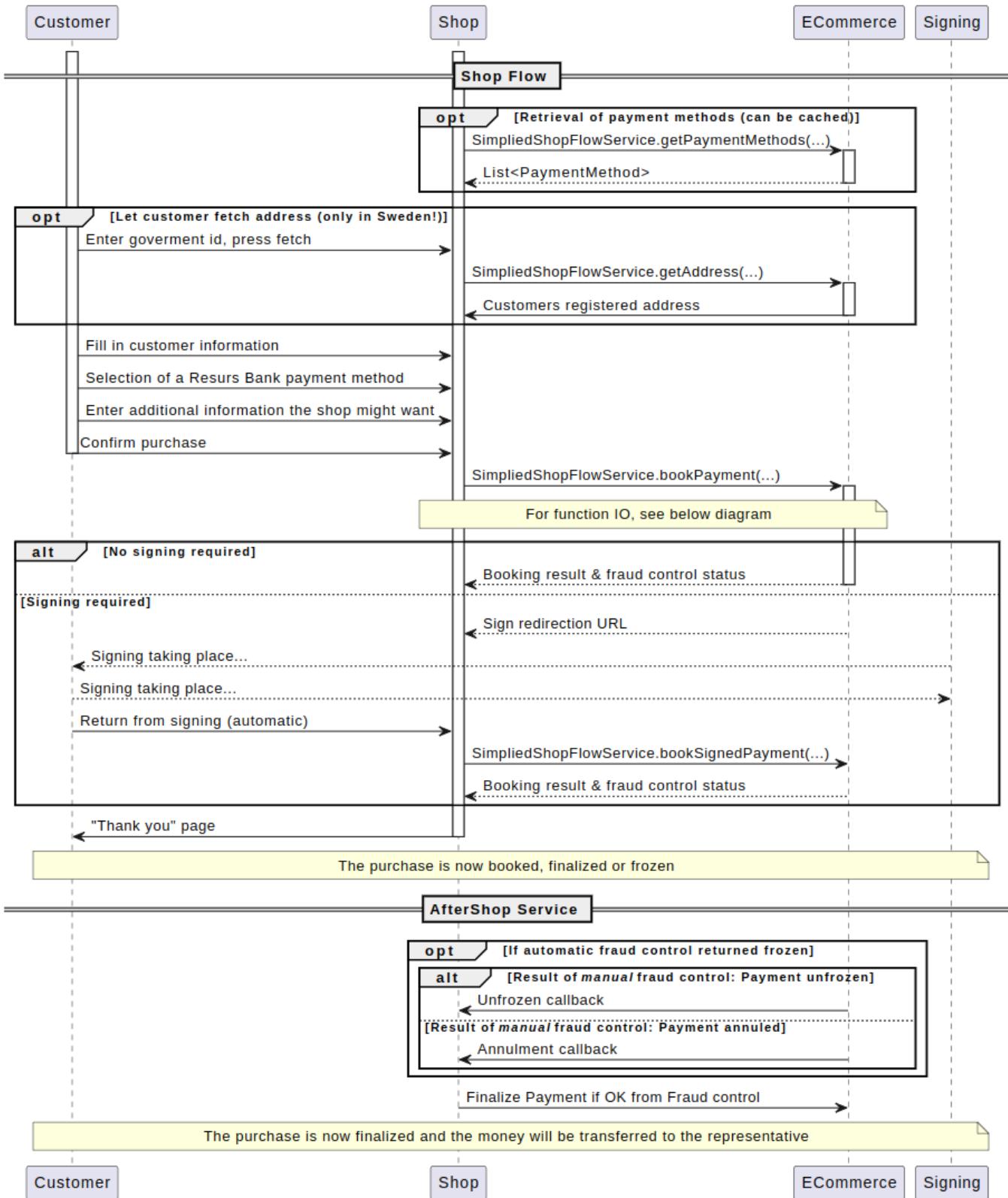
Like any other system, Resurs Bank eCommerce has a couple of concepts defining its domain. Understanding this domain reduces the risk of misunderstanding what functions do, and speeds up development. Thus we would recommend that you scroll through these pages before starting the final integration.

On this page:

- [High level chart of shop flow](#)
- [Annulment and crediting](#)
- [Associated metadata](#)
- [Credit cost information](#)
- [Fraud screening](#)
- [Invoices and credit notes](#)
- [Payment methods](#)
- [Payments](#)
- [Signing](#)

High level chart of shop flow

[Simplified Shop Flow](#)



Annulment and crediting

This document describes when to use annulment and when to use crediting. The table below shows a scenario that is probably far from an ordinary situation but is perfect for understanding the concepts.

Step	Action	Amount	Order Value	Available Purchase Amount	Annulable/Charged	Creditable
1	Creating limit	2.000 €	0 €	2.000 €	No	No
2	Book Payment	1.000 €	1.000 €	1.000 €	Yes - 1.000 €	No
3	Sell the entire order / Finalize Payment	1.000 €	1.000 €	1.000 €	No	Yes - 1.000 €
4	Crediting half order	500 €	500 €	1.500 €	No	Yes - 500 €
5	Additional purchases	300 €	800 €	1.200 €	Yes - 300 €	Yes - 500 €

An unspecified withdrawal can be made with the result that 300 € is removed from the payment by annulment and an unspecified withdrawal can be made with the result that 500 € is removed from the payment by crediting.

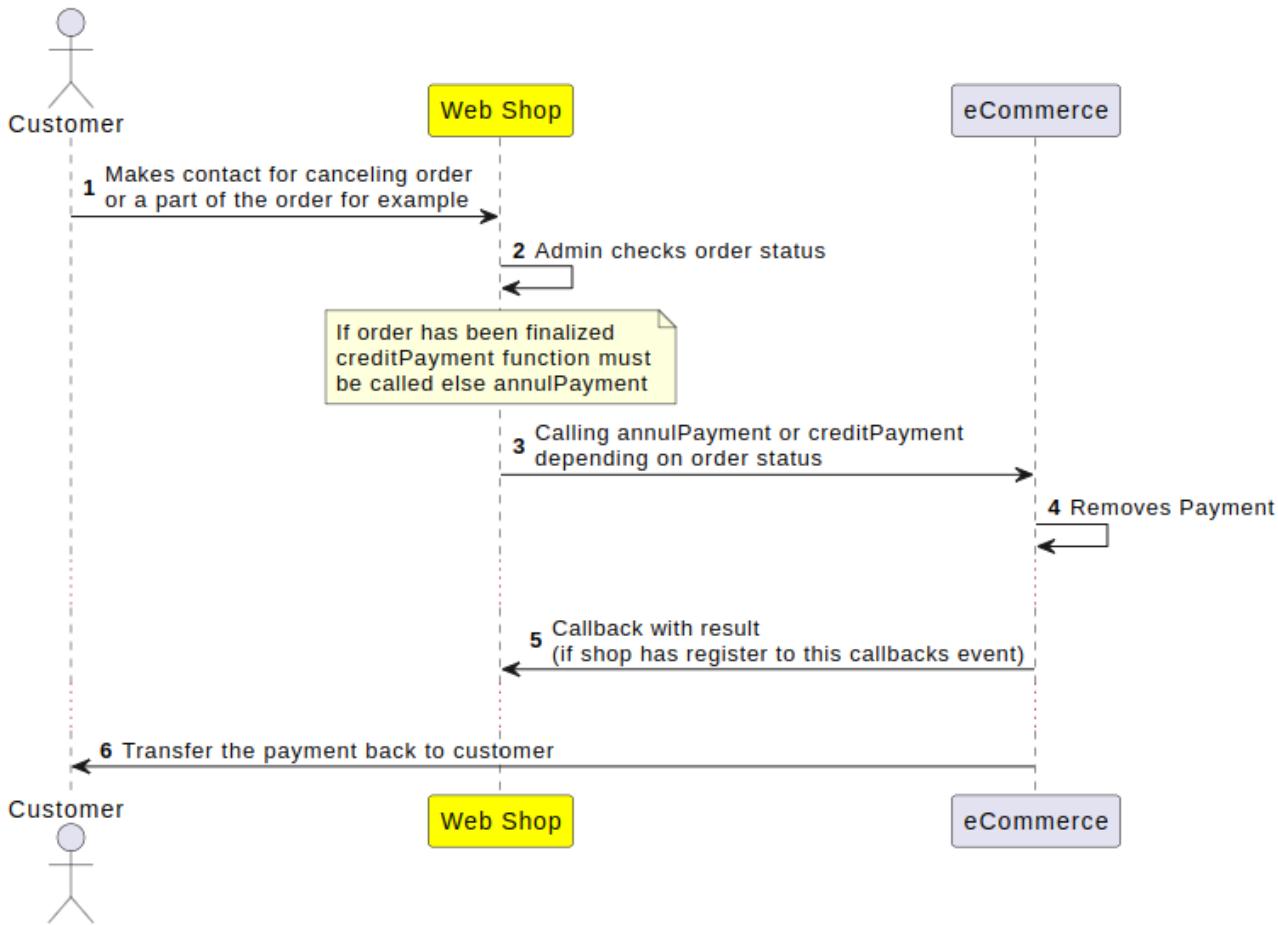
Annulment vs. Crediting

	Annulment	Crediting
Basic rule	Annulment can only be made on (part) payments that have not yet been finalized.	Crediting can only be made on (part) payments that has been finalized.
Money management	No money needs to be paid to the agent and no adjustment needs to be made.	Since the agent actually got paid from (part) payment when it was finalized, it must be regulated by money waiting to be paid to the agent, to be withdrawn. (Doesn't have to be the same for the customer)
Authorization	No transactions has been made at this point.	It recorded a credit transaction on the client/customer for the current agent.
Money, customer	No money has been shifted at this point, hence no impact.	Credit transaction means that the money is returned to the customer's account, or the customer's potential debt has been reduced by the same amount.
Money, agent	No money has been shifted at this point, hence no impact.	When the agent actually has got paid for a (part) payment when it was finalized, a credit means that a correction must be made by taking money waiting to be paid to the agent and pay to the customer. (Doesn't have to be the same for the customer.) If there are no such payments to the agent then the correction will wait until one comes, or the correction must be handled manually.

Completely or partially, annulment / credit.

whether if it's a annulment or credit, it can be made either completely or partially.

- AfterShopService.annulPayment(...) - full - or part annulment of a payment, with or without specification (only the amount is required)
- AfterShopService.creditPayment(...) - full - or part crediting of a payment, with or without specification (only the amount is required)



Associated metadata

What & why

The metadata is key/value data and its additional information in an order, determined by the e-retailer. It can be added to the order and can later be useful when [searching](#) for a payment. It can be anything, like information about the shipment. You can manage metadata through [PaymentAdmin](#) or using the [aftershop webservice](#)

Special metadata types

While we generally don't look at the metadata *there are* some keys we keep a lookout for: [Recognized metadata](#)

Credit cost information

! You are obliged to show credit information and link to Resurs Bank when showing prices based on credit, like a monthly cost.

Introduction

Credit cost information is information on cost elements (interest, fees, etc.) and total cost of our credits/loans. You have two options on how to show this information:

1. Download price information through our web service.
2. [Link to us](#) - we have a simple web-application showing the price information.

Download Credit cost information

This is done by a call to the [getCostOfPurchaseHtml](#) API method of the web service

This chunk of HTML should be shown in a modal window. As you can see it is styleable and we consider the element classes used to be part of our interface; we won't change them.

Link to us

With [getPaymentMethods](#) we return three links. The third of these goes to a simple web application which displays price information. Wanna know how it looks? [Here's a norwegian example](#).

Annuity

ECommerce can return a number of factors which when multiplied with a price represents a monthly cost for the product. See [getAnnuityFactors](#) method.

For example.

LED TV 4.995:- SEK, partial payments interest free for 24 months for 237 kr month.

You as the agent are forced to link to us for further price information when you show this kind of "prices".



Fraud screening

Fraud screening is about asserting the identity of the buyer, while the [limit applications](#) controls whether the buyer is eligible for the required credit. Fraud screening exists so neither party gets cheated and to increase security.

Fraud screening is performed when a purchase is made. If the automated routine finds suspicious patterns the payment will be **frozen** and Resurs Bank will assign a fraud officer to *ensure that the purchase is made by assumed buyer*.

When frozen, the payment cannot be [finalized](#). The payment will eventually be [unfrozen](#) or [annulled](#). Since manual handling is done in daytime the unfreeze event can happen severral days after the payment was frozen.



The status FROZEN is to be transparent to the customer. For all purposes, to the customer, the purchase has been made.

Purchase [finalization](#) and delivery are affected and stalled until the payment turns UNFROZEN. Thus, the merchant is responsible to monitor the status of the purchase, which can be done by either the [Payment Administration web GUI](#) or via registering and handling [Callbacks](#) .

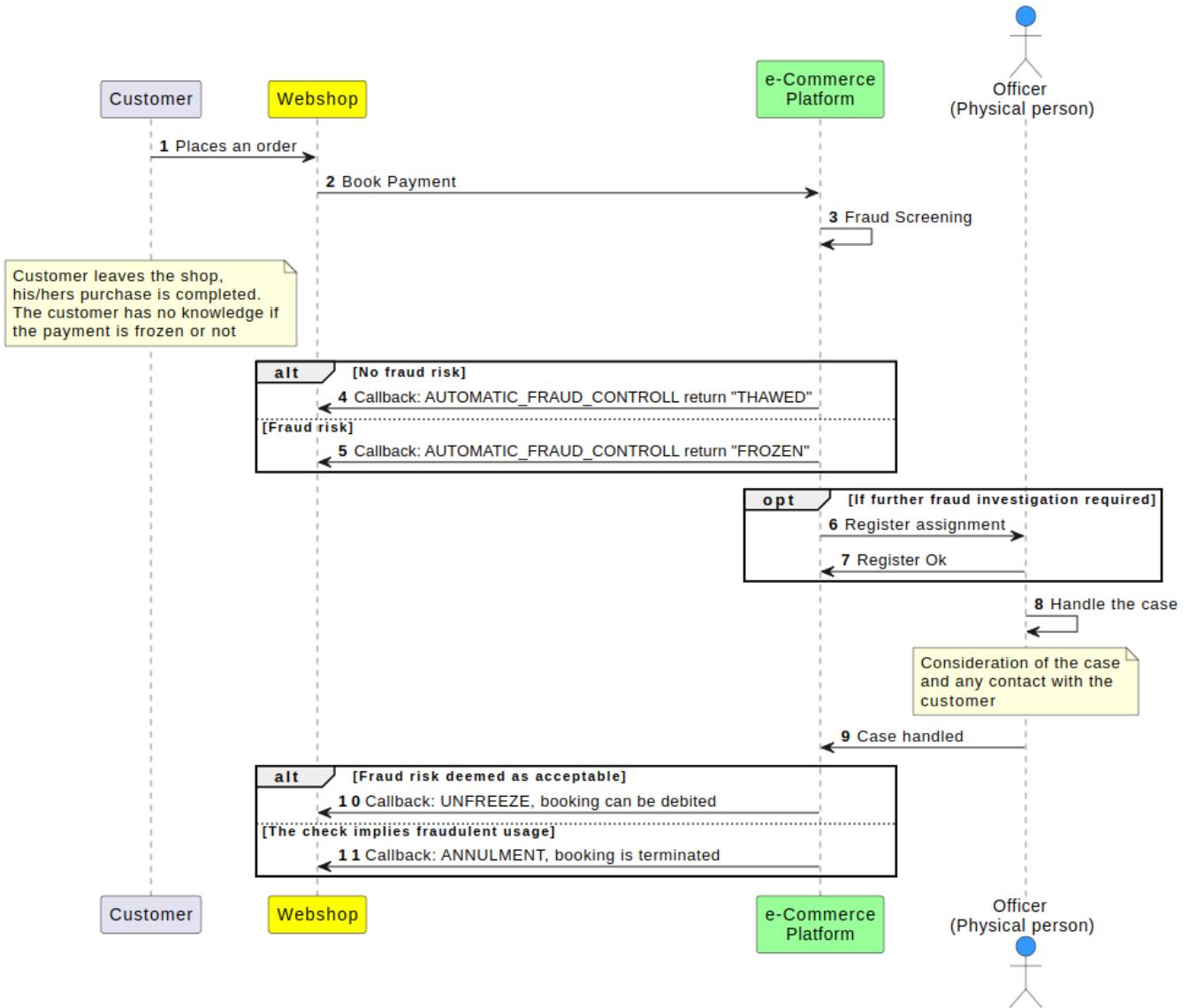
Status

The status of the payment fraud control.

Value	Description
FROZEN	The payment is currently frozen. This typically means that there is something that needs further investigation before the payment can be finalized.
BOOKED	The payment is not frozen, and may be finalized at any time.
CONTROL_IN_PROGRESS	<i>The deprecated flow:</i> the automatic fraud control is still in progress. It will result in FROZEN or BOOKED. In the simplified flow you will get an intermediate status of FROZEN when the control is still running.



Agent represent both agent and customer in this flow



Invoices and credit notes

Invoice settings

You can get and set next invoiceNumber in the After Shop service. Learn more about this [here...](#)

Get invoice

You can retrieve a invoice document in the After Shop Service. Learn more about this [here...](#)

Payment methods

A payment method is a way to finance a purchase. The payment methods available to choose from are:

- Invoice (*multiple*)
- Revolving credit (new card)
- Card payment
- Part payment

Limit

Every paymentmethod except card has a limit. The limit, the amount/payment span + standard increment, must be within for example 5,000 - 50,000 SEK. Note that the limit of the payment methods differs from each other. Read more about [limit applications](#)

Invoices

We support multiple types/settings of invoices: invoice for an individual customer or companies and we also provide invoices with part payment. The invoice will be sent by email to the customer. The invoice is generated when the payment has been finalized. If you prefer to handle the generation of the invoice yourself, you do this by not supplying an invoice number when finalizing the payment. However, the invoice must be pointing at us because we are creditors.

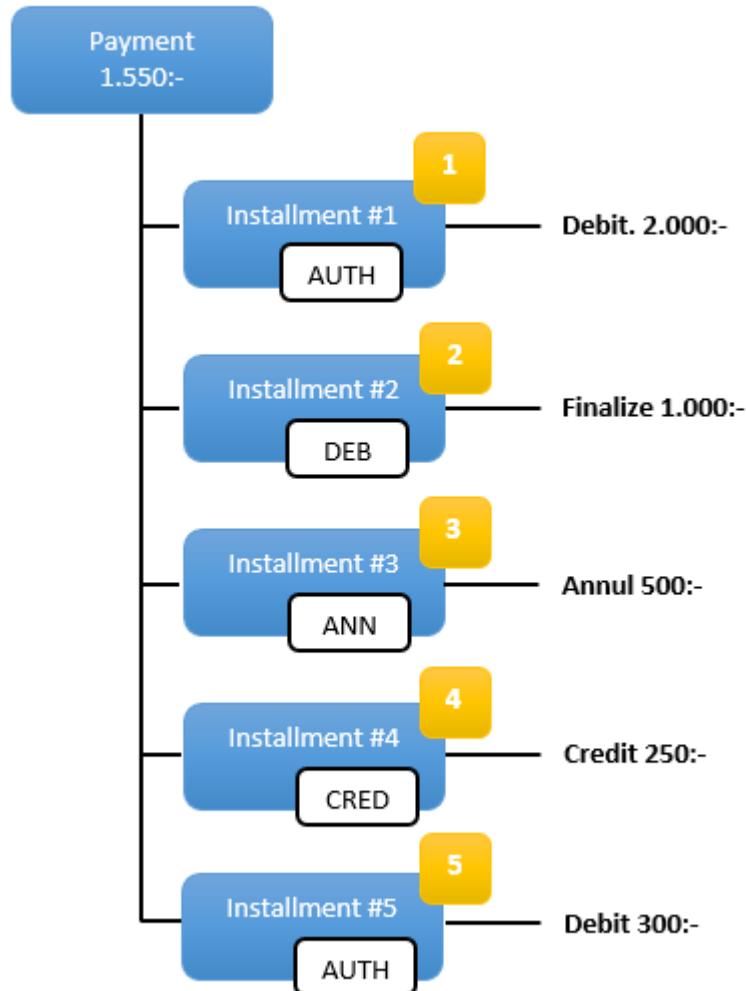
Payments

What is the difference between a payment and a payment-diff and how are they related? The simplest answer is that a **payment is a group of payment-diffs**.

Payment vs. Order

As defined above, payment sounds quite similar to ordergroups, and installment quite similar as orders, but there are some significant differences in how they are handled. Assume the following series of events:

1. Creating new order for 2,000 SEK
2. Sell half warrant (1000 SEK)
3. Annul half (500 SEK)
4. Credit of 25% of the sold portion (250 SEK)
5. Buy additional things for 300 SEK



The type of payment part.

Value	Description
AUTHORIZE	The payment part is an authorization request.
DEBIT	The payment part is a debit request.
CREDIT	The payment part is a credit request.
ANNUL	The payment part is an annulment request.

Types of payment diffs

Today, the system supports the following types of payment diffs:

Type	Description	Banking system	Order value	When is it used?	Comments
AUTHORIZE	Reservation of part / whole of the allotted limit.	Authorization of amounts in accounts.	Increases	When a purchase is booked.	In earlier drafts DEBIT has had dual functions, authorization and billing has been done by
DEBIT	Making of a debit.	Transaction of the amount on account. New authorization on the remaining amount.	Unchanged	When goods are shipped to the customer.	See above
CREDIT	Effecting a crediting.	Crediting the amount of the account.	Reduced	When a return is received.	
ANNUL	Annulment of the reservation amount.	New authorization on the remaining amount.	Reduced	When a booked purchase is canceled.	

Payment Status

Status Code	Description
DEBITABLE	Can be debited.
CREDITABLE	Can be credited.
IS_DEBITED	Is debited.
IS_CREDITED	Is credited.
IS_ANNULLED	Is annulled

Signing

Signing is a way for the buyer to identify him-/herself and therefor reduces risk for **fraud screening**. When signing is required varies depending on the purchase amount and other circumstances. Below you can see how the signing works.

URLs for prod/live with important checklist



Looking for test/staging environment?

You can find the test URLs here: [Test URLs](#)

- Firewalling
- Payment Admin
- Live URLs
 - Resurs Checkout
 - Hosted Flow
 - Webservices
 - WSDL files
 - Simplified Shop Flow:
 - After Shop Flow:
 - Configuration:
 - Developer:
 - Service endpoints
 - Simplified Shop Flow:
 - After Shop Flow:
 - Configuration:
 - Developer:
 - Checklist when changing to production

Firewalling



How to configure firewalls

Do you have a strictly configured environment? [Take a look here](#) to get proper settings for your firewall/web services.
Link: [FAQ#How do I configure my firewall/network](#)

Payment Admin

For the Payment Admin go to:
<https://paymentadmin.resurs.com/>

Live URLs

Resurs Checkout

For Resurs Checkout go to:
<https://checkout.resurs.com/checkout>

Hosted Flow

For hosted flow go to:
<https://ecommerce-hosted.resurs.com/back-channel>

Webservices

WSDL files

Simplified Shop Flow:

<https://ecommerce.resurs.com/ws/V4/SimplifiedShopFlowService?wsdl>

After Shop Flow:

<https://ecommerce.resurs.com/ws/V4/AfterShopFlowService?wsdl>

Configuration:

<https://ecommerce.resurs.com/ws/V4/ConfigurationService?wsdl>

Developer:

<https://ecommerce.resurs.com/ws/V4/DeveloperWebService?wsdl>

Service endpoints

Simplified Shop Flow:

<https://ecommerce.resurs.com/ws/V4/SimplifiedShopFlowService>

After Shop Flow:

<https://ecommerce.resurs.com/ws/V4/AfterShopFlowService>

Configuration:

<https://ecommerce.resurs.com/ws/V4/ConfigurationService>

Developer:

<https://ecommerce.resurs.com/ws/V4/DeveloperWebService>

Checklist when changing to production

- Request user name and password to services and to Resurs Payment Admin respectively in production.
- Change all URLs to the corresponding ones for production. If a plugin from Resurs is used both links will be there and you choose which one that shall be active.
- Check the price info and standard economic information document (SEKKI in Sweden) to see that agreed upon details are present.
- Amounts must be treated as in the test environment. You can't add a cost in production without including it as a order row with its cost.
- Do not round the amounts in production because it looks "neat" if the rounding is not validated in the test environment.
- Register callbacks with your production URL.
- Send in the IP of the end customer in the call, not yours as the agent.

Resurs Checkout Web

Overview

Resurs Checkout is an iFrame-based payment gateway that can be used from an online shop to process payments.



IPv6

Resurs Checkout has no IPv6 support.

Dina uppgifter

St**** El***** 198001010001 Ma***** 3, *** 55 G***** test@resurs.com, 0707123456	Inte du?
--	--------------------------

Annan leveransadress

Betalsätt

<input checked="" type="radio"/> Bankkortsbetalning	
Mer information ▾	
<input type="radio"/> Faktura Betala senare utan avgift	Resurs
<input type="radio"/> Delbetalning Betala i din takt	Resurs
<input type="radio"/> Nytt Resurs kort Betala allt eller dela upp	Resurs
<input type="radio"/> Resurs befintligt kort Köp nu, betala senare	Resurs
<input type="radio"/> Nytt revolverande konto Betala allt eller dela upp	Resurs
<input type="radio"/> Existerande revolverande konto Köp nu, betala senare	Resurs
<input type="radio"/> Kreditkortsbetalning	
<input type="radio"/> Direkt banköverföring	
<input type="radio"/> Swish	

Summa att betala

18 750 kr

Genomför köp

Genom att lämna information i kassan godkänner jag villkoren för
[Resurs Checkout](#) och användning av cookies.

Genom att klicka på "Genomför köp" accepterar jag
betalningsansvar och godkänner köpvillkoren hos [checkoutwebse](#)


[Läs mer här om hur vi hanterar dina personuppgifter](#)

Safe shopping by

Resurs

© 2022 Resurs Bank AB (publ)

Org.nr 516401-0208

Box 222 09, SE-250 24 Helsingborg

info@resursbank.se resursbank.se

What can I find here?

- Overview
 - Sequence diagram
 - Resurs Checkout Implementation
 - Prerequisites
 - Set up account
 - Authentication
 - End points (URLs)
 - Errors
 - API Overview
 - 1. Initiate payment
 - 2. Render Resurs Checkout snippet
 - 3. Update ongoing payment
 - 3. Manage Order
 - API - HTTP Request
 - POST /payments/{orderReference}
 - orderLines
 - metaData
 - successUrl
 - backUrl
 - paymentCreatedCallback
 Url
 - shopUrl
 - Response
 - Errors
 - PUT /payments/{orderReference}
 - orderLines
 - Response
 - GET /payments/{orderReference}
 - PUT /payments/{orderReference}
/updatePaymentReference
 - Interceptor notice
- Callbacks
 - POST /callbacks/{eventType}
 - GET /callbacks/{eventType}
 - GET /callbacks
 - DELETE /callbacks/{eventType}
- Order Administration
- Resurs Checkout Cookies
- Analytics and statistics usage
- Restrictions
- Terms and conditions of the shop

Resurs Checkout is responsive, it adapts automatically to desktops as well as mobile devices.

Resurs Checkout provides an easy to use checkout for all the payment methods that are configured in your PaymentAdmin/Merchant Portal interface such as invoice, partial payments and credit cards. The user can use a personal identity number (government ID) to retrieve address details which will populate the address fields automatically. Users who do not wish to enter their personal identity number may fill in their address details manually. Returning customers are identified by a browser cookie and will not need to re-enter the data into the checkout. This means that there are three ways in which a user's address details can be retrieved:

- Automatically by using a personal identity number (government ID)
- By the user manually entering the address details
- From a cookie stored in the user's browser

Supported payment methods (if enabled in the PaymentAdmin settings):

- Invoice
- Partial payment
- Credit cards
- Branded credit cards and application for a new credit card
- Swish

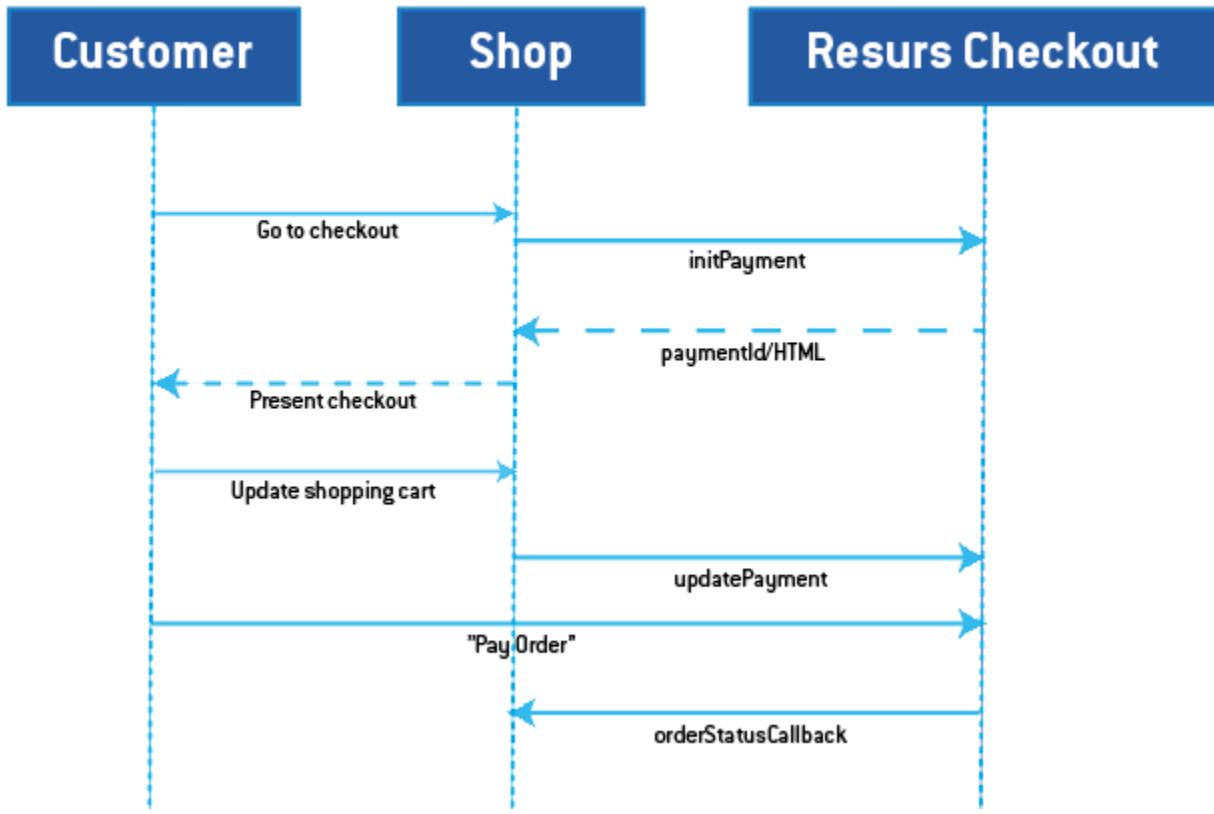
Note that all payment methods are to be discussed through the merchant and Resurs account manager. Onboarding can never setup a payment method without an order from the account manager.

Try Resurs Checkout for yourself

[Demo-shop](#)

Sequence diagram

The following image shows the interaction between the shop and the Resurs Checkout server.



- 1) The customer chooses to pay for items in the shopping cart in the online shop.
- 2) Your application server makes a POST to the REST-resource `/payments` and receives a response consisting of:
 1. The PaymentId valid for this session.
 2. An HTML-snippet that should be inserted in the checkout-webpage
- 3) The shop displays the checkout-page (where Resurs Checkout is now included) to the customer. The customer can now look up his/her address and select payment method.
- 4) While viewing the checkout the user may choose to add/remove articles from the shopping cart.
- 5) If the shopping cart is updated, your application must make a PUT to the REST-resource `/payments`. Resurs Checkout will then automatically be updated with the new total amount.
- 6) Customer selects to complete the payment.
- 7) Customer is :
 1. Paying with Visa/MasterCard
- 8) The customer is redirected to relevant pages to be able to finalize payment details and or performing signing with bank ID.
- 9) The customer browser is redirected to your "Thank you"-page or in case of failed payment, back to your checkout.
- 10) Your application server receives a paymentStatus callback based on the callback-url provided in step 2.

Resurs Checkout Implementation

Prerequisites

1. A merchant ID from Resurs Bank
2. An e-commerce system which can
 - a. make calls to the checkout service

- b. render a checkout page containing the Resurs Checkout iframe. See [Iframe communication \(v2API\)](#) documentation for more information about how to fully communicate and handle the iframe.
- c. display a payment confirmation page



HTTPS and certificate

Note that we only support HTTPS, both in test and production environment. You must have a valid and issued (not self signed) certificate for this.



TLS

Resurs Bank does not support Transport Layer Security (TLS) prior to Version 1.2

Set up account

To use Resurs Checkout you first have to set up an agreement with Resurs bank. This is when you get your merchant ID and your credentials for the webservice account.

Authentication

The Resurs Checkout API uses Basic Auth. To call the services you need to provide the credentials you received when setting up the agreement with Resurs Bank. [Read more...](#)

End points (URLs)

E-commerce:

- <https://checkout.resurs.com/checkout> - Production Environment
- <https://omnitest.resurs.com/checkout> - Test Environment

Test Data

For testing of Resurs Checkout use the below test personal identity numbers:

Sweden
Finland
Norway

Errors

Resurs APIs use HTTP 4xx 5xx status codes together with error messages to handle errors.

API Overview

Some of those API rest calls can be used without the need of Resurs Checkout. For example, if you need to get a full list of registered callbacks, you can make the request even if you don't use the Checkout itself.

URI	Method	Action
/payments/{orderReference}	POST	<p>Creates a payment session including:</p> <ul style="list-style-type: none"> • Shopping cart content (mandatory) • Customer information (Optional) <p>Output:</p> <ul style="list-style-type: none"> • iFrame/script HTML • Payment session id
/payments/{orderReference}	PUT	<p>Updates a payment session with:</p> <ul style="list-style-type: none"> • Shopping cart content (mandatory)
/callbacks/{eventType}	POST	Register a callback

/payments/{orderReference}	GET	Retrieves a payment. Used to fetch existing payments (not sessions).
----------------------------	------------	--

i **Want to run DELETE on payment session?**

Running a POST with the same orderReference a second time will overwrite the first payment session, making the previous one obsolete.

1. Initiate payment

First you have to initiate the payment. For details about this see [POST /checkout/payments/](#).

When you initiate the payment you may choose to [pre-filling data about the customer](#). We recommended this feature to optimize the Checkout for Returning Customers.

The JSON response from the *POST payments* call contains HTML snippet that allows the merchant to render the Checkout iframe. The JSON response payload also contains a unique *paymentSessionId* generated by Resurs Bank.

2. Render Resurs Checkout snippet

From the *POST payments* call, get the *html* property and embed it into your page where you would like the Resurs Checkout to be rendered. We recommend you to embed the HTML snippet server side.

3. Update ongoing payment

Whenever the content of your shopping cart is updated you need to notify Resurs Checkout. For details about this see [PUT /checkout/payments/](#).

3. Manage Order

When the payment has been booked at Resurs - you can now choose to manage the order by implementing APIs in [After Shop Service API](#) or manually in Resurs GUI [Manipulate Payments in Merchant Portal](#).

Note SWISH payments are automatically finalized. Therefore the only action available when payment is done is crediting (creditPayment in AfterShopService).

You can use debiting (finalizePayment) and annulling (annullPayment) as well as addition (additionalDebitOfPayment). Note that crediting (creditPayment) only can be done when the payment has been finalized.

To see what actions you can take on a payment depending on its status, [Read more here...](#)
To further understand what the Resurs-status means, [Read more here...](#)



If you are to use Nets, a requirement from Nets is max 32 characters for {orderReference}.
If you are to use Swish, a requirement from Swish is max 35 characters for {orderReference}.

Allowed characters are a-z A-Z 0-9 -

API - HTTP Request

Some of those API rest calls can be used without the need of Resurs Checkout. For example, if you need to get a full list of registered callbacks, you can make the request even if you don't use the Checkout itself.

POST /payments/{orderReference}

This service sets up the payment session and should be called whenever you want to display the checkout. It will return the paymentId and html to be included in the online shop.

Request parameters

```
{
  "orderLines": [
    {
      "artNo" : "1",
      "quantity": 1,
      "unitPrice": 100
    }
  ]
}
```

```

        "description" : "Order line:1",
        "quantity" : 1.0,
        "unitMeasure" : "pcs",
        "unitAmountWithoutVat" : 500.0,
        "vatPct" : 25.0,
            "totalAmountWithVat": 625,
            "totalVatAmount": 125
    },
    {
        "artNo" : "2",
        "description" : "Order line:2",
        "quantity" : 1.0,
        "unitMeasure" : "pcs",
        "unitAmountWithoutVat" : 10.0,
        "vatPct" : 25.0,
            "totalAmountWithVat": 12.5,
            "totalVatAmount": 1.25,
            "type": "ORDER_LINE"
    },
    {
        "artNo" : "80",
        "description" : "Discount",
        "quantity" : 1.0,
        "unitMeasure" : "pcs",
        "unitAmountWithoutVat" : -5.0,
        "vatPct" : 0.0,
            "totalAmountWithVat": -5.0,
            "totalVatAmount": 0.0,
            "type": "DISCOUNT"
    },
    {
        "artNo" : "90",
        "description" : "Shipping & Handling",
        "quantity" : 1.0,
        "unitMeasure" : "pcs",
        "unitAmountWithoutVat" : 5.0,
        "vatPct" : 0.0,
            "totalAmountWithVat": 5.0,
            "totalVatAmount": 0.0,
            "type": "SHIPPING_FEE"
    }
],
"metaData" : [
    {
        "key": "key1",
        "value": "value1"
    },
    {
        "key": "key2",
        "value": "value2"
    },
    {
        "key": "CustomerId",
        "value": "String 20 chars"
    },
    {
        "key": "invoiceExtRef",
        "value": "String 46 chars"
    }
],
"customer": {
    "governmentId": "8305147715",
    "mobile": "0707123456",
    "email": "test@resurs.se",
    "deliveryAddress": {
        "firstName": "Daniel",
        "lastName": "Johnsson",
        "addressRow1": "Ekslingan 8",
        "addressRow2": null,
        "postalArea": "Helsingborg",
        "postalCode": "25467",
        "city": "Helsingborg"
    }
}
]

```

```

        "countryCode": "SE"
    }
},
"successUrl": "https://shop.representative.com/order/12345/success/",
"backUrl": "https://shop.representative.com/order/12345/checkout/",
"shopUrl": "https://mystore.test.com"
}

```

Attribute	Data type	Mandatory	Description
orderLines		X	Object that represents an orderline.
orderLines/artNo	varchar (100)	X	Article number.
orderLines /description	text	X	Article name. Will be shown on the invoice. (Nillable? yes)
orderLines /quantity	decimal (15,5)	X	Quantity for the specific article.
orderLines /unitMeasure	varchar (15)	X	Unit type for the orderline e.g. kg, pcs.
orderLines /unitAmountWith outVat	decimal (15,5)	X	Net price per unit.
orderLines /vatPct	decimal (15,5)	X	Vat per unit (%).
orderLines /totalAmountWit hVat	decimal (15,5)		The total amount of the orderline, including VAT (do not forget to multiply by quantity)
orderLines /totalVatAmount	decimal (15,5)		The total VAT-amount of the orderline
metaData			Extra meta data for the payment. Recognized metadata
metaData/key	varchar (30)		Meta data key
metaData/value	text		Meta data value
customer			Used for pre-filling customer data in the checkout. It is possible to only send partial customer data. If customer has a cookie from Resurs, this will override all customer data that is sent in below
customer /governmentId	varchar (15)		GovernmentID of the customer - will automatically fetch country registered address of the customer (not applicable for Norway)
customer/mobile	varchar (45)		Mobilephone number.
customer/email	varchar (300)		Email address.
customer /deliveryAddress			If no delivery address is sent and governmentId is sent, delivery address will automatically be the customers country registered address (not applicable for Norway).
customer /deliveryAddress /firstName	varchar (100)		Given name
customer /deliveryAddress /lastName	varchar (100)		Family name
customer /deliveryAddress /addressRow1	varchar (250)		Street address, first line

customer /deliveryAddress /addressRow2	varchar (250)		Street address, second line
customer /deliveryAddress /postalArea	varchar (100)		City
customer /deliveryAddress /postalCode	varchar (45)		Postal/post code
customer /deliveryAddress /countryCode	varchar (2)		Country code of the delivery address - SE/NO/FI/DK. Has to be same country as the webservice-account is created for. If you're unsure, please contact us.
successUrl	text	X	<p>The address of the web page where the customer will be redirected when the payment has completed successfully.</p> <p> The URL must contain a unique identifier so that the URL can be linked to the correct order at the merchant.</p> <p>Use a SALT when creating the identifier for greater security.</p>
backUrl	text	X	<p>The address of the web page where the customer will be redirected if the payment is not completed successfully or if the customer navigates back him-/herself. A message will be shown and it will be the same regardless of error or back navigation, so a generic message is recommended.</p>
paymentCreatedCallbackUrl	text	-	<p>A callback URL that will be called when a payment is created. If a user closes the webrowser before he/she has been redirected to the success Url.</p> <p>There is also a possibility to register a permanent callback BOOKED. It's more safe to use the BOOKED callback since you can use a SALT in that call. See Available callback-types</p> <p> If both paymentCreatedCallbackUrl and BOOKED are used, paymentCreatedCallbackUrl is prioritized.</p> <p>If you need full support of callbacks we recommend you to not using this callback setup since it will stop other important callbacks to run.</p>
storeId	varchar	-	Used with permission from Resurs Bank, if you have a chain of stores. storeID defines which store in the chain it is.
shopUrl	text	X	<p>HOST ONLY based URL, regardless of the real http/uri-path without trailing slashes!</p> <p>Example: Your real store URL is https://my-store.com/with/this/long/uri/path/entry</p> <p>Your shopUrl will be https://my-store.com</p> <p>PURPOSE</p> <p>Used to send and receive customer based events between the shop and Resurs Bank. The events are closely described at the OmniCheckoutJS-Communication-documentation. Even if you are not explicitly using this communication it could be recommended to add it to your payload anyway.</p>

Response

Response body

```
{
    "paymentSessionId": "payment-session-id-as-uuid",
    "iframe": "<iframe id=\"rco-checkout-app-frame\" src=\"https://omnitest.resurs.com/web/dist/omni-checkout.html?56e31e24-6e73-4dfc-8ab2-9413fbe8e173\" frameborder=\"0\" width=\"100%\" scrolling=\"no\"></iframe>",
    "script": "<script type=\"text/javascript\" src=\"https://omnitest.resurs.com/web/dist/js/oc-shop.js\"></script>",
    "customer": {
        "governmentId": 8305147715,
        "mobile": "0701234567",
        "email": "test@example.com",
        "invoiceAddress": {
            "fullName": "Firstname Lastname",
            "firstName": "Firstname",
            "lastName": "Lastname",
            "addressRow1": "Addressrow1",
            "addressRow2": null,
            "postalArea": "postalArea",
            "postalCode": "12345",
            "countryCode": "SE"
        },
        "deliveryAddress": null,
        "customerType": "NATURAL",
        "mobileNotValidated": "0701234567",
        "emailNotValidated": "test@example.com"
    },
    "baseUrl": "https://omnitest.resurs.com/web/dist/omni-checkout.html",
    "html": "<iframe id=\"rco-checkout-app-frame\" src=\"https://omnitest.resurs.com/web/dist/omni-checkout.html?56e31e24-6e73-4dfc-8ab2-9413fbe8e173\" frameborder=\"0\" width=\"100%\" scrolling=\"no\"></iframe><script type=\"text/javascript\" src=\"https://omnitest.resurs.com/web/dist/js/oc-shop.js\"></script>"
}
```

Attribute	Description	
paymentSessionId	The ID for the payment session.	
html	The html snippet that must be displayed by the shop in order to display Resurs Checkout.	
iframe	The html snippet to display the iframe only.	
script	The script snippet that should be put in the header (for facelift, not the legacy version).	
customer	The customer object	
baseUrl	The baseurl for RCO.	

Errors

HTTP status code	Type	Example
400	Application/json	{ "message": "OrderReference already exists" }
401	Application/json	{ "message": "Bad credentials" }
500	Application/json	{ "message": "Internal error" }

PUT /payments/{orderReference}

This service should be called whenever the content of the shopping cart is updated. The iFrame's total sum, visible to the customer, is then updated via websocket.

Request parameters

```
{
    "orderLines": [
        {
            "artNo" : "1",
            "description" : "Order line:1",
            "quantity" : 1.0,
            "unitMeasure" : "pcs",
            "unitAmountWithoutVat" : 500.0,
            "vatPct" : 25.0
        },
        {
            "artNo" : "2",
            "description" : "Order line:2",
            "quantity" : 1.0,
            "unitMeasure" : "pcs",
            "unitAmountWithoutVat" : 10.0,
            "vatPct" : 25.0
        },
        {
            "artNo" : "3",
            "description" : "Order line:3",
            "quantity" : 1.0,
            "unitMeasure" : "pcs",
            "unitAmountWithoutVat" : 20.0,
            "vatPct" : 25.0,
            "type": "ORDER_LINE"
        },
        {
            "artNo" : "80",
            "description" : "Discount",
            "quantity" : 1.0,
            "unitMeasure" : "pcs",
            "unitAmountWithoutVat" : -5.0,
            "vatPct" : 0.0,
            "type": "DISCOUNT"
        },
        {
            "artNo" : "90",
            "description" : "Shipping & Handling",
            "quantity" : 1.0,
            "unitMeasure" : "pcs",
            "unitAmountWithoutVat" : 5.0,
            "vatPct" : 0.0,
            "type": "SHIPPING_FEE"
        }
    ]
}
```

Attribute	Mandatory	Description
orderLines	X	Object that represents an orderline.
orderLines/artNo	X	Article number.
orderLines/description		Article name.
orderLines/quantity	X	Quantity for the specific article.
orderLines/unitMeasure	X	Unit type for the orderline e.g. kg, pcs.
orderLines/unitAmountWithoutVat	X	Net price per unit.
orderLines/vatPct	X	Vat per unit.

orderLines/type	-	The type of the order line. <i>Optional</i> . It can be one of the following: ORDER_LINE - the same as if nothing is sent in, it will result in a normal order line. DISCOUNT - will be presented as a discount in Resurs Checkout SHIPPING_FEE - will be presented as a shipping fee in Resurs Checkout.
-----------------	---	--

Response

HTTP status code	Type	Example
200	Application/json	OK
400	Application/json	{ "message": "Invalid paymentId." }
401	Application/json	{ "message": "Bad credentials" }
404	Application/json	{ "message": "The order does not exist" }
500	Application/json	{ "message": "Internal error." }

GET /payments/{orderReference}

Response

```
{
    "id": "Omni-Session_2342345",
    "totalAmount": 500,
    "metaData": [
        {
            "key": "advertisement",
            "value": "false"
        }
    ],
    "limit": 500,
    "paymentDiffs": [
        {
            "type": "AUTHORIZE",
            "transactionId": null,
            "created": "2020-04-09T08:41:09.000+0000",
            "createdBy": "SHOP_FLOW",
            "paymentSpec": {
                "specLines": [
                    {
                        "id": "0",
                        "artNo": "ART123",
                        "description": "Product description",
                        "quantity": 2,
                        "unitMeasure": "kg",
                        "unitAmountWithoutVat": 200,
                        "vatPct": 25,
                        "totalVatAmount": 100,
                        "totalAmount": 500
                    }
                ],
                "totalAmount": 500,
                "totalVatAmount": 100,
                "bonusPoints": 0
            },
            "orderId": null,
            "invoiceId": null,
            "documentNames": []
        }
    ],
    "customer": {
        "governmentId": "8305147715",
        "address": {
            "fullName": "William Williamsson Eliassson",
            "firstName": "William",
            "lastName": "Eliassson",
            "street": "Södergatan 123",
            "zipCode": "111 11 Stockholm",
            "city": "Stockholm",
            "country": "Sweden"
        }
    }
}
```

```

        "addressRow1": "Ekslingan 13",
        "addressRow2": null,
        "postalArea": "Helsingborg",
        "postalCode": "25024",
        "country": "SE"
    },
    "phone": "+46701234567",
    "email": "info@resurs.se",
    "type": "NATURAL"
},
"deliveryAddress": {
    "fullName": "Testsson Test",
    "firstName": "Test",
    "lastName": "Testsson",
    "addressRow1": "Ekslingan 9",
    "addressRow2": null,
    "postalArea": "Helsingborg",
    "postalCode": "254 67",
    "country": "SE"
},
"booked": "2020-04-09T08:41:17.000+0000",
"finalized": null,
"paymentMethodId": "VISA SUPER CARD",
"paymentMethodName": "Mocked PSP",
"fraud": false,
"frozen": false,
"status": [
    "DEBITABLE"
],
"storeId": "107999",
"paymentMethodType": "PAYMENT_PROVIDER",
"totalBonusPoints": 0
}
}

```

HTTP status code	Type	Example
200	Application/json	OK
401	Application/json	{ "message": "Bad credentials" }
404	Application/json	{ "message": "The order does not exist"}
500	Application/json	{ "message": "Internal error." }

PUT /payments/{orderReference}/updatePaymentReference

Changes the payment reference on an **existing payment reference**, the old reference will be replaced. Observe that the payment must not be booked in Resurs before this event.

Interceptor notice



Interceptor

If you are using the javascript button-interceptor, you must wait for acknowledgement from this PUT event, before sending back the confirmation to the iframe. Also, make sure you synchronize your cart with the order at Resurs Bank side, to secure that no content has been changed in the cart while running through the interceptor.

```

PUT payments/{orderReference}/updatePaymentReference
{
    "paymentReference" : "newPaymentReference"
}

```

Callbacks

Read more about callbacks [here](#).

POST /callbacks/{eventType}



When registering a callback with JSON, note that "/checkout" is to be removed from the end-point.
Additionally eventType is case sensitive.
Example: <https://omnitest.resurs.com/callbacks/{BOOKED}>



If you use the parameter digest you must provide paymentId as digest parameter like in the example below. Otherwise you will get a successful registration but the callback won't be triggered and sent.

Input

```
{  
    "uriTemplate": "http://host.com/context/function.html?yourIDname={paymentId}&hash={digest}" ,  
    "basicAuthUserName": "user" ,  
    "basicAuthPassword": "password" ,  
    "digestConfiguration": {  
        "digestAlgorithm": "MD5" ,  
        "digestSalt": "SALT" ,  
        "digestParameters": [  
            "paymentId"  
        ]  
    }  
}
```

Attribute	Mandatory	Description
uriTemplate	X	
basicAuthUserName	-	Username
basicAuthPassword	-	Password
digestConfiguration	-	
digestAlgorithm	-	MD5, SHA1
digestSalt	-	
digestParameters	-	

GET /callbacks/{eventType}

Get a specific registered callback.

Output

```
{  
    "eventType": "UNFREEZE" ,  
    "uriTemplate": "http://host.com/context/function.html"  
}
```

GET /callbacks

Get all registered callbacks

Output

```
[  
  {  
    "eventType": "UNFREEZE",  
    "uriTemplate": "http://host.com/context/function.html"  
  },  
  {  
    "eventType": "BOOKED",  
    "uriTemplate": "http://test.resurs.com"  
  },  
  {  
    "eventType": "TEST",  
    "uriTemplate": "http://test.resurs.com"  
  }  
]
```

DELETE /callbacks/{eventType}

Delete a specific registered callback.

Order Administration

For details on how to view and manage your payment see the following documentation.

[Merchant Portal](#)

Resurs Checkout Cookies

Resurs Checkout uses cookies to store customer name and address in order to simplify the shopping process for returning customers. The user always has the option to not use cookies in which case customer details will have to be entered every time a payment is made (unless the customer is logged in to the online shop and customer information is included in the call to initiate the payment).

More details regarding our cookie policy:

<https://test.resurs.com/omnicheckoutweb/cookie-policy.html>

Analytics and statistics usage

Resurs Bank collects information about users and user behavior. This is done through e.g. Google analytics, cookies or similar services.

Restrictions

Payment is possible for persons, but not for companies

In integration with Nets, a requirement from Nets is max 32 characters for {orderReference}

In integration with Swish, a requirement from Swish is max 36 characters for {orderReference}

The orderReference-sequence is per webservice account, not per storeId. As a solution, if you are running storeId, you can add the store number as a prefix to the sequence

Terms and conditions of the shop

The merchant will have to provide a link to the terms and conditions, e.g. by email, to Resurs Bank that will put it into the Resurs Checkout iFrame.

API for retrieving masked card number



Note!

The payment has to be made thru our PSP-partner NETS and the merchant has been registered in our interface Merchant Portal. If you are uncertain about either of these prerequisites, please contact us to verify.

End-point

https://omnitest.resurs.com/checkout/payments/{orderReference}/card_details

Simply enter your requested order reference into the endpoint and run the request (GET-function)

If there is a card payment for your searched order reference, you will receive a 200-response containing the information below

Output example - Success

```
{  
    "maskedCardNumber": "492500*****0004"  
}
```

If there isn't a card payment for your searched order reference, you will receive a 400-response containing the information below

Output example - fail

```
{  
    "detailMessage": "Could not find payment for externalId: {orderReference} Do you find this error strange? E-mail  
information about the payment to ehandel@resurs.se  
or follow the contact information page here: https://test.resurs.com/docs/x/9gAF",  
    "faultInfo": {  
        "errorTypeDescription": "NOT_ALLOWED",  
        "errorTypeId": 4,  
        "fixableByYou": true,  
        "userErrorMessage": "Tekniskt fel"  
    }  
}
```

Checkout Integration, initial notes / tips / FAQ

Resurs Checkout is a full hosted iframe based checkout solution, with the purpose of getting started faster with a checkout, that you don't have to build by yourself from scratch. This is a small guide for how implementation could be done.

The development could be made in a few different ways, depending on your needs.

Development type	Described as	Requirements or recommended
GLOBAL REQUIREMENTS	Required for all solutions below	<ul style="list-style-type: none">Billing fields and shipping fields are handled by Resurs Bank, not the storeWhen the order are created, enforce address data to be the one that is returned from Resurs BankMake sure that, if the cart is editable from the checkout page, the iframe could be updated with the new content (websockets are required) Note: To make this work properly, you should let the site handle this Frontend (Ajax) Backend (Shop) PUT Resurs Checkout APIMake sure that you implement the standard Callbacks, either by this solution or via the Resurs Checkout json-object
SIMPLE	The store handles the orders internally with minor work, by for example the user session.	<ul style="list-style-type: none">A landing/order confirmation page where the order are created by the store
ADVANCED	The store handles the orders internally, but customer data are handled live	<ul style="list-style-type: none">A landing/order confirmation page where the order are updated further and possibly finishedOrders are created backend via frontend, before creating the order at Resurs Bank (See OmniJS)Make sure that your store handles order confirmation mail properly, or this method may send duplicate order confirmations (also important if Resurs Bank denies payments)

For PHP we use our in-house developed library [PHP and development libraries](#), to utilize the functions of making calls to both Resurs Checkout and our other shopflows. EComPHP are able to handle the creation of the Resurs Checkout iFrame, including updating orders when OmniJS (or similar) is active. EComPHP uses an [open source solution \(Develop-Snapshot\)](#) when it comes to handling http calls, therefore an activated cURL-module in PHP is required.

Iframe communication (v2API)

Table of contents

- Requirements
- What happened to ResursCheckoutJS?
- Integration
 - Making the POST-request in your backend scripts
 - Quick start: Most used methods in the \$ResursCheckout object
 - create()
 - onSubmit(), release(), recreate()
 - onCustomerChange(), onPaymentChange()
 - onPaymentFail()
- Undocumented features

Requirements

None.

What happened to ResursCheckoutJS?

ResursCheckoutJS - ***the older release based on postMsg*** - is no longer fully supported. The old documentation for that system has been replaced with this one.

If you already run on the old postMsg-platform, we recommend you to upgrade your platform to this current version (2) which is hosted in the form of a framework at Resurs Bank side.

Integration

When requesting an iframe for the old legacy release, all of the content code had to be put in the body instantly. Some platforms did not support this method and in some cases, you had to put some executable scripts in the <head> tags. For version 2, both variants is available but using the header method is the most recommended way to do this on. The easy described way to implement this solution is

Making the POST-request in your backend scripts

When generating the v2-iframe, you get a bunch of variables in response as before, but they are separated like this:

```
{  
  "paymentSessionId": "session-id-uuid",  
  "iframe": "<iframe id=\"rco-checkout-app-frame\" src=\"https://omnitest.resurs.com/v2/se/?session-id-  
  uuid\" frameBorder=\"0\" width=\"100%\" scrolling=\"no\" ></iframe>",  
  "script": "<script type=\"text/javascript\" src=\"https://omnitest.resurs.com/v2/se/resursCheckout.  
  js\"></script>",  
  "baseUrl": "https://omnitest.resurs.com/v2/se/",  
  "html": "<iframe id=\"rco-checkout-app-frame\" src=\"https://omnitest.resurs.com/v2/se/?session-id-  
  uuid\" frameBorder=\"0\" width=\"100%\" scrolling=\"no\" ></iframe><script type=\"text/javascript\" src=\"  
  https://omnitest.resurs.com/v2/se/resursCheckout.js\"></script>"  
}
```

The best way to use this payload is to put the iframe in the body as before, but put the script in the header of your site. In the above example, you'll need the iframe and the script-data. In your frontend, make sure you wait until the pageload is completed, for example:

```
jQuery(document).ready(function ($) {  
  // Put your actions here.  
});
```

Same example but without jquery dependencies:

```
window.onload = function() {  
  // Put your actions here.  
}
```

At the "start actions" point in the codeblock above, you can now initialize all event required from Resurs Bank. By waiting for document readiness, you'll also ensure the site that the object \$ResursCheckout exists, since this is the one you communicate with to-and-from the iframe. There's a bigger live example on how an [integration could be made at github](#).

Quick start: Most used methods in the \$ResursCheckout object

The most common use with \$ResursCheckout is the following events:

create()

This is the iframe initializer. We'd suggest that you put this one as the last action in the chain, as this initializes and creates the iframe. And before that we'd like to ensure that our other events is ready to run. This one takes four arguments:

paymentSessionId, baseUrl, hold and containerId. As you can see, we call for a method in the example below, with the name **getResursRcoContainer** in two of the arguments. Those are very important and can be fetched from the above payload. They have to be injected in the create command somehow and in the example, we have this data stored elsewhere in the javascript so it can easily imported to this place.

We also need to tell \$ResursCheckout where the iframe container is. The iframe container is the html element you put the iframe into, so \$ResursCheckout knows where to render the iframe. The hold feature, is present to tell \$ResursCheckout to not handle the payment instantly on the submission, but wait for your backend to finalize what's need to be done in the order creation. See the section **onSubmit()** for more information about the "unholding" part.

```
var paymentSessionId = 'session-id-returned-from-the-post-request';
var baseUrl = 'https://the-base-url';

$ResursCheckout.create({
    paymentSessionId: paymentSessionId,
    baseUrl: baseUrl,
    hold: true,
    containerId: 'resursbank_rco_container'
});
```

onSubmit(), release(), recreate()

This is the iframe-side-based trigger, when the submit button is clicked, which initializes the purchase at Resurs Bank.

```
$ResursCheckout.onSubmit(function (event) {
    // Order creation. For example, you want to put a backend call here, to create the order in your store.
    // When your backend is completed here, you can tell $ResursCheckout to "unhold" its waiting feature:
    $ResursCheckout.release();

    // If something goes sideways here, you can either handle the problem yourself or make use of the feature
    // recreate() to recreate the iframe:
    $ResursCheckout.recreate();
})
```

onCustomerChange(), onPaymentChange()

In its simplest way, depending on you platform, this is the place you want to either temporary store customer- and payment method data until **onSubmit** are triggered, or you could send a backend request to store this data in a backend session until order creation is ready.

```
$ResursCheckout.onCustomerChange(function (event) {
    resursBankRcoDataContainer.rco_customer = event;
});
$ResursCheckout.onPaymentChange(function (event) {
    resursBankRcoDataContainer.rco_payment = event
});
```

onPaymentFail()

An early state failure notification

```
$ResursCheckout.onPaymentFail(function (event) {
    // Execute what needs to be done with failed orders.
});
```

Undocumented features

The below list is further features in \$ResursCheckout that is not yet fully documented.

Function/event	Arguments	Description
onError		
onPaymentSuccess		
onReady		
onResize		
create	<pre>{ "paymentSessionId": "session-id-uuid", "baseUrl": "the-base-url", "hold": true, "containerId": "webpage-element-name" }</pre>	
lock		
unlock		
hold		
abortHold		

ResursCheckoutJS Legacy

Table of contents

- Requirements
 - Dependencies and script support
- What this script can do
 - The moment 22
 - tldr; Take a look at this page if the description below is too complex.
- What this script can't do
- What you should be careful with
- Can I use updatePaymentReference?
- Can I change the look of the iframe?
- The events in the JS module

Download

CHANGELOG

- Version 0.09
- Version 0.08
- Version 0.07

Usage examples

- Common handlers
 - Catching customer data before the checkout button
 - Failing purchases
 - Change the shape of the iframe
 - Sending own messages to the iframe
- Script debugging



RCOJS continuity

Project is discontinued.

Requirements

- A pre-set js-variable: RESURSCHECKOUT_IFRAME_URL (*OMNICHECKOUT_IFRAME_URL are kept for backwards compatibility*) needs to be set from your store, to define where events are sent from.
Note: If you're using a document.ready-control, RESURSCHECKOUT_IFRAME_URL must be set before this
- Make sure that shopUrl is sent and matches with the target iframe URL (not the complete one, you only need the first parts for example <https://http.p.host.name> without trailing slashes), when creating the iframe at API level.
- A html element, with an id, that holds the iframe (default id: resurs-checkout-container)

Dependencies and script support

ResursCheckoutJS has *no external* dependencies like jQuery, etc, since we want to be sure that the script fits in any web store you implement it in. The target browser must however support JSON-parsing ([JSON.stringify](#)).

What this script can do

The moment 22

tldr; Take a look at this page if the description below is too complex.

To create an order at Resurs Bank, Resurs Bank requires an order id, preferably from the store. Sometimes, store platforms can't return the order id before, the order has been created in the store first. This might be a problem, since creating an order this way, first, does not give very much information about the customer itself.

Resurs Checkout has a communication interface, that allows a web store to pick up data from the checkout-iframe and place the order as soon as the customer has been confirmed "OK" at Resurs Bank. However, the store still needs to return a valid order id. To get around this problem, your store plugin may create a temporary order id first, that it uses to initiate a payment at Resurs Bank. As the communication interface in the checkout allows background activities, it is also possible to create the order as soon as the customer confirms the payment at checkout. At this moment, the backend plugin has the chance to synchronize the proper order id, created by the store, with the temporary initiated order id at Resurs Bank.

ResursCheckoutJS is a small framework-like script, that handles the primary part of those issues. ResursCheckoutJS has *no external dependencies* like jQuery, etc, since we want to be sure that the script fits in any web store and plugin you decide to use it in. The script is written so that you can put it on a webpage without having it primarily activated (to avoid colliding with other scripts). It utilizes the message handler in the Resurs Checkout iframe, so that you can push an order into the store in the background, as the checkout is completed at Resurs Bank.

Communicating with the iframe is however not required in any matter, unless you are planning to book the order as described above, *before* the booking is made at Resurs Bank. This means that you actually can live completely without this interactivity.

What this script can't do

- It does not update the iframe summary if the cart changes. This should be backend (See <https://test.resurs.com/docs/x/BoJM#ResursCheckout-checkout/payments/{orderReference}-PUT>)

What you should be careful with

Be aware of expected behaviour glitches

Running in "interceptor"-mode might be, if you do not take some precautions, unsafe. Always make sure that you, during an intercepted booking synchronize your checkout cart in the store with Resurs Bank. This can be best done by, before creating the order in the webstore and before releasing the interceptor handle to Resurs Bank, by an extra PUT ([read here!](#)) as content can be changed for example in other browser tabs during the checkout. Since the iframe does not know what's going on in your store while it is open, it is highly recommended to secure this.

Can I use updatePaymentReference?

Yes. If you have needs to update your payment reference (orderid) before sending order confirmation to ecommerce (via the confirm order button), you can do an `updatePaymentReference` call first in, for example, your backend environment. As this reference has to be created before the order is created in Resurs Bank, you could use the `setBookingCallback()` function in ResursCheckoutJS to trigger this update event, when your order has been created locally.

Can I change the look of the iframe?

Yes and no. It depends on what you are planning to change. Fonts and button colors are primary possible to change, but you have to tell Resurs Bank about it. As of ResursCheckoutJS 0.09, there is however a event callback available, that gives you the opportunity to change the iframe style. This normally includes the iframe body background and the border shape.

The events in the JS module

Events that this module catches from Resurs Checkout (and other events sent from the checkout):

- **checkout:loaded** - Handled by this script when the iFrame has loaded and is ready
Available callback function: `resursCheckoutIframeReady`
- **checkout:user-info-change** - Stored until checkout is finished
Available callback function: `setCustomerChangedEventCallback`
- **checkout:payment-method-change** - Stored until checkout is finished
Available callback function: `setCustomerChangedEventCallback`
- **checkout:booking-order** - Passed with necessary customer data to a callback, or dropped if no callback is set
Available callback function: `setBookingCallback`
- **checkout:purchase-failed** - If there is a problem with the payment at Resurs Bank, this event will be sent back from the iframe
Available callback function: `setPurchaseFailCallback`
- **checkout:purchase-denied** - If there is a problem with the payment at Resurs Bank customer level, this event will be sent back from the iframe
Available callback function: `setPurchaseDeniedCallback`
- **purchase-button-clicked** - Event that runs on the purchase-button-click (*mind the typo, this is the current event name that is sent from the checkout.*)

Download

Latest version is always available at <https://bitbucket.org/resursbankplugins/resurscheckoutjs>

CHANGELOG

Version 0.09

Key	Summary	T	Created	Updated	Due	Assignee	Reporter	P	Status	Resolution
RCOJS-12	JS-event callback for loaded iframe	<input checked="" type="checkbox"/>	2017-06-02	2017-06-02		Unassigned	Thomas Tornevall		DONE	Done

[1 issue](#)

Version 0.08

Key	Summary	T	Created	Updated	Due	Assignee	Reporter	P	Status	Resolution
RCOJS-10	Safari browsers might fail triggering on checkout events	BUG	2017-05-17	2017-05-17		Marcus Gullberg	Thomas Tornevall	⚠️	DONE	Done

1 issue

Version 0.07

Key	Summary	T	Created	Updated	Due	Assignee	Reporter	P	Status	Resolution
RCOJS-9	Support multiple iframe-elements on page	✓	2017-04-10	2017-04-11		Thomas Tornevall	Thomas Tornevall	==	DONE	Done
RCOJS-8	Securing backwards compatibility by element scanning	✓	2017-04-08	2017-04-11		Thomas Tornevall	Thomas Tornevall	✗	DONE	Done

2 issues

Usage examples

Common handlers

Make sure omnicheckout.js (or the minified version) loads first, so put it between the <head></head>-tags. Initialization of the iframe communication works like the example below.

Initialization

```
// The RESURSCHECKOUT_IFRAME_URL should be dynamically set up, but this is an example, so we have put a static url here, to test environment.  
// Make sure you are not using this example in your production environment, since that will fail.  
var RESURSCHECKOUT_IFRAME_URL = "https://omnitest.resurs.com";  
var resursCheckout = ResursOmni();  
resursCheckout.init();
```

To catch data from the iframe, you need a callback function set up on your client side. You may either register this callback or use a default function, to catch the data. Creating the function that handles bookings may look like this:

Handle iframe communication

```
/**  
 * Handle booking event from OmniCheckout frame  
 * @param omniJsObject  
 */  
function omniBookEvent(checkoutJsObject) {  
    // start handling your bookings here  
    // -- ajax calls and stuff --  
    // on success, confirm the booking back to the iframe here  
    ResursOmni.confirmOrder(true);  
}
```

The omniJsObject is received from the iframe and normally contains enough data (ssn, address and deliveryaddress) to book the order on the client side store. Deliveryaddress is empty if it matches with the regular address (billing).

To use your own callback function you could do something like this instead:

Defining own callback for bookings

```
ResursOmni.setBookingCallback(  
    function(checkoutJsObject) {  
        // My own callback for handling bookpayment  
    }  
) ;
```

Catching customer data before the checkout button

In some cases, a shop needs to fetch address data before the checkout is being made, to make sure that shipping is properly set up. For example, depending on where you live (postal codes), you might want to update your shop with valid delivery options for the city the customer lives in. As of v0.06 you can set up a callback in ResursCheckoutJS to handle customer data through your backend, long before the customer clicks on the checkout button. This set up is as easy as setBookingCallback:

Defining own callback for bookings

```
ResursOmni.setCustomerChangedEventCallback(  
    function(checkoutCustomerData) {  
        // My own callback for handling backend upgrades, preferably through an AJAX call to yourself  
    }  
) ;
```



"Remember me"

Customer data will also be available, if the customer has checked the "Remember me"-option.

Failing purchases

The feature for sending back messages to the shop, that has been added in v0.04 works as the above setBookingCallback. If a purchase fails, because of for example a credit denial, Resurs Checkout will now send back a purchase-failed message to the parent page, that OmniJS can catch. If the function is set with **setPurchaseFailCallback()**, you can make your store cancel the current order and eventually restart with a new in cases where your store does not support automatic cancellations.

setPurchaseFailCallback

```
ResursOmni.setBookingCallback(  
    function() {  
        // My own callback for handling automatic cancellations if the payment fails  
    }  
) ;
```

Change the shape of the iframe

Directly after the init()-function you could set up a function that looks like something like this:

```
resursCheckout.setOnIframeReady(function(iframeElement) {  
    iframeElement.setAttribute('style', 'border-radius:10px; box-shadow:10px 5px 5px #990000; border:1px solid black; background:#FFFF00;');  
});
```

The event is, as you can see, sending over the element of where the iframe is located - so everything that is not cross-site-scripting related can actually be done here. Normally, the only things covered here, is the look of the iframe. This means, at least, that you can change the background color and the border shape of the iframe.

Sending own messages to the iframe

Since the iframe already supports extended communication like updates of the order summary, accepted terms and conditions, etc - and OmniJS currently don't - there is an implementation of sending your own messages to the iframe that may solve your request.

This example is an alternative to the confirmOrder()-call above:

Send own messages to the iframe

```
ResursOmni.postMessage({  
    eventType: 'checkout:order-status',  
    orderReady: true  
});
```

Script debugging

Data that comes from the iframe, can be debugged, by enabling debug mode like this:

Debug mode

```
omniCheckout.setDebug(true);
```

Pre-filling customer data in Resurs Checkout

Optimize the Checkout for Returning Customers

Customers are more likely to complete purchases if they can checkout quickly and accurately. Resurs Bank creates a fast and accurate checkout experience by prefilling, auto-completing and validating personal details.

Send us the customer's information, if you already have it, just before you render the checkout. When you create the order you may choose to add data about the customer. We recommended this feature for customers who have already registered with your website.

If you choose to add customer data, the customer-defined fields on the checkout form will be pre-populated, which will increase your conversion.

Customer-defined fields that can be pre-populated:

governmentId ->
mobile -> Mobile phone number
email -> E-mail address
firstName -> Given name
lastName -> Family name
addressRow1 -> Street address, first line.
addressRow2 -> Street address, second line.
postalCode -> Postal/post code
postalArea -> City
countryCode -> ISO 3166 alpha-2. Country. Has to be same country as the webservice-account is created for

POST-example containing customer data

```
{
  "orderLines": [
    {
      "artNo": "ART123",
      "description": "Product description",
      "quantity": 2,
      "unitMeasure": "kg",
      "unitAmountWithoutVat": 200,
      "vatPct": 25
    }
  ],
  "customer": {
    "governmentId": "198305147715",
    "mobile": "0771112233",
    "email": "test@resurs.se",
    "deliveryAddress": {
      "fullName": "Vincent Williamsson Alexandersson",
      "firstName": "Vincent",
      "lastName": "Williamsson Alexandersson",
      "addressRow1": "Glassgatan 15",
      "addressRow2": null,
      "postalArea": "Göteborg",
      "postalCode": "41655",
      "countryCode": "SE"
    }
  },
  "successUrl": "http://google.com",
  "backUrl": "http://failblog.cheezburger.com/",
  "paymentCreatedCallbackUrl": "https://test.se",
  "shopUrl": "https://test.se"
}
```

updatePaymentReference notices

The rest call updatePaymentReference usually accept updates of references without any further notices except for a HTTP code 200 (success). However, there's a few ways to handle exceptions at it throws a bit different content depending on what happens in the flow. Unfortunately, due to long-time-development, the exceptions may be a bit inconsequent. The below examples explains how the errors are returned from the rest API. Another thing to know about this flow is that our library component EComPHP can usually parry those exceptions and give back a proper response regardless of the outcome.

Event	Description	JSON content	Code	EComPHP comments
Successful updatePaymentReference	No further explanation. We are successful.		200	Will return success
Colliding updatePaymentReference	When you run same credentials over more platforms or for some reason, you are trying to update a payment reference that is already created at Resurs Bank	{"errorCode": "intValue", "description": "errorMessage"}	8	Initially throws 40X, but catches extended error messages in the http error body. Will throw exception with code 8.
Second updatePaymentReference	Running updatePaymentReference where the reference is already updated.	{"errorCode": "stringBasedCode", "detailedMessage": "errorMessage"}	404 or PAYMENT_SESSION_NOT_FOUND	Error body contains an extended error message. However, since the errorCode is usually PAYMENT_SESSION_NOT_FOUND , PHP does not support this with a standard Exception(errorString, errorCode) as errorCode must be defined as a long value. EComPHP will in this case fall back to the http-head error (404), with only the error string intact ("payment session does not exist").

As we can see here, the errors are always different, so trying to implement this in your own solution should not be very hard as long as you listen to all of the alternatives . However, if you only parse the http head, you won't be able to match the correct exception since many of the rest calls is using the **404 not found**-method. By using EComPHP ([1.3.18 or above!](#)) in this case (if it is possible) you could probably trust both 8 and 404 since they mean different things (404 can probably always be considered a successful update for example).

Handling above exceptions with extended libraries like EComPHP

See [EComPHP Error Adaption](#).

Resurs Checkout POS

Overview

Resurs Checkout POS is an iFrame browser-based payment gateway that can be used in a Point of Sale-system (POS). Payments are finalized automatically since the customer is to be expected to receive the product directly. If you however submit an offer "Betala Senare" (will be explained in depth below), the payment will only be authorized and will need to be finalized. Note that all purchases in the POS-environment require e-signing. The Resurs Checkout POS part is marked with a blue frame in the screenshot below.

Try Resurs Checkout POS for yourself

[Demo-shop](#)

Screenshot from the iFrame

1. Hämta kundens uppgifter <input type="text" value="Ååmmdd-xxxx"/> <input type="text" value="8305147715"/> <input type="button" value="Byt person"/>	2. Välj betalsätt <input checked="" type="radio"/> Faktura Fakturan skickas till angiven e-postadress. Läs villkoren för faktura här . <input type="radio"/> Delbetalning <input type="radio"/> Nytt kort <input type="button"/> <input type="radio"/> Befintligt kort <input type="button"/>	3. Slutför <input checked="" type="checkbox"/> Jag försäkrar att till kreditförmöndare lämnade uppgifter är korrekta och fullständiga och att kunden givit sitt tillstånd till en eventuell kreditprövning. Summa: 2500.00 kr <input type="button" value="Genomför köp"/>
---	---	---

[Ändra uppgifter](#)

Annan leveransadress [\(i\)](#)

Safe shopping by:

© 2017 Resurs Bank AB (publ) Org.nr 516401-0208, Box 222 09, SE-250 24 Helsingborg, info@resursbank.se, www.resursbank.se
[Användarvillkor Resurs Checkout](#)

What can I find here?

- Overview
 - Try Resurs Checkout POS for yourself
 - Payment methods offered within the checkout
- Resurs Checkout Implementation
 - Authentication
 - End points (URLs)
 - Errors
 - Test Data
 - API Overview
 - 1. Initiate Resurs CheckOut POS
 - 3. Get information regarding the payment
 - 4. Retrieve the Order
 - 5. Manage Order
 - API - HTTP Request
 - POST /payments/{orderReference}
 - orderLines
 - metaData
 - successUrl
 - backUrl
 - paymentCreatedCallbackUrl
 - shopUrl
 - Response
 - Errors
 - GET /payments/{orderReference}
 - Delayed Checkout - "Betala sen"
 - Sequence diagram Resurs Bank Checkout POS - Delayed Checkout
 - Restrictions

Payment methods offered within the checkout

Resurs Checkout provides an easy to use checkout for payment methods such as invoice and partial payments, . The cashier enters the customer's personal identity number (government ID) to retrieve address details which will populate the address fields automatically. Other mandatory information that needs to be entered are cellphone number and email address.

Supported payment methods are;

- Invoice
- Partial payment
- Credit cards
- Branded credit cards and application for a new credit card
- VISA/Mastercard (Delayed Checkout only)
- Swish (Delayed Checkout only - Sweden)

Note that all payment methods are to be discussed through the merchant and Resurs account manager. Onboarding can never setup a payment method without an order from the account manager.

Resurs Checkout Implementation

To use Resurs Checkout you first have to set up an agreement with Resurs bank. This is when you get your merchant ID and your credentials.

This tutorial how to render the Resurs Checkout HTML snippet on your checkout page.



TLS

Resurs Bank does not support Transport Layer Security (TLS) prior to Version 1.2

Authentication

Resurs Checkout API uses HTTP Basic Auth for authentication. To call the services you need to provide the credentials you received when setting up the agreement with Resurs Bank. [Read more here](#).

End points (URLs)

- <https://poscheckout.resurs.com/pos> - Production Environment
- <https://postest.resurs.com/pos> - Test Environment

Errors

Resurs APIs use HTTP 4xx 5xx status codes together with error messages to handle errors.

Test Data

For testing your implementation, follow the below link to test civic numbers with different scenarios:

[Sweden](#)

API Overview

URI	Method	Action
/payments/{orderReference}	POST	<p>Creates a payment session including:</p> <ul style="list-style-type: none">• Shopping cart content (mandatory)• Customer information (Optional) <p>Output:</p> <ul style="list-style-type: none">• iFrame/script HTML• Payment session id
/payments/{orderReference}	GET	Retrieves a payment. Used to fetch existing payments (not sessions).



Want to run **DELETE** on payment session?

Running a POST with the same orderReference a second time will overwrite the first payment session, making the previous one obsolete.

1. Initiate Resurs CheckOut POS

When you want to display Resurs Checkout you need to initiate the payment. For details about this see [/pos/payments/ - POST](#).

When you initiate the payment you may choose to [pre-filling data about the customer](#). We recommended this feature to optimize the Checkout for Returning Customers.

The JSON response from the *POST payments* call contains a unique *paymentSessionId* generated by Resurs Bank. The JSON response payload also contains the HTML snippet that you need to include on your page to render the checkout.

3. Get information regarding the payment

To receive information regarding the payment whether the customer has completed the purchase or not, you can GET-poll with *paymentSessionId* as the ID which is received in the initializePayment-response. [Read more...](#)

4. Retrieve the Order

Use *orderReference* to fetch the order from Resurs Bank. [Read more...](#)

5. Manage Order

Now you can choose to manage the order by implement APIs in [After Shop Service API](#) or manually in Resurs GUI [Manipulate Payments in Merchant Portal](#).

Note that the payments are automatically finalized. You will only need to implement the creditPayment for the After Shop API (see Delayed Checkout - "Betala sen" for exception).

API - HTTP Request

POST /payments/{orderReference}

This service sets up the payment session and should be called whenever you want to display the checkout. It will return the paymentId and html to be included in the online shop.

Request parameters

```
{  
    "orderLines": [  
        {  
            "artNo" : "1",  
            "description" : "Order line:1",  
            "quantity" : 1.0,  
            "unitMeasure" : "pcs",  
            "unitAmountWithoutVat" : 500.0,  
            "vatPct" : 25.0,  
                "totalAmountWithVat": "625",  
                "totalVatAmount": "125"  
        },  
        {  
            "artNo" : "2",  
            "description" : "Order line:2",  
            "quantity" : 1.0,  
            "unitMeasure" : "pcs",  
            "unitAmountWithoutVat" : 10.0,  
            "vatPct" : 25.0,  
                "totalAmountWithVat": "12.5",  
                "totalVatAmount": "2.5",  
                "type": "ORDER_LINE"  
        },  
        {  
            "artNo" : "80",  
            "description" : "Discount",  
            "quantity" : 1.0,  
            "unitMeasure" : "pcs",  
            "unitAmountWithoutVat" : -5.0,  
            "vatPct" : 0.0,  
                "totalAmountWithVat": "-5.0",  
                "totalVatAmount": "0.0",  
                "type": "DISCOUNT"  
        },  
        {  
            "artNo" : "90",  
            "description" : "Shipping & Handling",  
            "quantity" : 1.0,  
            "unitMeasure" : "pcs",  
            "unitAmountWithoutVat" : 5.0,  
            "vatPct" : 0.0,  
                "totalAmountWithVat": "5.0",  
                "totalVatAmount": "0.0",  
                "type": "SHIPPING_FEE"  
        }  
    ],  
    "metaData" : [  
        {  
            "key": "key1",  
            "value": "value1"  
        },  
        {  
            "key": "key2",  
            "value": "value2"  
        },  
        {  
            "key": "CustomerId",  
            "value": "String 20 chars"  
        },  
        {  
    ]
```

```

        "key": "invoiceExtRef",
        "value": "String 46 chars"
    }
],
"customer": {
    "governmentId": "",
    "mobile": "0707123456",
    "email": "test@resurs.se",
    "deliveryAddress": {
        "firstName": "Daniel",
        "lastName": "Johnsson",
        "addressRow1": "Ekslingan 8",
        "addressRow2": null,
        "postalArea": "Helsingborg",
        "postalCode": "25467",
        "countryCode": "SE"
    }
},
"successUrl": "https://shop.representative.com/order/12345/success/",
"backUrl": "https://shop.representative.com/order/12345/checkout/",
"paymentCreatedCallbackUrl": "https://shop.representative.com/backend/order/12345/",
    "storeId": "9012",
"shopUrl": "https://mystore.test.com"
}

```

Attribute	Data type	Mandatory	Description
orderLines		X	Object that represents an orderline.
orderLines/artNo	varchar (100)	X	Article number.
orderLines /description	text	X	Article name. Will be shown on the invoice. (Nillable? yes)
orderLines /quantity	decimal (15,5)	X	Quantity for the specific article.
orderLines /unitMeasure	varchar (15)	X	Unit type for the orderline e.g. kg, pcs.
orderLines /unitAmountWithOutVat	decimal (15,5)	X	Net price per unit.
orderLines /vatPct	decimal (15,5)	X	Vat per unit (%).
orderLines /totalAmountWithVat	decimal (15,5)		The total amount of the orderline, including VAT (do not forget to multiply by quantity)
orderLines /totalVatAmount	decimal (15,5)		The total VAT-amount of the orderline
metaData			Extra meta data for the payment. Recognized metadata
metaData/key	varchar (30)		Meta data key
metaData/value	text		Meta data value
customer			Used for pre-filling customer data in the iFrame. It is possible to only send partial customer data. If customer has a cookie from Resurs, this will override all customer data that is sent in below
customer /governmentId	varchar (15)		GovernmentID of the customer - will automatically fetch country registered address of the customer (not applicable for Norway)
customer/mobile	varchar (45)		Mobilephone number.

customer/email	varchar (300)		Email address.
customer /deliveryAddress			If no delivery address is sent and governmentId is sent, delivery address will automatically be the customers country registered address (not applicable for Norway).
customer /deliveryAddress /firstName	varchar (100)		Given name
customer /deliveryAddress /lastName	varchar (100)		Family name
customer /deliveryAddress /addressRow1	varchar (250)		Street address, first line
customer /deliveryAddress /addressRow2	varchar (250)		Street address, second line
customer /deliveryAddress /postalArea	varchar (100)		City
customer /deliveryAddress /postalCode	varchar (45)		Postal/post code
customer /deliveryAddress /countryCode	varchar (2)		Country code of the delivery address - SE/NO/FI/DK. Has to be same country as the webservice-account is created for. If you're unsure, please contact us.
successUrl	text	X	<p>The address of the web page where the customer will be redirected when the payment has completed successfully.</p> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;">  The URL must contain a unique identifier so that the URL can be linked to the correct order at the merchant. </div> <p>Use a SALT when creating the identifier for greater security.</p>
backUrl	text	X	<p>The address of the web page where the customer will be redirected if the payment is <u>not</u> completed successfully or if the customer navigates back him-/herself. A message will be shown and it will be the same regardless of error or back navigation, so a generic message is recommended.</p>
paymentCreatedCallbackUrl	text	-	<p>A callback URL that will be called when a payment is created. If a user closes the webrowser before he/she has been redirected to the success Url.</p> <p>There is also a possibility to register a permanent callback BOOKED. It's more safe to use the BOOKED callback since you can use a SALT in that call. See Available callback-types</p> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;">  If both paymentCreatedCallbackUrl and BOOKED are used, paymentCreatedCallbackUrl is prioritized. </div> <div style="margin-top: 10px;">  If you need full support of callbacks we recommend you to not using this callback setup since it will stop other important callbacks to run. </div>
storeId	varchar	-	Used with permission from Resurs Bank, if you have a chain of stores. storeID defines which store in the chain it is.

shopUrl	text	X	<p>The shop's domain, this is used to send user events from the iframe to the shop. How to do the communication is described at OmniCheckoutJS-Communication. It is recommended to always send this value into your payload.</p> <div style="border: 1px solid red; padding: 10px;"> <p>Required at iframe event level</p> <p>To be able to communicate correctly with the iframe events, this URL must be set - <i>without trailing slashes</i>.</p> <p>The format of URL must be HTTP_HOST only, not a full URI.</p> <p>Example:</p> <p>Your shop full Url is located at https://mystore.test.com/full/path/to/checkout/uri/</p> <p>The <code>shopUrl</code> will be:</p> <pre><code>"shopUrl": "https://mystore.test.com"</code></pre> </div>
---------	------	---	--

Response

Response

```
{
  "paymentSessionId": "ebfffab5a-3746-41c6-a020-f0cf11949897",
  "html": "<iframe id=\"rco-checkout-app-frame\" src=\"https://postest.resurs.com/web/dist/omni-checkout.html?ebfffab5a-3746-41c6-a020-f0cf11949897\" frameborder=\"0\" width=\"100%\" scrolling=\"no\"></iframe><script type=\"text/javascript\" src=\"https://postest.resurs.com/web/dist/js/oc-shop.js\"></script>"}
```

Attribute	Description	
paymentSessionId	The ID for the payment session.	
html	The html snippet that must be displayed by the shop in order to display Resurs Checkout.	

Errors

HTTP status code	Type	Example
400	Application/json	{ "message": "OrderReference already exists" }
401	Application/json	{ "message": "Bad credentials" }
500	Application/json	{ "message": "Internal error" }

GET /payments/{orderReference}

Response

```
{
  "id": "Omni-Session_2342345",
  "totalAmount": 500,
  "metaData": [
    {
      "key": "advertisement",
      "value": "false"
    }
  ],
  "limit": 500,
  "paymentDiffs": [
    {
      "type": "AUTHORIZE",
      "amount": 500
    }
  ]
}
```

```

        "transactionId": null,
        "created": -5648895212887551616,
        "createdBy": "SHOP_FLOW",
        "paymentSpec": {
            "specLines": [
                {
                    "id": "0",
                    "artNo": "ART123",
                    "description": "Product description",
                    "quantity": 2,
                    "unitMeasure": "kg",
                    "unitAmountWithoutVat": 200,
                    "vatPct": 25,
                    "totalVatAmount": 100,
                    "totalAmount": 500
                }
            ],
            "totalAmount": 500,
            "totalVatAmount": 100,
            "bonusPoints": 0
        },
        "orderId": null,
        "invoiceId": null,
        "documentNames": []
    }
],
"customer": {
    "governmentId": "8305147715",
    "address": {
        "fullName": "William Williamsson Eliassson",
        "firstName": "William",
        "lastName": "Eliassson",
        "addressRow1": "Ekslingan 13",
        "addressRow2": null,
        "postalArea": "Helsingborg",
        "postalCode": "25024",
        "country": "SE"
    },
    "phone": "+46701234567",
    "email": "info@resurs.se",
    "type": "NATURAL"
},
"deliveryAddress": {
    "fullName": "Testsson Test",
    "firstName": "Test",
    "lastName": "Testsson",
    "addressRow1": "Ekslingan 9",
    "addressRow2": null,
    "postalArea": "Helsingborg",
    "postalCode": "254 67",
    "country": "SE"
},
"booked": -5648516529463551616,
"finalized": null,
"paymentMethodId": "VISA SUPER CARD",
"paymentMethodName": "Mocked PSP",
"fraud": false,
"frozen": false,
"status": [
    "DEBITABLE"
],
"storeId": "107999",
"paymentMethodType": "PAYMENT_PROVIDER",
"totalBonusPoints": 0
}

```

HTTP status code	Type	Example
200	Application/json	OK

401	Application/json	{ "message": "Bad credentials" }
404	Application/json	{ "message": "The order does not exist"}
500	Application/json	{ "message": "Internal error." }

Delayed Checkout - "Betala sen"

NOTE!

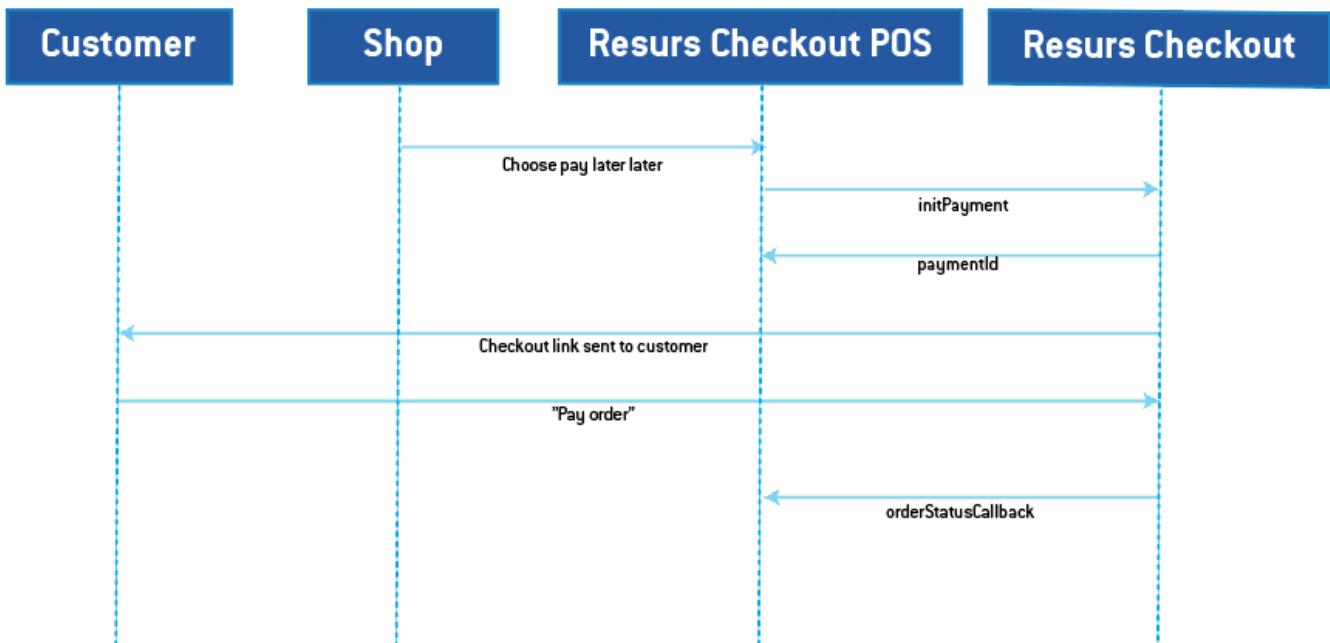
This feature is only available if this has been agreed to with Resurs salesperson

A feature in the Resurs Checkout POS is to send out what we call a "delayed checkout", via "Betala sen". The cashier is only to insert the customer's information and set a valid to-date. Then, a sms/email with a URL to the iFrame containing the cart is sent to the customer. The customer can then choose to make purchase whenever they want.

The payment/order is not created at Resurs Bank by this point, Resurs Bank simply keeps the payment session alive for a period of time of the cashiers choice, default 24 hours.

Using this feature, you will need to finalize the payment via [finalizePayment](#) in the [After Shop API](#).

Sequence diagram Resurs Bank Checkout POS - Delayed Checkout



Restrictions

Payment is possible for persons, but not for companies.

Resurs Checkout has no IPv6 support.

If card payment via Nets is to be offered in the Delayed Checkout, a requirement from Nets is max 32 characters for {orderReference}.

If Delayed Checkout is to be offered, each store has to provide their own shop terms, e-mailed to Onboarding Support in a public URL, prior to going live

When going live, you have to host your own public success- and backUrl to where the customers/clerk are to go when the payment is completed or failed

The orderReference-sequence is per webservice account, not per storeId. As a solution, if you are running storeId, you can add the store number as a prefix to the sequence

Pre-filling customer data in Resurs Checkout POS

Optimize the Checkout for Returning Customers

Customers are more likely to complete purchases if they can checkout quickly and accurately. Resurs Bank creates a fast and accurate checkout experience by prefilling, auto-completing and validating personal details.

Send us the customer's information, if you already have it, just before you render the checkout. When you create the order you may choose to add data about the customer. We recommended this feature for customers who have already registered with your website.

If you choose to add customer data, the customer-defined fields on the checkout form will be pre-populated, which will increase your conversion.

Customer-defined fields that can be pre-populated:

governmentId ->
mobile -> Mobile phone number
email -> E-mail address

POST-example containing customer data

```
{  
    "orderLines": [  
        {  
            "artNo": "ART123",  
            "description": "Product description",  
            "quantity": 2,  
            "unitMeasure": "kg",  
            "unitAmountWithoutVat": 200,  
            "vatPct": 25  
        }  
    ],  
    "customer": {  
        "governmentId": "198305147715",  
        "mobile": "0771112233",  
        "email": "test@resurs.se",  
        "deliveryAddress": {  
            "fullName": "Vincent Williamsson Alexandersson",  
            "firstName": "Vincent ",  
            "lastName": "Williamsson Alexandersson",  
            "addressRow1": "Glassgatan 15",  
            "addressRow2": null,  
            "postalArea": "Göteborg",  
            "postalCode": "41655",  
            "countryCode": "SE"  
        }  
    },  
    "successUrl": "http://google.com",  
    "backUrl": "http://failblog.cheezburger.com/"  
}
```

Resurs Checkout PUSH

Overview

Resurs self service Checkout is a POS-service where you send in the cart to Resurs, where Resurs provide the URI for the iFrame and the partner sends the link to the customer themselves, via for example text message or email. With this solution, there is no need to open a browser in the POS to send out the payment link.

The screenshot shows a cart summary titled "Att betala". It lists three items: Test A (1 unit, 1567.43 kr), Test B (1 unit, 685.73 kr), and Robatt (1 unit, -25 kr). The total amount is 2 228.16 kr.

Produkt	Antal	Summa
Test A 101	1	1 567,43 kr
Test B 102	1	685,73 kr
Robatt 901	1	-25 kr
		Totalt 2 228,16 kr

The screenshot shows the payment selection step. It includes fields for shipping address (filled with placeholder data) and delivery address (unchecked). The payment methods listed are:

- Bankkort Visa/Mastercard (selected)
- Fakturakampanj
- Debetning
- Betala med Resurs-kort
- Faktura privat
- Kreditkort Visa/Mastercard
- Ansök och betala med nytt Resurs-kort

A summary box shows "Summa: 2 228,16 kr" and a note about a 25 kr discount. A "Genomför köp" button is present, along with a privacy notice and a link to the terms of service.

Safe shopping by © 2021 Resurs Bank AB (publ) Org.nr 516401-0208
Box 222 09, SE-250 24 Helsingborg, info@resursbank.se, resursbank.se

What can I find here?

- Overview
 - Payment methods offered within the checkout
- Resurs self service checkout implementation
 - Authentication
 - End points (URLs)
 - Errors
 - Test Data
 - API Overview
 - 1. Initiate Resurs self service checkout
 - 2. Retrieve the Order
 - 3. Manage Order
 - API - HTTP Request
 - POST /payments/{orderReference}
 - orderLines
 - metaData
 - successUrl
 - backUrl
 - Response
 - Errors
 - GET /payments/{orderReference}
 - Response
 - Restrictions/prerequisites

Payment methods offered within the checkout

Resurs Checkout provides an easy to use checkout for payment methods such as invoice and partial payments. The customer enters personal identity number (government ID) to retrieve address details which will populate the address fields automatically. Other mandatory information that needs to be entered are cellphone number and email address.

Supported payment methods are;

- Invoice
- Partial payment
- Credit cards
- Branded credit cards and application for a new credit card
- VISA/Mastercard
- Swish (Sweden)

Note that all payment methods are to be discussed through the merchant and Resurs account manager. Onboarding can never setup a payment method without an order from the account manager.

Resurs self service checkout implementation

To use Resurs self service checkout you first have to set up an agreement with Resurs bank. This is when you get your merchant ID and your credentials.

This tutorial is how to render the Resurs self service checkout link which can be sent via SMS or email by you.



TLS

Resurs Bank does not support Transport Layer Security (TLS) prior to Version 1.2

Authentication

Resurs Checkout API uses HTTP Basic Auth for authentication. To call the services you need to provide the credentials you received when setting up the agreement with Resurs Bank. [Read more here](#).

End points (URLs)

- <https://poscheckout.resurs.com/pos> - Production Environment
- <https://postest.resurs.com/pos> - Test Environment

Errors

Resurs APIs use HTTP 4xx 5xx status codes together with error messages to handle errors.

Test Data

For testing your implementation, follow the below link to test civic numbers with different scenarios:

[Sweden](#)

API Overview

URI	Method	Action
/payments/{orderReference}	POST	<p>Creates a payment session including:</p> <ul style="list-style-type: none">• Shopping cart content (mandatory)• Customer information (Optional) <p>Output:</p> <ul style="list-style-type: none">• Link to checkout location• Payment session id

/payments/{orderReference}	GET	Retrieves a payment. Used to fetch existing payments (not sessions).
----------------------------	------------	--

i Want to run **DELETE** on payment session?

Running a POST with the same orderReference a second time will overwrite the first payment session, making the previous one obsolete.

1. Initiate Resurs self service checkout

First you have to initiate the payment. For details about this see [/checkout/payments/ - POST](#).

When you initiate the payment you may choose to [pre-filling data about the customer](#). We recommended this feature to optimize the Checkout for Returning Customers.

The JSON response from the *POST payments* call contains a unique *paymentSessionId* generated by Resurs Bank. The JSON response payload also contains the checkout location that you need to include in sms or email.

2. Retrieve the Order

To retrieve information whether the customer has completed his/hers purchase you have two choices;

- Poll the *orderReference* to fetch the order from Resurs Bank via GET in REST ([Read more...](#)) or via *getPayment* from AfterShopService ([Read more...](#))
- Implement callbacks ([Read more...](#)) where the BOOKED-callback will give you confirmation that the order is booked at Resurs end.

Note that you can only have one set of callbacks per production account. If you are unsure of the amount of production accounts you are to use, please contact us.

3. Manage Order

If the payment has been booked at Resurs - you can now choose to manage the order by implementing APIs in [After Shop Service API](#) or manually in Resurs GUI [Manipulate Payments in Merchant Portal](#).

Note that for the moment, the payments are automatically finalized. Therefore the only action available when payment is done is crediting (creditPayment in AfterShopService).

Soon, when the flag for finalizelfBooked true/false is released and you are sending in *false* as the value, you can use debiting (finalizePayment) and annulling (annullPayment) as well as addition (additionalDebitOfPayment).

Note that crediting (creditPayment) only can be done when the payment has been finalized.

To see what actions you can take on a payment depending on its status, [Read more here...](#)

To further understand what the Resurs-status means, [Read more here...](#)

API - HTTP Request

POST /payments/{orderReference}

This service sets up the payment session and should be called whenever you want to display the checkout. It will return the paymentId and html to be included in the online shop.

Request parameters

```
{
  "orderLines": [
    {
      "artNo" : "1",
      "description" : "Order line:1",
      "quantity" : 1.0,
      "unitMeasure" : "pcs",
      "unitAmountWithoutVat" : 500.0,
      "vatPct" : 25.0,
      "totalAmountWithVat": "625",
      "totalVatAmount": "125"
      "link" : "https://www.linktoproductpage1.se"
    },
    {
      ...
    }
  ]
}
```

```

        "artNo" : "2",
        "description" : "Order line:2",
        "quantity" : 1.0,
        "unitMeasure" : "pcs",
        "unitAmountWithoutVat" : 10.0,
        "vatPct" : 25.0,
            "totalAmountWithVat": "12.5",
            "totalVatAmount": "1.25"
            "link" : "https://www.linktoproductpage2.se",
            "type": "ORDER_LINE"
    },
    {
        "artNo" : "80",
        "description" : "Discount",
        "quantity" : 1.0,
        "unitMeasure" : "pcs",
        "unitAmountWithoutVat" : -5.0,
        "vatPct" : 0.0,
            "totalAmountWithVat": "-5.0",
            "totalVatAmount": "0.0"
            "type": "DISCOUNT"
    },
    {
        "artNo" : "90",
        "description" : "Shipping & Handling",
        "quantity" : 1.0,
        "unitMeasure" : "pcs",
        "unitAmountWithoutVat" : 5.0,
        "vatPct" : 0.0,
            "totalAmountWithVat": "5.0",
            "totalVatAmount": "0.0"
            "type": "SHIPPING_FEE"
    }
],
"metaData" : [
    {
        "key": "key1",
        "value": "value1"
    },
    {
        "key": "key2",
        "value": "value2"
    },
    {
        "key": "CustomerId",
        "value": "String 20 chars"
    },
    {
        "key": "invoiceExtRef",
        "value": "String 46 chars"
    }
],
"customer": {
    "governmentId": "",
    "mobile": "0707123456",
    "email": "test@resurs.se",
    "deliveryAddress": {
        "firstName": "Daniel",
        "lastName": "Johnsson",
        "addressRow1": "Ekslingan 8",
        "addressRow2": null,
        "postalArea": "Helsingborg",
        "postalCode": "25467",
        "countryCode": "SE"
    }
},
"delayedCheckoutOptions" : {
    "useDelayed" : true,
    "expiresAfterNumberOfDays" : "2",
    "expiresAfterNumberOfHours" : "10",
    "finalizeIfBooked" : false
}

```

```

},
"successUrl": "https://shop.representative.com/order/12345/yourSuccessPage/",
"backUrl": "https://shop.representative.com/order/12345/yourFailPage/"
}

```

Attribute	Data type	Mandatory	Description
orderLines		X	Object that represents an orderline.
orderLines/artNo	varchar (100)	X	Article number.
orderLines /description	text	X	Article name. Will be shown on the invoice. (Nillable? yes)
orderLines/quantity	decimal (15,5)	X	Quantity for the specific article.
orderLines /unitMeasure	varchar (15)	X	Unit type for the orderline e.g. kg, pcs.
orderLines /unitAmountWithout Vat	decimal (15,5)	X	Net price per unit.
orderLines/vatPct	decimal (15,5)	X	Vat per unit (%).
orderLines /totalAmountWithVat	decimal (15,5)		The total amount of the orderline, including VAT (do not forget to multiply by quantity)
orderLines /totalVatAmount	decimal (15,5)		The total VAT-amount of the orderline
orderLines/link	URI		Link to the product page. Will be available to the customer above the Resurs Checkout iFrame in the cart summary. Resurs inbed the link in a "Se produkt"-button.
orderLines/type		-	The type of the order line. <i>Optional</i> . It can be one of the following: ORDER_LINE - the same as if nothing is sent in, it will result in a normal order line. DISCOUNT - will be presented as a discount in Resurs Checkout SHIPPING_FEE - will be presented as a shipping fee in Resurs Checkout.
metaData			Extra meta data for the payment. Recognized metadata
metaData/key	varchar (30)		Meta data key
metaData/value	text		Meta data value
customer			Used for pre-filling customer data in the iFrame. It is possible to only send partial customer data. If customer has a cookie from Resurs, this will override all customer data that is sent in below
customer /governmentId	varchar (15)		GovernmentID of the customer - will automatically fetch country registered address of the customer (not applicable for Norway)
customer/mobile	varchar (45)		Mobilephone number.
customer/email	varchar (300)		Email address.
customer /deliveryAddress			If no delivery address is sent and governmentId is sent, delivery address will automatically be the customers country registered address (not applicable for Norway).
customer /deliveryAddress /firstName	varchar (100)		Given name
customer /deliveryAddress /lastName	varchar (100)		Family name

customer /deliveryAddress /addressRow1	varchar (250)		Street address, first line
customer /deliveryAddress /addressRow2	varchar (250)		Street address, second line
customer /deliveryAddress /postalArea	varchar (100)		City
customer /deliveryAddress /postalCode	varchar (45)		Postal/post code
customer /deliveryAddress /countryCode	varchar (2)		Country code of the delivery address - SE/NO/FI/DK. Has to be same country as the webservice-account is created for. If you're unsure, please contact us.
delayedCheckoutOptions		x	
delayedCheckoutOptions/useDelayed		x	Is to always be set as "true"
delayedCheckoutOptions /expiresAfterNumberOfDay		x	Number of days the payment link will be active. After this value of time, the payment link will automatically be cancelled by Resurs. Total maximum time the link can be active is 60 days. Can be used in unison with expiresAfterNumberOfHours, which time then will be added together. Default time, if no value is added, is 30 days.
delayedCheckoutOptions /expiresAfterNumberOfHours			Number of hours the payment link will be active. After this value of time, the payment link will automatically be cancelled by Resurs. Total maximum time the link can be active is 60 days. Can be used in unison with expiresAfterNumberOfDays, which time then will be added together.
delayedCheckoutOptions /finalizeIfBooked	true /false	x	Value: Default: true. Will automatically finalize (debit) the payment when order is booked on Resurs side. If your chosen value is false, you will need to manually finalize (debit) the payment via either AfterShopFlow or Merchant Portal.
successUrl	text	X	The address of the web page where the customer will be redirected when the payment has completed successfully.
backUrl	text	X	The address of the web page where the customer will be redirected if the payment is not completed successfully or if the customer navigates back him-/herself. A message will be shown and it will be the same regardless of error or back navigation, so a generic message is recommended.
storeId	varchar	-	Used with permission from Resurs Bank, if you have a chain of stores. storeID defines which store in the chain it is.

Response

Response
{ "paymentSessionId": "9e3aeb06-cafa-4149-96eb-5d5e806bbd27", "delayedCheckoutLocation": "https://web-integration-delayed-checkout.integration.resurs.com/9e3aeb06-cafa-4149-96eb-5d5e806bbd27" }

Attribute	Description	
paymentSessionId	The ID for the payment session.	
delayedCheckoutLocation	Link to checkout location. Max 100 characters.	

Errors

HTTP status code	Type	Example
400	Application/json	{ "message": "OrderReference already exists" }

401	Application/json	{ "message": "Bad credentials" }
500	Application/json	{ "message": "Internal error" }

GET /payments/{orderReference}

Response

```
{
    "id": "Omni-Session_2342345",
    "totalAmount": 500,
    "metaData": [
        {
            "key": "advertisement",
            "value": "false"
        }
    ],
    "limit": 500,
    "paymentDiffs": [
        {
            "type": "AUTHORIZE",
            "transactionId": null,
            "created": -5648895212887551616,
            "createdBy": "SHOP_FLOW",
            "paymentSpec": {
                "specLines": [
                    {
                        "id": "0",
                        "artNo": "ART123",
                        "description": "Product description",
                        "quantity": 2,
                        "unitMeasure": "kg",
                        "unitAmountWithoutVat": 200,
                        "vatPct": 25,
                        "totalVatAmount": 100,
                        "totalAmount": 500
                    }
                ],
                "totalAmount": 500,
                "totalVatAmount": 100,
                "bonusPoints": 0
            },
            "orderId": null,
            "invoiceId": null,
            "documentNames": []
        }
    ],
    "customer": {
        "governmentId": "8305147715",
        "address": {
            "fullName": "William Williamsson Eliassson",
            "firstName": "William",
            "lastName": "Eliassson",
            "addressRow1": "Ekslingan 13",
            "addressRow2": null,
            "postalArea": "Helsingborg",
            "postalCode": "25024",
            "country": "SE"
        },
        "phone": "+46701234567",
        "email": "info@resurs.se",
        "type": "NATURAL"
    },
    "deliveryAddress": {
        "fullName": "Testsson Test",
        "firstName": "Test",
        "lastName": "Testsson",
        "addressRow1": "Ekslingan 9",
        "addressRow2": null,
        "city": "Helsingborg",
        "zip": "25024",
        "country": "SE"
    }
}
```

```

        "postalArea": "Helsingborg",
        "postalCode": "254 67",
        "country": "SE"
    },
    "booked": -5648516529463551616,
    "finalized": null,
    "paymentMethodId": "VISA SUPER CARD",
    "paymentMethodName": "Mocked PSP",
    "fraud": false,
    "frozen": false,
    "status": [
        "DEBITABLE"
    ],
    "storeId": "107999",
    "paymentMethodType": "PAYMENT_PROVIDER",
    "totalBonusPoints": 0
}

```

Response

HTTP status code	Type	Example
200	Application/json	OK
401	Application/json	{ "message": "Bad credentials" }
404	Application/json	{ "message": "The order does not exist"}
500	Application/json	{ "message": "Internal error." }

Restrictions/prerequisites

SMS/Email service which can send the payment link to the customer

Payment is possible for persons, but not for companies.

Resurs Checkout has no IPv6 support.

If card payment via Nets is to be offered in the Delayed Checkout, a requirement from Nets is max 32 characters for {orderReference}.

If Swish (SE) is to be offered in the Delayed Checkout, a requirement from Swish is max 36 characters for {orderReference}

Each store has to provide their own shop terms, e-mailed to Onboarding Support in a public URL, prior to going live (will be available for customer in Resurs Checkout)

When going live, you have to host your own public success- and backUrl to where the customers are to go when the payment is completed or failed

The orderReference-sequence is per webservice account, not per storeId. As a solution, if you are running storeId, you can add the store number as a prefix to the sequence

Only one set of callbacks can be configured per production account

Payments are authorized 90 days before annulment. After that, the payment must be done once again from scratch

Pre-filling customer data

Optimize the Checkout for Returning Customers

Customers are more likely to complete purchases if they can checkout quickly and accurately. Resurs Bank creates a fast and accurate checkout experience by prefilling, auto-completing and validating personal details.

The customer information is sent in the initial POST and will be auto-populated when the iFrame is presented in the customer's browser.

Customer-defined fields that can be pre-populated:

governmentId, mobile number, email, name and address.

POST-example containing customer data

```
{
  "orderLines": [
    {
      "artNo": "ART123",
      "description": "Product description",
      "quantity": 2,
      "unitMeasure": "kg",
      "unitAmountWithoutVat": 200,
      "vatPct": 25
    }
  ],
  "customer": {
    "governmentId": "198305147715",
    "mobile": "0771112233",
    "email": "test@resurs.se",
    "deliveryAddress": {
      "fullName": "Vincent Williamsson Alexandersson",
      "firstName": "Vincent",
      "lastName": "Williamsson Alexandersson",
      "addressRow1": "Glassgatan 15",
      "addressRow2": null,
      "postalArea": "Göteborg",
      "postalCode": "41655",
      "countryCode": "SE"
    }
  },
  "successUrl": "http://google.com",
  "backUrl": "http://failblog.cheezburger.com/"
}
```

Simplified Flow API

Get started

1. Get [Test URLs](#)
2. Download soapUI and run some of the tests.
3. Figure out what functionality you'll need: [Concepts and Domain](#)
4. Setup your [development environment](#)
5. Start coding the [Shop flow](#).
6. Implement [Callbacks](#)



HTTPS and certificate

Note that we only support HTTPS, both in test and production environment. You must have a valid and issued (not self signed) certificate for this.

Implement Shop flow.

The examples below demonstrates how to interact with the Application Service.

Authentication is done by using the http [basic authentication mechanism](#).

1. Get all available payment methods - [getPaymentMethods](#)

This call can be cached for 24 hours.

Request getPaymentMethods

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:sim="http://ecommerce.resurs.com/v4/msg/simplifiedshopflow">
    <soapenv:Header/>
    <soapenv:Body>
        <sim:getPaymentMethods>
            <!--Optional:-->
            <language>?</language>
            <!--Optional:-->
            <customerType>?</customerType>
            <!--Optional:-->
            <purchaseAmount>?</purchaseAmount>
        </sim:getPaymentMethods>
    </soapenv:Body>
</soapenv:Envelope>
```

Response getPaymentMethodsResponse

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body>
        <ns3:getPaymentMethodsResponse xmlns:ns3="http://ecommerce.resurs.com/v4/msg/simplifiedshopflow" xmlns:ns2="http://ecommerce.resurs.com/v4/msg/exception">
            <return>
                <id>INVOICE</id>
                <description>Faktura via e-post</description>
                <legalInfoLinks>
                    <appendPriceLast>false</appendPriceLast>
                    <endUserDescription>Allmänna villkor</endUserDescription>
                    <url>https://test.resurs.com/documenthandler/Dokument.pdf?customerType=natural&&docType=commonTerms&&land=SE&&language=sv</url>
                </legalInfoLinks>
                <legalInfoLinks>
                    <appendPriceLast>true</appendPriceLast>
                    <endUserDescription>Standardiserad europeisk konsumentkreditinformation (SEKKI)</endUserDescription>
                    <url>https://test.resurs.com/sekki-mock/sekki?bankProductId=LG686069&&chainId=107&&countryCode=SE&&storeId=1000&&amount=</url>
                </legalInfoLinks>
            </return>
        </ns3:getPaymentMethodsResponse>
    </soap:Body>
</soap:Envelope>
```

```

<legalInfoLinks>
    <appendPriceLast>true</appendPriceLast>
    <endUserDescription>Prisinformation</endUserDescription>
    <url>https://test.resurs.com/priceinfo-mock/prisskyltning.html?countryCode=SE&amp;
authorizedBankproductId=LG686069&amp;cardType=&amp;storeId=265&amp;representativeId=107&amp;creditAmount=</url>
</legalInfoLinks>
<minLimit>10.00</minLimit>
<maxLimit>50000.00</maxLimit>
<type>INVOICE</type>
<customerType>NATURAL</customerType>
<specificType>INVOICE</specificType>
</return>
<return>
    <id>PARTPAYMENT</id>
    <description>Delbetalning via post</description>
    <legalInfoLinks>
        <appendPriceLast>false</appendPriceLast>
        <endUserDescription>Allmänna villkor</endUserDescription>
        <url>https://test.resurs.com/documenthandler/Dokument.pdf?customerType=natural&amp;
docType=commonTerms&amp;land=SE&amp;language=sv</url>
    </legalInfoLinks>
    <legalInfoLinks>
        <appendPriceLast>true</appendPriceLast>
        <endUserDescription>Standardiserad europeisk konsumentkreditinformation (SEKKI)<
/&endUserDescription>
        <url>https://test.resurs.com/sekki-mock/sekki?bankProductId=LG686069&amp;chainId=107&amp;
countryCode=SE&amp;storeId=1000&amp;amount=</url>
    </legalInfoLinks>
    <legalInfoLinks>
        <appendPriceLast>true</appendPriceLast>
        <endUserDescription>Prisinformation</endUserDescription>
        <url>https://test.resurs.com/priceinfo-mock/prisskyltning.html?countryCode=SE&amp;
authorizedBankproductId=LG686069&amp;cardType=&amp;storeId=265&amp;representativeId=107&amp;creditAmount=</url>
        </legalInfoLinks>
        <minLimit>10.00</minLimit>
        <maxLimit>50000.00</maxLimit>
        <type>REVOLVING_CREDIT</type>
        <customerType>NATURAL</customerType>
        <specificType>PART_PAYMENT</specificType>
    </return>
    <return>
        <id>CARD</id>
        <description>Kortköp</description>
        <minLimit>0</minLimit>
        <maxLimit>2147483647</maxLimit>
        <type>CARD</type>
        <customerType>NATURAL</customerType>
        <specificType>CARD</specificType>
    </return>
</ns3:getPaymentMethodsResponse>
</soap:Body>
</soap:Envelope>

```

Get all available payment methods by using `getPaymentMethods` with `language`, `customerType` and `purchaseAmount` parameters.
By using this parameters for each booking you will only get the payment methods applicable.
In your application, present Resurs Bank payment methods with the corresponding price information links.
For each payment method a "[read more...](#)" is required. Use the price information links from `getPaymentMethods` or present html from `getCostOfPurchaseHtml` in a pop-up.
Present the Payment Methods in the same order as in response from `getPaymentMethods`.

2. Fetch customer address:

- [getAddress \(SE\)](#)

Enter government id.

Request getAddress

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:sim="http://ecommerce.resurs.com/v4/msg/simplifiedshopflow">
  <soapenv:Header/>
  <soapenv:Body>
    <sim:getAddress>
      <governmentId>198305147715</governmentId>
      <customerType>NATURAL</customerType>
    </sim:getAddress>
  </soapenv:Body>
</soapenv:Envelope>
```

Response getAddress

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns2:getAddressResponse xmlns:ns3="http://ecommerce.resurs.com/v4/msg/exception" xmlns:ns2="http://ecommerce.resurs.com/v4/msg/simplifiedshopflow">
      <return>
        <fullName>Vincent Williamsson Andersson</fullName>
        <firstName>Vincent</firstName>
        <lastName>Andersson</lastName>
        <addressRow1>Glassgatan 15</addressRow1>
        <postalArea>Göteborg</postalArea>
        <postalCode>41655</postalCode>
        <country>SE</country>
      </return>
    </ns2:getAddressResponse>
  </soap:Body>
</soap:Envelope>
```

- getAddressByPhone (NO)

Enter phonenumber

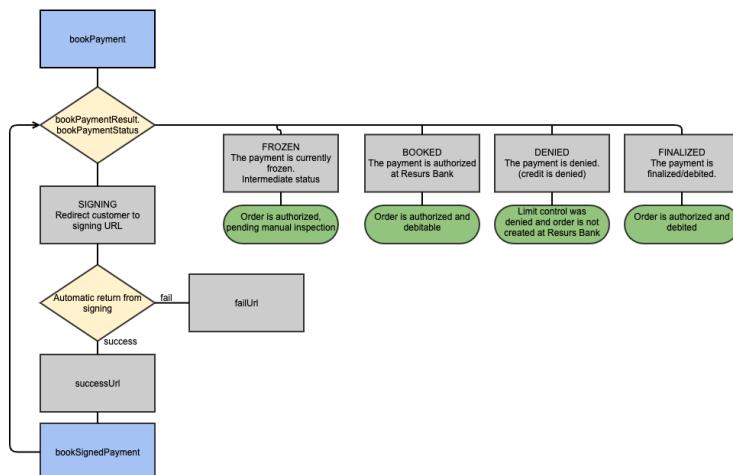
Request getAddressByPhone

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:sim="http://ecommerce.resurs.com/v4/msg/simplifiedshopflow">
  <soapenv:Header/>
  <soapenv:Body>
    <sim:getAddressByPhone>
      <phoneNumber>40000001</phoneNumber>
      <customerIpAddress>80.80.80.80</customerIpAddress>
    </sim:getAddressByPhone>
  </soapenv:Body>
</soapenv:Envelope>
```

Response getAddressByPhone

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns3:getAddressResponse xmlns:ns3="http://ecommerce.resurs.com/v4/msg/simplifiedshopflow" xmlns:ns2="http://ecommerce.resurs.com/v4/msg/exception">
      <return>
        <fullName>Bjarne Bock Krohg</fullName>
        <firstName>Bjarne</firstName>
        <lastName>Krohg</lastName>
        <addressRow1>Prinsens Gate 14</addressRow1>
        <postalArea>Bergen</postalArea>
        <postalCode>5014</postalCode>
        <country>NO</country>
      </return>
    </ns3:getAddressResponse>
  </soap:Body>
</soap:Envelope>
```

3. Book a payment - bookPayment



All the data concerning the payment is entered through this method.

If a customer wants to deliver the order to a different address than the billing address the field deliveryAddress is used. This will trigger a customer authentication.

- Enter customer information. (See our [Customer data - Regular expressions](#) for different fields.)
- Select a Resurs Bank payment method.
- Enter additional information the shop might want.
- Confirm purchase.

Request bookPayment

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Body xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <bookPayment xmlns="http://ecommerce.resurs.com/v4/msg/simplifiedshopflow">
      <paymentData xmlns="">
        <preferredId>210000002</preferredId>
        <preferredTransactionId xsi:nil="true"/>
        <paymentMethodId>PARTPAYMENT</paymentMethodId>
        <customerIpAddress>127.0.0.1</customerIpAddress>
        <waitForFraudControl>true</waitForFraudControl>
        <annulIfFrozen>false</annulIfFrozen>
        <finalizeIfBooked>false</finalizeIfBooked>
      </paymentData>
      <orderData xmlns="">
        <specLines>
          <id>10</id>
```

```

<artNo>255121</artNo>
<description>HDMI 3m</description>
<quantity>1</quantity>
<unitMeasure>pcs</unitMeasure>
<unitAmountWithoutVat>199.2</unitAmountWithoutVat>
<vatPct>25.0</vatPct>
<totalVatAmount>49.80</totalVatAmount>
<totalAmount>249.00</totalAmount>
</specLines>
<specLines>
<id>20</id>
<artNo>101622</artNo>
<description>Slim Case</description>
<quantity>1</quantity>
<unitMeasure>pcs</unitMeasure>
<unitAmountWithoutVat>159.2</unitAmountWithoutVat>
<vatPct>25.0</vatPct>
<totalVatAmount>39.80</totalVatAmount>
<totalAmount>199.00</totalAmount>
</specLines>
<specLines>
<id>30</id>
<artNo>FREIGHT</artNo>
<description>Shipping and Handling</description>
<quantity>1</quantity>
<unitMeasure>pcs</unitMeasure>
<unitAmountWithoutVat>39.2</unitAmountWithoutVat>
<vatPct>25</vatPct>
<totalVatAmount>9.80</totalVatAmount>
<totalAmount>49.00</totalAmount>
</specLines>
<totalAmount>497.00</totalAmount>
<totalVatAmount>99.40</totalVatAmount>
</orderData>
<metaData xmlns="">
<key>Express Shipping</key>
<value>True</value>
</metaData>
<customer xmlns="">
<governmentId>198305147715</governmentId>
<address>
<fullName>Vincent Williamsson</fullName>
<firstName>Vincent</firstName>
<lastName>Williamsson</lastName>
<addressRow1>Glassgatan 15</addressRow1>
<postalArea>Göteborg</postalArea>
<postalCode>41655</postalCode>
<country>SE</country>
</address>
<phone>070112233</phone>
<email>test@resurs.se</email>
<type>NATURAL</type>
<cellPhone>070112233</cellPhone>
<yourCustomerId>321456987</yourCustomerId>
<deliveryAddress>
<fullName>Vincent Williamsson</fullName>
<firstName>Vincent</firstName>
<lastName>Williamsson</lastName>
<addressRow1>Glassgatan 15</addressRow1>
<postalArea>Göteborg</postalArea>
<postalCode>41655</postalCode>
<country>SE</country>
</deliveryAddress>
<contactGovernmentId/>
</customer>
<card xmlns="" xsi:nil="true"/>
<signing xmlns="">
<successUrl>http://www.mysite.se/Checkout/ResursDone?orderNumber=210000001</successUrl>
<failUrl>http://www.mysite.se/Checkout/ResursFailed</failUrl>
</signing>
<invoiceData xmlns="" xsi:nil="true"/>

```

```

    </bookPayment>
</s:Body>
</s:Envelope>

```

Response

```

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body>
        <ns3:bookPaymentResponse xmlns:ns3="http://ecommerce.resurs.com/v4/msg/simplifiedshopflow" xmlns:ns2="http://ecommerce.resurs.com/v4/msg/exception">
            <return>
                <paymentId>210000002</paymentId>
                <bookPaymentStatus>SIGNING</bookPaymentStatus>
                <signingUrl>https://test.resurs.com/mock-mock/authenticate?resursToken=4f017bc9-07ec-45b4-b4c5-f6d206766393</signingUrl>
                <approvedAmount>497</approvedAmount>
                <customer>
                    <governmentId>8305147715</governmentId>
                    <address>
                        <fullName>Vincent Williamsson Andersson</fullName>
                        <firstName>Vincent</firstName>
                        <lastName>Andersson</lastName>
                        <addressRow1>Glassgatan 15</addressRow1>
                        <postalArea>Göteborg</postalArea>
                        <postalCode>41655</postalCode>
                        <country>SE</country>
                    </address>
                    <phone>+4670112233</phone>
                    <email>test@resurs.se</email>
                    <type>NATURAL</type>
                </customer>
            </return>
        </ns3:bookPaymentResponse>
    </soap:Body>
</soap:Envelope>

```

When you submit the bookPayment the response will be one of these bookPaymentStatus:

- FINALIZED - The payment is finalized. Notify the customer and continue with the normal flow.
- BOOKED - The payment is booked and you will have to finalize it on your own. To finalize booked payments automatically you can set a flag in the bookPayment.
- FROZEN - The payment is currently frozen. This typically means that there is something that needs further investigation before the payment can be finalized.
- SIGNING - The payment requires Signing. Redirect customer to signing URL. After a successful signing the order needs to be booked with bookSignedPayment.
- DENIED - The payment is denied. The payment method can't be used for the customer and the flow stops. Another payment method without credit might work for the customer.

4. Book signed payment - [bookSignedPayment](#)

After a successful signing the order needs to be booked:

Request

```

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:sim="http://ecommerce.resurs.com/v4/msg/simplifiedshopflow">
    <soapenv:Header/>
    <soapenv:Body>
        <sim:bookSignedPayment>
            <paymentId>210000002</paymentId>
        </sim:bookSignedPayment>
    </soapenv:Body>
</soapenv:Envelope>

```

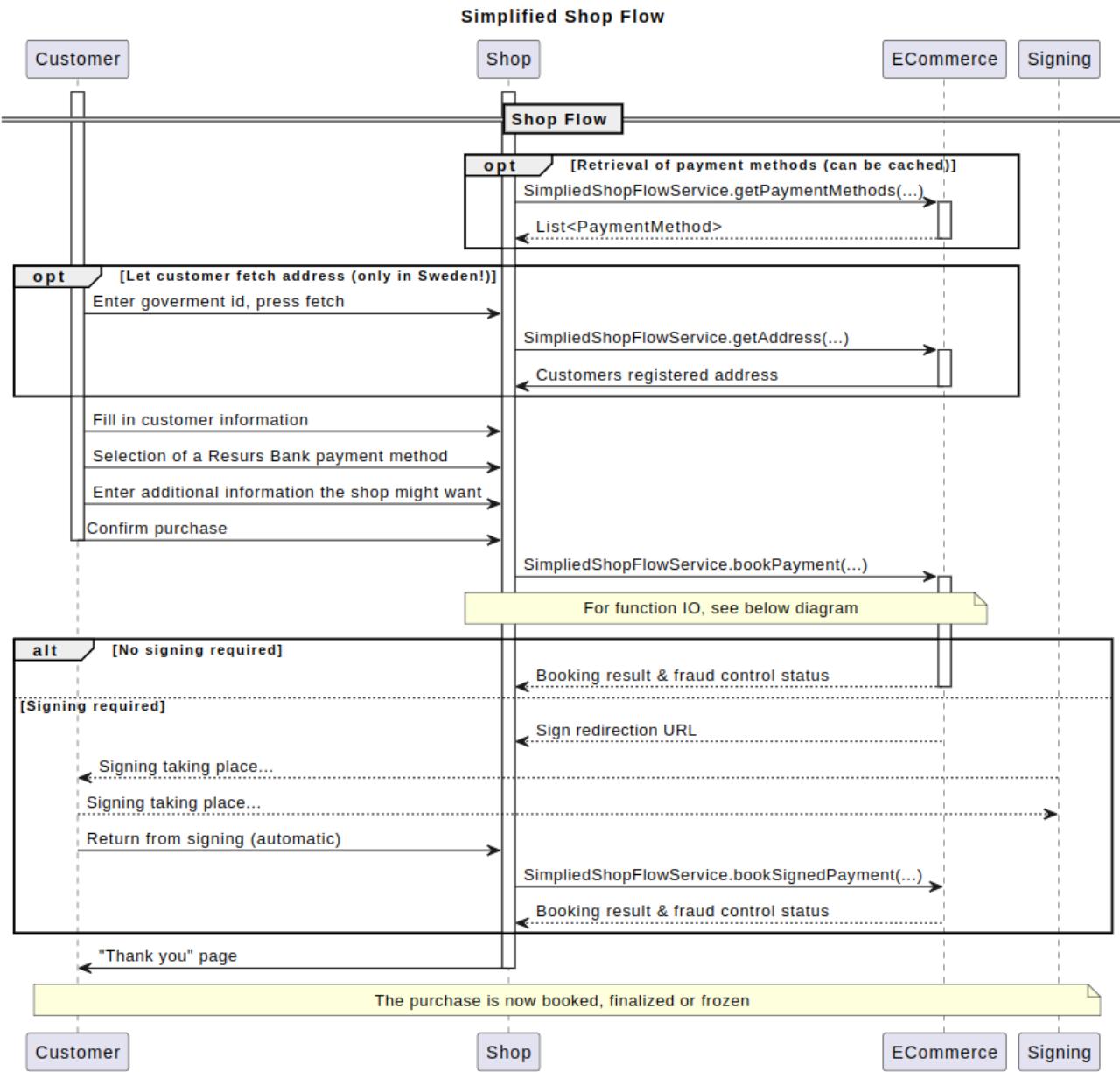
Response

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns2:bookPaymentResponse xmlns:ns3="http://ecommerce.resurs.com/v4/msg/exception" xmlns:ns2="http://ecommerce.resurs.com/v4/msg/simplifiedshopflow">
      <return>
        <paymentId>210000002</paymentId>
        <bookPaymentStatus>BOOKED</bookPaymentStatus>
        <signingUrl xsi:nil="true" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
        <approvedAmount>497</approvedAmount>
        <customer>
          <governmentId>8305147715</governmentId>
          <address>
            <fullName>Vincent Williamsson Andersson</fullName>
            <firstName>Vincent</firstName>
            <lastName>Andersson</lastName>
            <addressRow1>Glassgatan 15</addressRow1>
            <postalArea>Göteborg</postalArea>
            <postalCode>41655</postalCode>
            <country>SE</country>
          </address>
          <phone>+4670112233</phone>
          <email>test@resurs.se</email>
          <type>NATURAL</type>
        </customer>
      </return>
    </ns2:bookPaymentResponse>
  </soap:Body>
</soap:Envelope>
```

The purchase is now booked, finalized, frozen or denied.

You can choose to manage the order in [Resurs Merchant Portal](#) gui, or by using Resurs Order Management API [After Shop Service](#).

Implement Callbacks



getAddress (SE)

customerType

Description: The type of customer.

Value	Description
LEGAL	The customer is a legal customer, i.e. a company.
NATURAL	The customer is a natural customer, i.e. a person.

This function is only available in Sweden! (due to legal reasons)

Introduction

This function is offered as a service, since most webshops collect address information early in the check out flow. This function fetches address information about an individual or a company. If the customer exists, its registered address is returned. If the customer is of type LEGAL (organization /company) firstname and lastname will be null (empty). This address may be used as billing - and delivery address. The customerIpAddres is to prevent address harvesting. We allow only a limited number of queries within a time frame. Bashing will result in an [error](#) when the threshold is exceeded.

getAddress

Retrieves address information. Note that the customerType parameter is optional right now, but in short notice this will be required (minOccurs=1)

Input (Literal)

Name	Type	Occurs	Nillable?	Description
governmentId	string	1..1	No	The government identity of the customer for which to retrieve the address.
customerType	customerType	0..1	No	The type of customer to retrieve. In many cases, this is easily determined from the government identity, but for Swedish companies in sole proprietorship, the same identity is used for both the person as a natural customer, and the company as a legal customer.
customerIpAddress	string	0..1	No	The IP address from which the customer has accessed the service. To prevent bashing. This parameter is mandatory even if it has minOccurs set to zero.

Output(Literal)

Name	Type	Occurs	Nillable?	Description
return	address	1..1	No	If a match could be made, the customer address.

Faults

Name	Content	Description
ECommerceErrorException	ECommerceError	Failed to retrieve the address information. See error for details.

The customerType and customerIpAddress parameters are **mandatory** even if the schema says they're not. (They were late additions, and we had to do this way not to break compatibility)

Code Examples - Request & Response

getAddress

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:shop="http://ecommerce.resurs.com/v4/msg/simplifiedshopflow">
  <soapenv:Header/>
  <soapenv:Body>
    <shop:getAddress>
      <governmentId>7312195873</governmentId>
      <customerType>NATURAL</customerType>
      <customerIpAddress>80.80.80.80</customerIpAddress>
    </shop:getAddress>
  </soapenv:Body>
</soapenv:Envelope>
```

If the customer exists, its registered address is returned. If the customer is of type LEGAL (an organization/comapny) `firstName` and `lastName` will be null

getAddressResponse

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns3:getAddressResponse xmlns:ns3="http://ecommerce.resurs.com/v4/msg/shopflow" xmlns:ns2="http://ecommerce.resurs.com/v4/msg/exception">
      <return>
        <fullName>Lastname Firstname</fullName>
        <firstName>Firstname</firstName>
        <lastName>Lastname</lastName>
        <addressRow1>Street XX</addressRow1>
        <postalArea>City</postalArea>
        <postalCode>12345</postalCode>
        <country>SE</country>
      </return>
    </ns3:getAddressResponse>
  </soap:Body>
</soap:Envelope>
```

getPaymentMethods

getPaymentMethods

Retrieves detailed information on the payment methods available to the [representative](#).

Input (Literal)

Name	Type	Occurs	Nillable?	Description
language	String	0..1	Yes	The language code as defined by the ISO 639-1 standard, <i>optional</i> . If not specified the default language of the representatives country will be used. '[sv, no, da, fi]'
customerType	String	0..1	Yes	Filter Payment Methods based on the CustomerType, <i>optional</i> . '[NATURAL, LEGAL]'
purchaseAmount	String	0..1	Yes	Filter Payment Method based on the purchaseAmount, <i>optional</i> .

Output (Literal)

Name	Type	Occurs	Nillable?	Description
return	paymentMethod	0..*	No	Return a list of all payment methods available to the representative.

Faults

Name	Content	Description
ECommerceErrorException	ECommerceError	Failed to retrieve the payment methods. See error for details.

Introduction

Each payment consist of:

- An identity
- A description intended for the developer, but can be shown to the customer if you wish.
- minLimit & maxLimit: The payment method can only be used for purchases between these values.
- A specificType. This might be:
 - INVOICE: Essentially an [installment credit](#). Generates a PDF invoice when [finalized](#)
 - REVOLVING_CREDIT: [Revolving credits](#) and installment payments which do **not** generate invoices, in most cases this is a new Resurs Bank card.
 - CARD: A Resurs Bank card
 - PART_PAYMENT: You get the first payment slip the month after the purchase and you choose the payment plan then. The payment type does not generate an invoice.
- Legal info links. Must always be shown (at least the last link) where a Resurs Bank payment method is marketed in some way or can alternatively show information from [getCostOfPurchaseHtml](#) in a popup. This is a legal requirement.

Since payment methods do not contain information such as [logos](#), descriptions, fees, etc., it is likely that the webshop has to store the information about the methods of payment locally. The payment methods should be cached for about 24 hours to improve user experience by reducing loading time.

Resurs Bank may add/remove payment methods. A payment method is withdrawn from this listing 6 hours before it stops to work.

Code example - response from server

getPaymentMethodsResponse

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns3:getPaymentMethodsResponse xmlns:ns3="http://ecommerce.resurs.com/v4/msg/shopflow" xmlns:ns2="http://ecommerce.resurs.com/v4/msg/exception">
      <return>
        <id>6</id>
        <description>Company invoice</description>
        <legalInfoLinks>
```

```

        <appendPriceLast>false</appendPriceLast>
        <endUserDescription>General terms</endUserDescription>
        <url>https://secure.resurs.se/documenthandler/Dokument.pdf?customerType=legal&amp;
docType=commonTerms&amp;land=SE&amp;language=sv</url>
    </legalInfoLinks>
    <legalInfoLinks>
        <appendPriceLast>false</appendPriceLast>
        <endUserDescription>Standardiserad europeisk konsumentkreditinformation (SEKKI)<
/endUserDescription>
        <url>https://secure.resurs.se/documenthandler/Dokument.pdf?bankprodukt=NZ690101&amp;
kedja=107&amp;land=SE&amp;language=sv</url>
    </legalInfoLinks>
    <legalInfoLinks>
        <appendPriceLast>true</appendPriceLast>
        <endUserDescription>Prisinformation</endUserDescription><url>https://secure.resurs.se/documenthandler
/Dokument.pdf?bankprodukt=76189069&kedja=995264&land=SE
        <url>https://secure.resurs.se/priceinfo/prisskyltning.html?countryCode=SE&amp;
authorizedBankproductId=NZ690101&amp;representativeId=107&amp;creditAmount=</url>
    </legalInfoLinks>
    <minLimit>1.00</minLimit>
    <maxLimit>50000.00</maxLimit>
    <type>INVOICE</type>
        <specificType>INVOICE</specificType>
    <customerType>LEGAL</customerType>
    </return>
<return>
    <id>7</id>
    <description>Resurskort</description>
    <minLimit>0</minLimit>
    <maxLimit>2147483647</maxLimit>
    <type>CARD</type>
        <specificType>CARD</specificType>
    <customerType>NATURAL</customerType>
    <customerType>LEGAL</customerType>
</return>
<return>
    <id>8</id>
    <description>New card</description>
    <legalInfoLinks>
        <appendPriceLast>false</appendPriceLast>
        <endUserDescription>General terms</endUserDescription>
        <url>https://secure.resurs.se/documenthandler/Dokument.pdf?customerType=natural&amp;
docType=commonTerms&amp;land=SE&amp;language=sv</url>
    </legalInfoLinks>
    <legalInfoLinks>
        <appendPriceLast>false</appendPriceLast>
        <endUserDescription>Standardiserad europeisk konsumentkreditinformation (SEKKI)<
/endUserDescription>
        <url>https://secure.resurs.se/documenthandler/Dokument.pdf?bankprodukt=7B019069&amp;
kedja=107&amp;land=SE</url>
    </legalInfoLinks>
    <legalInfoLinks>
        <appendPriceLast>true</appendPriceLast>
        <endUserDescription>Prisinformation</endUserDescription>
        <url>https://secure.resurs.se/priceinfo/prisskyltning.html?countryCode=SE&amp;
authorizedBankproductId=7B019069&amp;representativeId=107&amp;creditAmount=</url>
    </legalInfoLinks>
    <minLimit>10.00</minLimit>
    <maxLimit>130000.00</maxLimit>
    <type>REVOLVING_CREDIT</type>
        <specificType>REVOLVING_CREDIT</specificType>
    <customerType>NATURAL</customerType>
    </return>
<return>
    <id>9</id>
    <description>Invoice</description>
    <legalInfoLinks>
        <appendPriceLast>false</appendPriceLast>
        <endUserDescription>Allmänna villkor</endUserDescription>
        <url>https://secure.resurs.se/documenthandler/Dokument.pdf?customerType=natural&amp;
docType=commonTerms&amp;land=SE&amp;language=sv</url>

```

```
</legalInfoLinks>
<legalInfoLinks>
    <appendPriceLast>false</appendPriceLast>
    <endUserDescription>Standardiserad europeisk konsumentkreditinformation (SEKKI)<
/endUserDescription>
    <url>https://secure.resurs.se/documenthandler/Dokument.pdf?bankprodukt=LG686069&amp;
kedja=107&amp;land=SE</url>
</legalInfoLinks>
<legalInfoLinks>
    <appendPriceLast>true</appendPriceLast>
    <endUserDescription>Prisinformation</endUserDescription>
    <url>https://secure.resurs.se/priceinfo/prisskyltning.html?countryCode=SE&amp;
authorizedBankproductId=LG686069&amp;representativeId=107&amp;creditAmount=</url>
</legalInfoLinks>
<minLimit>10.00</minLimit>
<maxLimit>50000.00</maxLimit>
<type>INVOICE</type>
    <specificType>REVOLVING_CREDIT</specificType>
<customerType>NATURAL</customerType>
</return>
</ns3:getPaymentMethodsResponse>
</soap:Body>
</soap:Envelope>
```

bookPayment

bookPayment

Books the payment.

Input (Literal)

Name	Type	Occurs	Nillable?	Description
paymentData	paymentData	1..1	No	The data that is for the payment, e.g. payment method etc.
orderData	paymentSpec	1..1	No	The payment specifications. What the payment should handle, the amounts, spec lines etc.
metaData	mapEntry	0..*	Yes	Extra meta data for the payment. Recognized metadata .
customer	extendedCustomer	1..1	No	The customer data. Here you specify the billing address, delivery address etc.
card	cardData	0..1	Yes	If the payment is related to a card/account, or if you apply for a new card/account.
signing	signing	1..1	No	For when a payment requires a Signing, contains customer URLs for a successful or failed signing.
invoiceData	invoiceData	0..1	Yes	The data for the invoice.
storeId	storeId	0..1	No	Used with permission from Resurs Bank, if you have a chain of stores. storeID defines which store in the chain it is.

Output (Literal)

Name	Type	Occurs	Nillable?	Description
return	bookPaymentResult	1..1	No	The result of the payment booking.

Faults

Name	Content	Description
ECommerceErrorException	ECommerceError	Failed to book the payment. See error for details.

Response

```
<soapenv:Body>
  <sim:bookPayment>
    <paymentData>
      <!--Optional:-->
      <preferredId>12345</preferredId>
      <preferredTransactionId>Transaction id 10</preferredTransactionId>
      <paymentMethodId>Invoice</paymentMethodId>
      <customerIpAddress>127.0.0.1</customerIpAddress>
      <!--Optional:-->
      <waitForFraudControl>true</waitForFraudControl>
      <!--Optional:-->
      <annulIfFrozen>false</annulIfFrozen>
      <!--Optional:-->
      <finalizeIfBooked>false</finalizeIfBooked>
    </paymentData>
    <orderData>
      <!--Zero or more repetitions:-->
      <specLines>
        <id>1</id>
        <artNo>1</artNo>
        <description>test</description>
        <quantity>1</quantity>
        <unitMeasure>st</unitMeasure>
        <unitAmountWithoutVat>100</unitAmountWithoutVat>
        <vatPct>25</vatPct>
      </specLines>
    </orderData>
  </sim:bookPayment>
</soapenv:Body>
```

```

        <totalVatAmount>25</totalVatAmount>
        <totalAmount>125</totalAmount>
    </specLines>
    <totalAmount>125</totalAmount>
    <!--Optional:-->
    <totalVatAmount>25</totalVatAmount>
</orderData>
<!--Zero or more repetitions:-->
<metaData>
    <key>xxx</key>
    <!--Optional:-->
    <value>tttt</value>
</metaData>
<metaData>
    <key>CustomerId</key>
    <value>String 20 chars</value>
</metaData>
<metaData>
    <key>invoiceExtRef</key>
    <value>String 46 chars</value>
</metaData>
<customer>
    <!--Optional:-->
    <governmentId>#${#TestCase#GovernmentID}</governmentId>
    <address>
        <!--Optional:-->
        <fullName>Test Testsson</fullName>
        <!--Optional:-->
        <firstName>Test</firstName>
        <!--Optional:-->
        <lastName>Testsson</lastName>
        <addressRow1>TEST </addressRow1>
        <!--Optional:-->
        <addressRow2>Test3</addressRow2>
        <postalArea>Helsingborg</postalArea>
        <postalCode>25220</postalCode>
        <country>SE</country>
    </address>
    <!--Optional:-->
    <email>test@test.com</email>
    <type>NATURAL</type>
    <cellPhone>0707112233</cellPhone>
    <!--Optional:-->
    <yourCustomerId>Cust-1000</yourCustomerId>
    <!--Optional:-->
    <deliveryAddress>
        <fullName>Test Testsson</fullName>
        <firstName>Test</firstName>
        <lastName>Testsson</lastName>
        <addressRow1>Test gatan 25</addressRow1>
        <postalArea>Helsingborg</postalArea>
        <postalCode>25220</postalCode>
        <country>SE</country>
    </deliveryAddress>
</customer>
<signing>
    <successUrl>https://shop.representative.com/order/12345/success/</successUrl>
    <failUrl>https://shop.representative.com/order/12345/fail/</failUrl>
</signing>
<!--Optional:-->
<invoiceData>
    <invoiceId>Invoice-id 1000</invoiceId>
    <invoiceDate>2013-12-02</invoiceDate>
    <invoiceDeliveryType>NONE</invoiceDeliveryType>
</invoiceData>
<!--Optional:-->
<storeId>?</storeId>
</sim:bookPayment>
</soapenv:Body>

```



Using Swish payment method?

Note that if you are using Swish as a payment method via Resurs, you must include the customer's cellphone number in the <cellPhone>-row.
Using <phone> will result in an error in production

bookSignedPayment

bookSignedPayment

Books a payment that has been signed.

Input (Literal)

Name	Type	Occurs	Nillable?	Description
paymentId	id	1..1	No	The ID for the payment

Output (Literal)

Name	Type	Occurs	Nillable?	Description
return	bookPaymentResult	1..1	No	The result of the payment booking.

Faults

Name	Content	Description
ECommerceErrorException	ECommerceError	Failed to book the payment. See error for details.

```
<soapenv:Body>
  <sim:bookSignedPayment>
    <paymentId>1</paymentId>
  </sim:bookSignedPayment>
</soapenv:Body>
```

getAnnuityFactors

getAnnuityFactors

Retrieves the annuity factors for a given payment method. The duration is given in months. While this makes most sense for payment methods that consist of part payments (i.e. new account), it is possible to use for all types. It returns a list of with one annuity factor per payment plan of the payment method. There are typically between three and six payment plans per payment method.

Input (Literal)

Name	Type	Occurs	Nillable?	Description
paymentMethodId	id	1..1	No	The identity of the payment method for which to retrieve the annuity factors. While this makes most sense for payment methods involving part payments, it is possible to use for all types. (See PaymentMethodType for more information about payment method types.)

Output (Literal)

Name	Type	Occurs	Nillable?	Description
return	annuityFactor	0..*	No	A list with one annuity factor per payment plan of the payment method. There are typically between three and six payment plans per payment method.

Faults

Name	Content	Description
EcommerceErrorException	EcommerceError	Failed to retrieve the annuity factors. See error for details.

 The data in our test environment does not always match the data you will be getting in our live environment. This operation can return different answers depending on your live setup.

 The annuity factor multiplied with the total cost of a product is the monthly cost using that payment plan.

Request example for payment method PARTPAYMENT

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:sim="http://ecommerce.resurs.com/v4/msg/simplifiedshopflow">
    <soapenv:Header/>
    <soapenv:Body>
        <sim:getAnnuityFactors>
            <paymentMethodId>PARTPAYMENT</paymentMethodId>
        </sim:getAnnuityFactors>
    </soapenv:Body>
</soapenv:Envelope>
```

Response example for payment method PARTPAYMENT

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns3:getAnnuityFactorsResponse xmlns:ns2="http://ecommerce.resurs.com/v4/msg/exception" xmlns:ns3="http://ecommerce.resurs.com/v4/msg/simplifiedshopflow">
      <return>
        <factor>0.333333333</factor>
        <duration>3</duration>
        <paymentPlanName>3 Månader - 0% ränta</paymentPlanName>
      </return>
      <return>
        <factor>0.166666667</factor>
        <duration>6</duration>
        <paymentPlanName>6 Månader - 0% ränta</paymentPlanName>
      </return>
      <return>
        <factor>0.083333333</factor>
        <duration>12</duration>
        <paymentPlanName>12 Månader - 0% ränta</paymentPlanName>
      </return>
      <return>
        <factor>0.041666667</factor>
        <duration>24</duration>
        <paymentPlanName>24 Månader - 0% ränta</paymentPlanName>
      </return>
      <return>
        <factor>0.0290571842</factor>
        <duration>48</duration>
        <paymentPlanName>48 Månader - 16,72% ränta</paymentPlanName>
      </return>
      <return>
        <factor>0.0230808901</factor>
        <duration>72</duration>
        <paymentPlanName>72 Månader - 17,96% ränta</paymentPlanName>
      </return>
    </ns3:getAnnuityFactorsResponse>
  </soap:Body>
</soap:Envelope>
```

getCostOfPurchaseHtml

getCostOfPurchaseHtml

Retrieves detailed cost of purchase information in HTML format. Resurs Bank is legally obliged to show this information everywhere its payment methods are marketed. This information can either be fetched with this method or linked. If linking is preferred, the links returned by the payment method ([getPaymentMethods](#)) are to be used. Returns a styleable HTML table containing the cost of purchase information.

Input (Literal)

Name	Type	Occurs	Nullable?	Description
paymentMethodId	id	1..1	No	The identity of the payment method for which to retrieve the detailed cost of purchase information.
amount	positiveDecimal	1..1	No	The amount on which to base the calculations.

Output (Literal)

Name	Type	Occurs	Nullable?	Description
return	String	1..1	No	A styleable HTML table containing the cost of purchase information.

Faults

Name	Content	Description
ECommerceErrorException	ECommerceError	Failed to retrieve the cost of purchase information. See error for details.

Example Code - Request & Response

getCostOfPurchaseHtml

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:sim="http://ecommerce.resurs.com/v4/msg/simplifiedshopflow">
  <soapenv:Header/>
  <soapenv:Body>
    <sim:getCostOfPurchaseHtml>
      <paymentMethodId>INVOICE</paymentMethodId>
      <amount>1000</amount>
    </sim:getCostOfPurchaseHtml>
  </soapenv:Body>
</soapenv:Envelope>
```



More about [legal requirements](#)



The data in our test environment does not always match the data you will be getting in our live environment. This operation can return different answers depending on your live setup.

getCostOfPurchaseHtmlResponse

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <n3:getCostOfPurchaseHtmlResponse xmlns:n3="http://ecommerce.resurs.com/v4/msg/simplifiedshopflow"
      xmlns:n2="http://ecommerce.resurs.com/v4/msg/exception">
```

```

<return><p>
<h3 class="headHeader"></h3><br/>
<span class="headText">Kreditkostnader och månadsbelopp - kredit (SEK): 3600.00<br/></span>
<table class="priceTable">
    <tr class="oddRow">
        <td class="header">Kreditbelopp (SEK)</td>
        <td class="content">3600</td>
        <td class="content">3600</td>
        <td class="content">3600</td>
        <td class="content">3600</td>
        <td class="content">3600</td>
        <td class="content">3600</td>
        <td class="content">3600</td>
    </tr>
    <tr class="evenRow">
        <td class="header">Betalningsalternativ</td>
        <td class="content">3 MÅN</td>
        <td class="content">6 MÅN</td>
        <td class="content">12 MÅN</td>
        <td class="content">24 MÅN</td>
        <td class="content">36 MÅN</td>
        <td class="content">48 MÅN</td>
        <td class="content">72 MÅN</td>
    </tr>
    <tr class="oddRow">
        <td class="header">Löptid (mån)</td>
        <td class="content">3</td>
        <td class="content">6</td>
        <td class="content">12</td>
        <td class="content">24</td>
        <td class="content">36</td>
        <td class="content">48</td>
        <td class="content">72</td>
    </tr>
    <tr class="evenRow">
        <td class="header">Kreditränta [rörlig årsränta] (%)</td>
        <td class="content">0,00</td>
        <td class="content">0,00</td>
        <td class="content">0,00</td>
        <td class="content">0,00</td>
        <td class="content">9,7</td>
        <td class="content">16,72</td>
        <td class="content">17,96</td>
    </tr>
    <tr class="oddRow">
        <td class="header">Administrations- avgift (SEK/månad)</td>
        <td class="content">29</td>
        <td class="content">29</td>
        <td class="content">29</td>
        <td class="content">29</td>
        <td class="content">29</td>
        <td class="content">29</td>
        <td class="content">29</td>
    </tr>
    <tr class="evenRow">
        <td class="header">Uppläggningssavgift* (SEK)</td>
        <td class="content">0,00</td>
        <td class="content">0,00</td>
        <td class="content">0,00</td>
        <td class="content">0,00</td>
        <td class="content">0,00</td>
        <td class="content">0,00</td>
        <td class="content">0,00</td>
    </tr>
    <tr class="oddRow">
        <td class="header">Ordinarie månadsbelopp** (SEK)</td>
        <td class="content">1229</td>
        <td class="content">629</td>
        <td class="content">329</td>
        <td class="content">179</td>
        <td class="content">129</td>
    </tr>

```

```
<td class="content">134</td>
<td class="content">112</td>
</tr>
</table>
<p>

<span class="tailText">
*Ev. Uppläggningsavgift betalas månaden efter val.<br/>
<br>
Exemplet förutsätter att räntan och avgifterna är oförändrade under hela kreditperioden. Andra sätt att utnyttja krediten kan leda till såväl en högre som lägre effektiv ränta. Den effektiva räntan är beräknad i enlighet med Konsumentverkets riktlinjer.<br>Enligt kontovillkoren kan det finnas krav på längsta inbetalningsbelopp, vilket kan medföra högre månadsbelopp än vad som annars skulle blivit fallet. I sådana fall förkortas i stället återbetalningstiden. RB kreditprövar ansökan enlig lag. Vid inhämtande av uppgifter från extern databas erhålls kopia av lämnade uppgifter från kreditupplysningsföretaget. RB förbehåller sig fri prövningsrätt.
</span>

<br/>
<p>
Standardiserad europeisk konsumentkreditinformation (SEKKI):
<a class="legalInfoLink" href="https://test.resurs.com/sekki-mock/sekki?bankProductId=LG686069&chainId=107&countryCode=SE&language=sv&storeId=107&amount=3600" target="_blank">Svenska (öppnas i nytt fönster)</a>
</p>
<p>
Allmänna villkor:
<a class="legalInfoLink" href="https://test.resurs.com/documenthandler/Dokument.pdf?customerType=natural&docType=commonTerms&land=SE&language=sv" target="_blank">Svenska (öppnas i nytt fönster)</a>
</p></return>
</ns3:getCostOfPurchaseHtmlResponse>
</soap:Body>
</soap:Envelope>
```

Widget for information on installment amounts.

If you want to display a "pay from xx per month" information on the product pages, a good idea is to create a widget for configuration.

In this widget the admin will choose a payment method that shall form the base for the installment information. The admin will then choose the installment plan (number of months) without interest for this payment method that gives the lowest amount. The annuity factor for this selected installment plan shall then be multiplied with each product's price to get the amount to present at the product pages.

The widget must:

1. Have the possibility to be switched on and off, if this shall be used in the shop or not.
2. Only show the payment methods available for part payment, that is only display methods with tag <specificType>PART_PAYMENT</specificType> or <specificType>REVOLVING_CREDIT</specificType> for selection.
3. Present the different number of months/installment plans for the payment method selected in step 2.
4. Have a configurable minimum price. The widget shall only be displayed for products which price, including VAT, is within the price range of this given minimum price and the amount from tag maxLimit from getPaymentMethods. This minimum price is to ensure that the monthly installment isn't below 150 Kr, which is the lower limit for which a bill is sent from Resurs.

The prices should be rounded up to the next even amount.

You can collect available annuity factors by the ECom library.

Cache as much as possible to avoid unnecessary calls to ECom.

getAddressByPhone (NO)

 This function is only available in Norway!

getAddressByPhone

Retrieves address information.

Input (Literal)

Name	Type	Occurs	Nillable?	Description
phoneNumber	string	1..1	No	The government identity of the customer for which to retrieve the address.
customerIpAddress	string	0..1	Yes	The IP address from which the customer has accessed the service. To prevent bashing.

Output(Literal)

Name	Type	Occurs	Nillable?	Description
return	address	1..1	No	If a match could be made, the customer address.

Faults

Name	Content	Description
ECommerceErrorException	ECommerceError	Failed to retrieve the address information. See error for details.

Introduction

This function is offered as a service, since most webshops collect address information early in the checkout flow. This function fetches address information. If we get a match on the phone number, its registered address is returned. This address may be used as billing- and delivery address. The customerIpAddress is to prevent address harvesting. We allow only a limited number of queries within a time frame. Bashing will result in an [error](#) when the threshold is exceeded.

Code Examples - Request & Response

getAddressByPhone

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:sim="http://ecommerce.resurs.com/v4/msg/simplifiedshopflow">
  <soapenv:Header/>
  <soapenv:Body>
    <sim:getAddressByPhone>
      <phoneNumber>40000001</phoneNumber>
      <customerIpAddress>80.80.80.80</customerIpAddress>
    </sim:getAddressByPhone>
  </soapenv:Body>
</soapenv:Envelope>
```

If the customer exists, its registered address is returned. If the customer is of type LEGAL (an organization/comapny) `firstName` and `lastName` will be null

getAddressResponse

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns3:getAddressResponse xmlns:ns3="http://ecommerce.resurs.com/v4/msg/simplifiedshopflow" xmlns:ns2="http://ecommerce.resurs.com/v4/msg/exception">
      <return>
        <fullName>Bjarne Bock Krohg</fullName>
        <firstName>Bjarne</firstName>
        <lastName>Krohg</lastName>
        <addressRow1>Prinsens Gate 14</addressRow1>
        <postalArea>Bergen</postalArea>
        <postalCode>5014</postalCode>
        <country>NO</country>
      </return>
    </ns3:getAddressResponse>
  </soap:Body>
</soap:Envelope>
```

Merchant API 2.0

Swagger-documentation for Merchant API

Output in accounting file

Api	payment_id (set by Resurs)	orderReference	transactionId	accounting file
Create Payment	8b9dab18-e1fe-430e-891b-42edb89ba77a	ORDER001		
Capture			transactionId-DEBIT001	transactionId-DEBIT001
Capture			transactionId-DEBIT002	transactionId-DEBIT002
Refund			transactionId-CREDIT001	transactionId-CREDIT001
<hr/>				
Create Payment	8b9dab18-e1fe-430e-891b-42edb89ba77a	ORDER001		
Capture			N/A	ORDER001
Capture			N/A	ORDER001
Refund			N/A	ORDER001
<hr/>				
Create Payment	8b9dab18-e1fe-430e-891b-42edb89ba77a	N/A		
Capture			N/A	8b9dab18-e1fe-430e-891b-42edb89ba77a
Capture			N/A	8b9dab18-e1fe-430e-891b-42edb89ba77a
Refund			N/A	8b9dab18-e1fe-430e-891b-42edb89ba77a

Broker Application Flow Denmark

Basic API flow

Authentication

Every request requires an authorization header with a Bearer-token. A token lasts for 3600 seconds (1 hour). To get a token you may use your test-credentials received from Resurs Bank.

URL to get token: <https://merchant-api.integration.resurs.com/oauth2/token>

- Client ID
- Client Secret
- Scope= mock-merchant-api

Link to the call in swagger documentation: [Get Token](#)

Curl to get token

```
curl --location --request POST 'https://merchant-api.integration.resurs.com/oauth2/token' \
--header 'accept: application/json' \
--header 'Content-Type: application/x-www-form-urlencoded' \
--data-urlencode 'client_id=fill out client_id' \
--data-urlencode 'client_secret=fill out client_secret' \
--data-urlencode 'scope=mock-merchant-api' \
--data-urlencode 'grant_type=client_credentials'
```

Get Store ID

Get available stores

A client may have access to multiple stores, therefore we need to know which store to make the application or payment for. This can be done by getting the available stores. Each store has a store-id. This id will be used in the next step to specify for which store we would like to get the payment methods.

URL to get available stores: <https://merchant-api.integration.resurs.com/v2/stores>

Link to the call in swagger documentation: [Get Stores](#)

Curl to get available stores

```
curl --location --request GET 'https://merchant-api.integration.resurs.com/v2/stores' \
--header 'Authorization: Bearer <TOKEN>'
```

Get available payment methods

A store may have multiple payment methods available. The list of available payment methods will show what payment methods there are to apply from at the chosen store.

Each payment method has a paymentmethod id, which will be used in the next step, when the application is created.

When presenting the payment methods to the customer, please note that by law you are required to display our terms/link to our terms regarding the credit payment methods. The link is found in this same request with "type: PRICE_INFO"

URL to get available payment methods: https://merchant-api.integration.resurs.com/v2/stores/{store_id}/payment_methods

Link to the call in swagger documentation: [Get Payment Methods](#)

Curl to get available payment methods

```
curl --location --request POST 'https://merchant-api.integration.resurs.com/v2/stores/{store\_id}/payment\_methods' \
--header 'Content-Type: application/json' \
--header 'Authorization: Bearer <TOKEN>'
```

Get payment specification

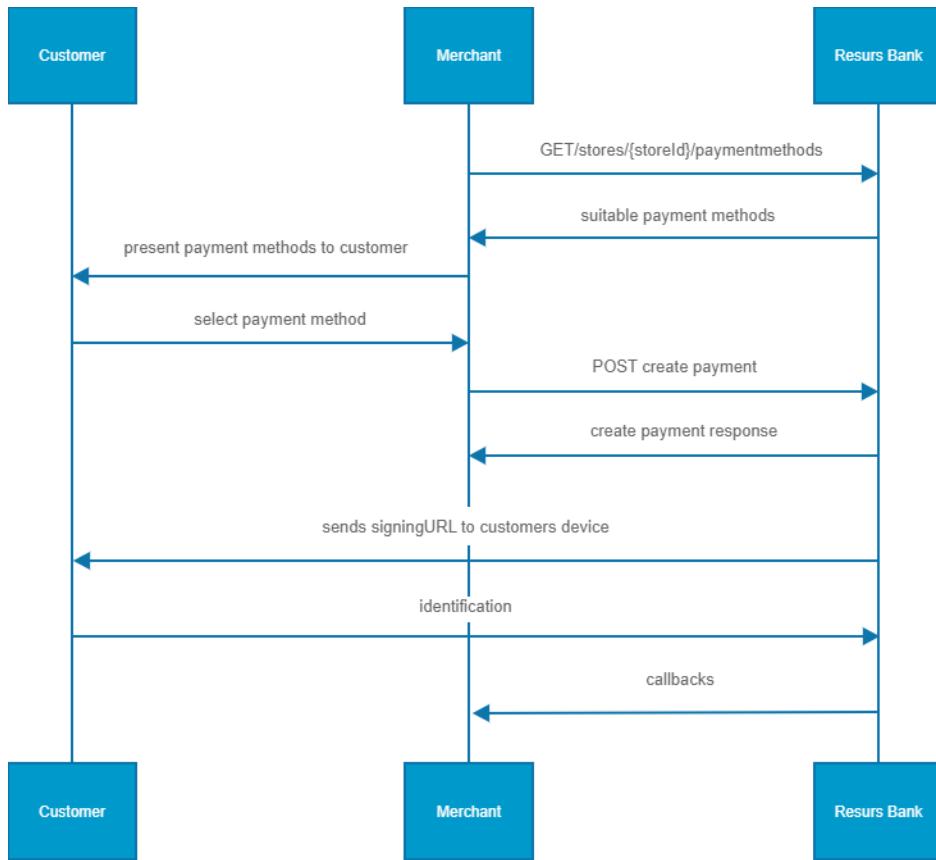
To get an overview of what the API

Merchant API E-Commerce



Generic, explanatory documentation

Please note that the flow and its requests below is an example and other flows or variations may exist. Please advice with onboarding@resurs.se regarding specific actions prior to mapping up any logic or beginning any coding.



What can I find here?

- Basic API flow
 - Authentication
 - Get Store ID
 - Get available payment methods
- Create Payment
 - Create payment
 - Create payment-responses and what to do next
- Postman-collection of the requests above

Basic API flow

Authentication

Every request requires an authorization header with a Bearer-token. A token lasts for 3599 seconds. To get a token you may use your test-credentials received from Resurs Bank.

URL to get token: <https://merchant-api.integration.resurs.com/oauth2/token>

- Client ID
- Client Secret
- Scope= merchant-api

Link to the call in swagger documentation: [Get Token](#)

Curl to get token

```
curl --location --request POST 'https://merchant-api.integration.resurs.com/oauth2/token'  
--header 'accept: application/json' \  
--header 'Content-Type: application/x-www-form-urlencoded' \  
--data-urlencode 'client_id=fill out client_id' \  
--data-urlencode 'client_secret=fill out client_secret' \  
--data-urlencode 'scope=merchant-api' \  
--data-urlencode 'grant_type=client_credentials'
```

Get Store ID

Get available stores

A client may have access to multiple stores, therefore we need to know which store to make the application or payment for. This can be done by getting the available stores. Each store has a store-id. This id will be used in the next step to specify for which store we would like to get the payment methods.

URL to get available stores: <https://merchant-api.integration.resurs.com/v2/stores>

Link to the call in swagger documentation: [Get Stores](#)

Curl to get available stores

```
curl --location --request GET 'https://merchant-api.integration.resurs.com/v2/stores'  
--header 'Authorization: Bearer <TOKEN>'
```

Get available payment methods

A store may have multiple payment methods available. The list of available payment methods will show what payment methods there are to use from at the chosen store. Do not forget to sort out the payment methods which min-/max purchase limit are outside the cart value. If you input an amount in the request, the API will automatically not show the unavailable payment methods for you - you do not need to sort out payment methods yourselves. Each payment method has a paymentmethod id, which will be used in the next step, when the payment is created. When presenting the payment methods to the customer, please note that by law you are required to display our terms/link to our terms regarding the credit payment methods. The link is found in this same request with "type: PRICE_INFO"

URL to get available payment methods: https://merchant-api.integration.resurs.com/v2/stores/{store_id}/payment_methods

Link to the call in swagger documentation: [Get Payment Methods](#)

Curl to get available payment methods

```
curl --location --request POST 'https://merchant-api.integration.resurs.com/v2/stores/{store\_id}/payment\_methods'  
--header 'Content-Type: application/json' \  
--header 'Authorization: Bearer <TOKEN>'
```

Create Payment

Create payment

The call below is used to create a new payment and authorize the payment.

- orderReference = your internal order reference, if it's not provided by you it will be set by Resurs Bank. Will be provided in accounting file if not another value is provided in capture-request later on.
- initiatedOnCustomersDevice = true because the payment in the webshop's checkout is initiated on the customers device

- handleFrozenPayments = payments may be frozen for a short time due to fraud controls. When a payment is frozen the customer is informed that the purchase is ready, but the goods shall not be shipped until the payment is captured

For further information regarding initiatedOnCustomersDevice and handleFrozenPayments, please go to "Schema" "CreatePaymentRequest" "Options" ; [Create Payment Options](#)

URL to create payment: <https://merchant-api.integration.resurs.com/v2/payments>

Link to the call in swagger documentation: [Create Payment](#)

Curl to create payment

```
curl --location --request POST 'https://merchant-api.integration.resurs.com/v2/payments' \
--header 'Authorization: Bearer <TOKEN>' \
--header 'Content-Type: application/json' \
--data-raw

{
  "storeId": "{{store_id}}",
  "paymentMethodId": "{{payment_method_id}}",
  "order": {
    "orderLines": [
      {
        "description": "Book",
        "quantity": 1,
        "quantityUnit": "pcs",
        "vatRate": 25,
        "totalAmountIncludingVat": 500.0
      }
    ],
    "orderReference": "orderref-12345"
  },
  "customer": {
    "customerType": "NATURAL",
    "governmentId": "xxxxxx",
    "email": "xxx@yyy.com",
    "mobilePhone": "xxxxxx"
  },
  "options": {
    "initiatedOnCustomersDevice": true,
    "handleFrozenPayments": true,
    "callbacks": {
      "authorization": {
        "url": "{{callback_url_authorization}}"
      },
      "management": {
        "url": "{{callback_url_management}}"
      }
    },
    "redirectionUrls": {
      "customer": {
        "failUrl": "{{fail_url_customer}}",
        "successUrl": "{{success_url_customer}}"
      }
    },
    "timeToLiveInMinutes": 120
  }
}
```

Create payment-responses and what to do next

status: TASK_REDIRECT_REQUIRED redirect customer to "customerUrl" and await callback. Callbacks with callback-status AUTHORIZED, REJECTED, FROZEN or CAPTURED can be received before or after redirection is performed.

Postman-collection of the requests above



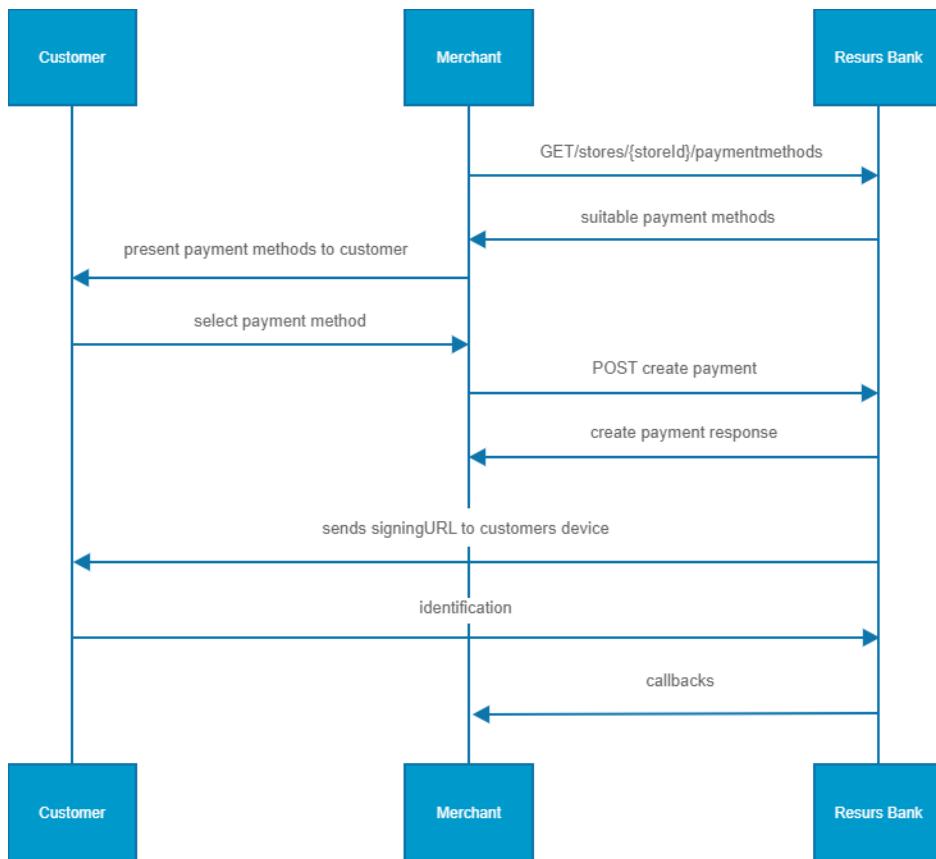
Merchant API2 Ecommerce.json

Merchant API POS



Generic, explanatory documentation

Please note that the flow and its requests below is an example and other flows or variations may exist. Please advice with onboarding@resurs.se regarding specific actions prior to mapping up any logic or beginning any coding.



What can I find here?

- [Basic API flow](#)
 - [Authentication](#)
 - [Get Store ID](#)
 - [Get available payment methods](#)
- [Create Payment](#)
 - [Create payment](#)
 - [Create payment-responses and what to do next](#)
- [Postman-collection of the requests above](#)

Basic API flow

Authentication

Every request requires an authorization header with a Bearer-token. A token lasts for 3599 seconds. To get a token you may use your test-credentials received from Resurs Bank.

URL to get token: <https://merchant-api.integration.resurs.com/oauth2/token>

- Client ID
- Client Secret
- Scope= mock-merchant-api

Link to the call in swagger documentation: [Get Token](#)

Curl to get token

```
curl --location --request POST 'https://merchant-api.integration.resurs.com/oauth2/token'  
--header 'accept: application/json' \  
--header 'Content-Type: application/x-www-form-urlencoded' \  
--data-urlencode 'client_id=fill out client_id' \  
--data-urlencode 'client_secret=fill out client_secret' \  
--data-urlencode 'scope=mock-merchant-api' \  
--data-urlencode 'grant_type=client_credentials'
```

Get Store ID

Get available stores

A client may have access to multiple stores, therefore we need to know which store to make the application or payment for. This can be done by getting the available stores. Each store has a store-id. This id will be used in the next step to specify for which store we would like to get the payment methods.

URL to get available stores: <https://merchant-api.integration.resurs.com/v2/stores>

Link to the call in swagger documentation: [Get Stores](#)

Curl to get available stores

```
curl --location --request GET 'https://merchant-api.integration.resurs.com/v2/stores'  
--header 'Authorization: Bearer <TOKEN>'
```

Get available payment methods

A store may have multiple payment methods available. The list of available payment methods will show what payment methods there are to apply from at the chosen store.

Each payment method has a paymentmethod id, which will be used in the next step, when the application is created.

URL to get available payment methods: https://merchant-api.integration.resurs.com/v2/stores/{store_id}/payment_methods

Link to the call in swagger documentation: [Get Payment Methods](#)

Curl to get available payment methods

```
curl --location --request POST 'https://merchant-api.integration.resurs.com/v2/stores/{store\_id}/payment\_methods'  
--header 'Content-Type: application/json' \  
--header 'Authorization: Bearer <TOKEN>'
```

Create Payment

Create payment

The body below is an example used to create a new payment and authorize the payment. Note that more fields exist and options can vary depending on user-case. See the Swagger-documentation linked below for more information.

- orderReference = your internal order reference, if it's not provided by you it will be set by Resurs Bank
- initiatedOnCustomersDevice = if clerk has asked for, investigated and confirmed that customer ID is OK, this setting can be set as *false*. If not, or you want Resurs to be responsible for ID'ing the customer, set this option as *true*.
- deliverLinks = instructs whether or not to deliver links to the customer in case customer interaction like consents or contract signing is required
- handleFrozenPayments = payments may be frozen for a short time due to fraud controls. When a payment is frozen the goods shall not be delivered to the customer until the payment is captured
- handlemanualinspection = if handling manual inspection, the flow may be stopped and the customer must talk to Resurs Bank personnel before the credit can be approved

For further information about orderReference, InitiatedOnCustomersDevice and handleFrozenPayments, please go to "Schema" "CreatePaymentRequest" "Options" [Create Payment Options](#)

URL to create payment: <https://merchant-api.integration.resurs.com/v2/payments>

Link to the call in swagger documentation: [Create Payments](#)

Curl to create payment

```
curl --location --request POST 'https://merchant-api.integration.resurs.com/v2/payments' \
--header 'Authorization: Bearer <TOKEN>' \
--header 'Content-Type: application/json' \
--data-raw

{
  "storeId": "{{store_id}}",
  "paymentMethodId": "{{payment_method_id}}",
  "order": {
    "orderLines": [
      {
        "description": "Book",
        "quantity": 1,
        "quantityUnit": "pcs",
        "vatRate": 25,
        "totalAmountIncludingVat": 500.0
      }
    ],
    "orderReference": "{{orderref}}"
  },
  "customer": {
    "customerType": "NATURAL",
    "governmentId": "xxxxxx",
    "email": "test@resurs.com",
    "mobilePhone": "xxxxxx"
  },
  "options": {
    "initiatedOnCustomersDevice": true,
    "deliverLinks": true,
    "handleManualInspection": false,
    "automaticCapture": true,
    "callbacks": {
      "authorization": {
        "url": "{{callback_url_authorization}}"
      },
      "management": {
        "url": "{{callback_url_management}}"
      }
    },
    "redirectionUrls": {
      "customer": {
        "failUrl": "{{fail_url_customer}}",
        "successUrl": "{{success_url_customer}}"
      }
    },
    "timeToLiveInMinutes": 120
  }
}
```

Create payment-responses and what to do next

TASK_REDIRECT_REQUIRED The customer is to be redirected to "customerUrl" either sent by you or by Resurs (depending whether options deliverLinks is true/false). Callbacks with callback-status AUTHORIZED, REJECTED, FROZEN or CAPTURED can be received before or after redirection is performed.

If you want to check if the payment has been *Authorized* without redirecting (it may be that successUrl is shown directly in the customerUrl and an authentication is not needed), then call GET /v2/payments/{payment_id} and check for *status: ACCEPTED*

If paper signing is to be used (For Finland only) for signing an application/payment, see [Physical Agreement Finland](#)

Postman-collection of the requests above



Merchant API2 POS.json

Payment Management

Callback Status	Possible Actions
AUTHORIZED	CAPTURE CANCEL
CAPTURED	REFUND
FROZEN	await for callback-status AUTHORIZED or REJECTED
What can I find here?	
<ul style="list-style-type: none">• Capture payment• Refund payment• Cancel payment• Postman-collection of the requests above	

Capture payment

This call is used to capture a payment after the payment is created.
If no order lines are supplied, then all not yet captured order lines will be captured.

URL to capture payment: https://merchant-api.integration.resurs.com/v2/payments/{payment_id}/capture

Link to the call in swagger documentation: [Capture Payment](#)

Curl to capture the payment

```
curl --location --request POST 'https://merchant-api.integration.resurs.com/v2/payments/{payment\_id}/capture' \
--header 'Content-Type: application/json' \
--header 'Authorization: Bearer <TOKEN>' \
--data-raw ' \
{ \
    "creator": "fictive@resurs.com", \
    "orderLines": [ \
        { \
            "description": "Book", \
            "quantity": 1, \
            "quantityUnit": "pcs", \
            "vatRate": 25, \
            "totalAmountIncludingVat": 500.0 \
        } \
    ] \
}
```

Refund payment

This call is used to refund a payment that is captured.
If no order lines are supplied, then all captured not yet refunded order lines will be refunded.

URL to refund payment: https://merchant-api.integration.resurs.com/v2/payments/{payment_id}/refund

Link to the call in swagger documentation: [Refund Payment](#)

Curl to refund the payment

```
curl --location --request POST 'https://merchant-api.integration.resurs.com/v2/payments/{payment\_id}/refund' \
--header 'Content-Type: application/json' \
--header 'Authorization: Bearer <TOKEN>' \
--data-raw ' \
{ \
    "creator": "fictive@resurs.com", \
    "orderLines": [ \
        { \
            "description": "Book", \

```

```
        "quantity": 1,
        "quantityUnit": "pcs",
        "vatRate": 25,
        "totalAmountIncludingVat": 500.0
    ]
}
```

Cancel payment

This call is used to cancel a payment that is created but not yet captured.
If no order lines are supplied, everything that can be canceled will be canceled.

URL to cancel payment: https://merchant-api.integration.resurs.com/v2/payments/{payment_id}/cancel

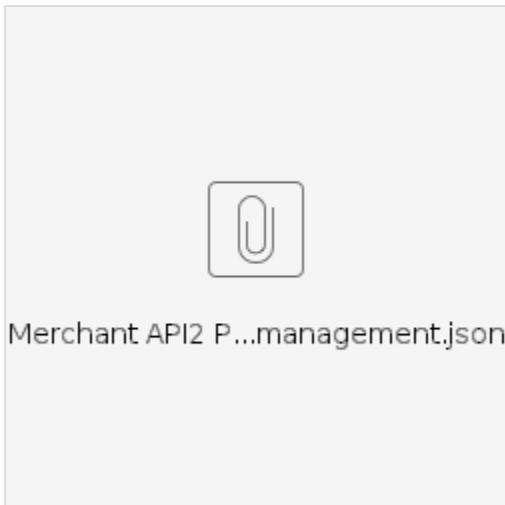
Link to the call in swagger documentation: [Cancel Payment](#)

Curl to cancel the payment

```
curl --location --request POST 'https://merchant-api.integration.resurs.com/v2/payments/{payment\_id}/cancel'
--header 'Content-Type: application/json' \
--header 'Authorization: Bearer <TOKEN>' \
--data-raw '

{
  "creator": "fictive@resurs.com",
  "orderLines": [
    {
      "description": "Book",
      "quantity": 1,
      "quantityUnit": "pcs",
      "vatRate": 25,
      "totalAmountIncludingVat": 500.0
    }
  ]
}
```

Postman-collection of the requests above



Physical Agreement Finland



Example

Please note that the flow is only applicable for partners and customers in Finland and can only be set up in agreement with Key Account Manager

What can I find here?

- Create an agreement
- Sign an agreement

Create an agreement

In order to print a physical agreement the agreement must first be created.

URL to create physical agreement: <https://merchant-api.integration.resurs.com/v2/agreements>

Link to the call in swagger documentation: [Create physical agreement](#)

Curl to create agreement

```
curl --location --request POST 'https://merchant-api.integration.resurs.com/v2/agreements' \
--header 'Authorization: Bearer Token' \
--header 'Content-Type: application/json' \
--data-raw '{
    "applicationId": "",
    "applicantIdentification": {
        "type": "ID",
        "reference": "reference"
    }
}'
```

Sign an agreement

After creating an agreement, the agreement must be signed and printed.

URL to sign physical agreement: <https://merchant-api.integration.resurs.com/v2/agreements/sign>

Link to the call in swagger documentation: [Sign Physical Agreement](#)

Curl to sign agreement

```
curl --location --request POST 'https://merchant-api.integration.resurs.com/v2/agreements/sign' \
--header 'Authorization: Bearer Token' \
--header 'Content-Type: application/json' \
--data-raw '{
    "paymentId": "",
    "applicantIdentification": {
        "type": "ID",
        "reference": "reference"
    }
}'
```

Platform Plugins

Content of this page

- Supported open source e-commerce website platforms
- E-Commerce Platforms
 - SSL Certificate, https and certificates
 - Do not forget to check your certificates! See FAQ for more information.
 - Maintaining Compatibility
 - PHP Platform Requirements
 - Avoid PHP 5.6, try push your PHP platform upwards - not downwards.
 - Autotesting
- Issue tracking
- Magento modules
- OpenCart
- PrestaShop Payment Gateways
- WooCommerce

Supported open source e-commerce website platforms

E-commerce Platforms			
Shop flow	Magento v2.3.x-v2.4.x	WooCommerce Read more	OpenCart v1.5.x - v3.x
	 2		
			End of life: September, 2023
			
Resources	✓ Not Denmark	✗	✓ Not Denmark
Simplified Flow	✓	✗	✗
End of life: Q1, 2024			
Resources	✗	✓	✗

Merchant API 2.0				
After shop				
Deleting whole order	✓	✓	✗	
Deleting part order	✗	✗	✗	
Creating whole order	✓	✓	✗	
Creating part order	✓	✓*	✗	
Annulment whole order	✓	✓	✗	
Annulment part order	✗	✓*	✗	
Additional Debit of Payment	✗	✗	✗	
Callback support				
Callbacks report to shop	✓	✓	✓	
Discounts and fees				
Handle gift card	✗	✗	✓	
Handle invoice fee	✗	✓	✗	

Handle discount	✓	✓	✓
Handle shipping fee	✓	✓	✓
Other			
Display monthly cost with Resources Bank in product catalog	✓	✓	✓
Use platform order number as reference in Resources Bank	✓	✓	✓
Supports multistore	✓	✗	✓

SSL Certificate, https and certificates

Do not forget to check your certificates! See [FAQ](#) for more information.

Maintaining Compatibility

The more plugins you activate that handles your layout - and changes the default behavior of regular styling - the more the risk that the plugin won't work as intended.

PHP Platform Requirements

Avoid PHP 5.6, try push your PHP platform upwards - not downwards.

- We officially only support PHP versions which are actively maintained. Support for legacy versions is not guaranteed.
- JSON / ext-json
- Soap / ext-xml

Autotesting

All tests related to PHP releases older than 7.3 has been dropped. Our [Pipelines](#) runs with 7.3, 7.4 and 8.0. [Bamboo](#) dropped all tests, except for 7.3 for several reasons.

We have dropped all our testings for older PHP releases. only runs with PHP 7.3, 7.4 and 8.0 - as an extra test suite, only runs on 7.3. Since the market changes rapidly we realized that we had to drop all old tests, as the started to break while we were going forward.

Issue tracking

Have a look at the landing page for [Development](#).



The plugins listed and provided from this page are open source and community-driven software. By means, it is software that *could* be used by merchants for their ecommerce.

The community solution opens up for developers to make special adaptations, forks and features. You don't have to use them and many developers choosed to write their own integrations. It is your responsibility to test and make control that the plugins works properly within your platform. If it doesn't work, feel free to contact us via onboarding@resurs.se. **Do not** publish our plugins straight into a production server, before those steps. The plugins on this page are written generically and supposed to work with many solutions. This said, it is not guaranteed to fit or work with all solutions.

Also, if you feel you want to modify the plugins and share it, fork the projects from Bitbucket and create your pull request for us.

Magento modules

Table of contents

- [Plugin versions](#)
 - [Flags](#)
 - **END OF LIFE**
- [Deprecation notice*](#)
- [Requirements and support](#)
 - [Magento 2.0](#)
- [Known problems](#)
 - [Slow BOOKED callback](#)

Modules in this section is used for integrating Resurs Bank payment solutions with Magento based e-commerce stores.

Plugin versions

Flow/Name	Magento Version	PHP	Details	Country	Documentation	Links and downloadables	Support
Resurs Checkout Web - Magento 1	CE 1.9.x	>= 5.6	Resurs Checkout Docs	SE, FI, NO	ResursCheckout/Magento 1.7-1.9	Bitbucket	END OF LIFE
Resurs Checkout Web - Magento 2	CE 2.2-2.3	>= 5.5.22	Resurs Checkout Docs	SE, FI, NO	ResursCheckout/Magento 2.x	Bitbucket	END OF LIFE
Resurs Checkout Web - Magento 2	CE 2.4+	>= 8.1	Resurs Checkout Docs	SE, FI, NO	ResursCheckout/Magento 2.4+ RB/Magento 2.4+ Installation Instruction	Marketplace	SUPPORTED
Resurs Hosted Flow Plugin	CE 1.7-1.9	>= 5.4		DK		Bitbucket	END OF LIFE

Flags

END OF LIFE

The EOL flag above is a way for us to mark how our support works and when there is an **End Of Life** for each plugin.

For our Resurs OldFlow Plugin, there is an end of life that have been passed, where no new development are initiated. In the case of OldFlow plugin, it has also been taking out of maintenance. No patches has been released [since february 2016](#) as many of the functions in the last release has been replaced by other plugins (see the list above). Most of the plugins are still technically supported (i.e. by the flow), but patching the is something we avoid.

Deprecation notice*

As Magento development flies forward in time, we're about to leave especially the 1.7-universe. Our new modules has a target version of 1.9 - 2.x, so running on older versions might break something. Magento 1.7 is the oldest version that we recently supported. The new releases of our modules is not guaranteed to work properly. Our "Resurs Checkout"-module MIGHT work for 1.7, but it has not been fully tested.

Requirements and support

- For supported PHP platforms, see above.
- Merchant credentials from Resurs Bank (ID and Password)
- For Magento, *at least* community edition (the commercial releases are not officially supported)

Magento 2.0

[Resurs Bank Installation Instructions](#)

Known problems

Slow BOOKED callback

The problem might be caused by rendered templates. Depending on the template some responses could be very slow and cause timeouts. Full description is located at the [FAQ](#).

Magento Plugin - OldFlow version

The "OldFlow version" is a plugin for Magento CE 1.7-1.9.2 that works with Resurs Bank deprecated flow. This page covers information about that plugin, features and installation instructions.

Checkout Compatibility

- Standard checkout (This is what we call "Vanilla")
- [Templates master](#) Fire Checkout
- [One step checkout](#)

Features

- Merchant can configure dynamic Resurs Bank payment methods from Backend
- Extension will display dynamic Resurs Bank payment methods pulled from Resurs Bank Server.
- Extension will fetch customer address via API and update their cart billing address if a customer is from Sweden.
- Extension is compatible with the standard Magento Checkout, [Firecheckout](#) third party extension and [OneStepCheckout](#) third party extension.
- Merchant can configure additional "Payment Fee" to each payment method.
- The Invoice Fee rate can be applied to all the payment methods.
- Customized Payment description can be added to each payment method by Merchant from back end
- This extension supports five languages – English, Danish, Finnish, Norwegian and Swedish
- Bundled product support
- Multiple webshops (scopes)
- Flow for finalize, annul, credit from Magento instead of Betalningsadmin/Payment admin (Read more about this at [Magento: Finalize-, credit- and annulpayment details \(v1.3.8.1\)](#))

Current flow

Magento uses Resurs' [deprecated flow](#), be sure to understand this. Also read about the [concepts and domain](#), order status FROZEN and its handling are important in the fraud and order concept.

Installation (Package Content)

Installation notes, error logging and development

Important installation and development notices has moved to a separate page:

Have a look at "[Release- and installation notes, error logging and development for Magento OldFlow](#)", since many problems that may occur due to Magento internal routines, configuration, bugs, etc, can be solved by reading there.

The ZIP archive package will have the content below:

The extension archive file named **Resursbank-X.X.X.tgz** and patches are located in the patch-path.

This can be installed directly using Magento Connect option in back end (which is recommended since we do not support manual installations). However, the plugin does also work, even if the plugin has been installed manually.

After installation, give write permissions to this Resurs Bank Payment extension directory:
/patch directory

Patch checkout files

All the necessary steps have been taken in order to bring all functionalities without core modification but using overriding features into the extension. However several of the items were not able to implement via override as they were from third party extensions. They must be patched.

Back up the original files of those extensions (OneStepCheckout and FireCheckout) and then choose one of the checkouts below to see how to patch them. Due to licensing, we will not distribute full content of the files, just the code for the patch.

1. OneStepCheckout: See [Manually patching Onestep checkout](#)
2. FireCheckout: See [Manually patching Firecheckout](#)

Magentoconnect MANAGER

Extensions Settings

[Return to Admin](#) [Log Out](#)

Settings

Put store on the maintenance mode while installing/upgrading/backup creation

Create Backup

Install New Extensions

1 Search for modules via [Magento Connect](#).

2 Paste extension key to install:

Direct package file upload

1 Download or build package file.

2 Upload package file: No file chosen

Manage Existing Extensions

Channel: Magento Community Edition

Clear all sessions after successfull install or upgrade:

Package Name	Installed	Actions	Summary
Resursbank	1.2.1 (beta)	<input type="button" value=""/>	The Resurs Bank Payment Gateway Magento Extension

This extension can be seamlessly installed (and should) as a Magento extension through Magento Connect (recommended). It can also be installed manually but we do not officially support this, since you are responsible yourself for setting up all file permissions correctly, etc

- Extract the extension archive file under the root directory,
- Click overwrite option to yes,
- Now the extension has been installed successfully.
- After manual installation, the first time the website may take some time for loading of the extension as it needs to set up database tables and all other default configurations.
- After installation, provide proper write permissions to this Resurs Bank Extension directory.
- Then clear Magento and Browser Cache once from back end.

Upgrade

Since no files can be overwritten, these steps has to be done on upgrading/reinstallation of the plugin, if using Magento Connect Manager. See Package Content above for other details about patches

- Uninstall Resurs Bank plugin
- Check if there are any files left in app/code/local/Resursbank (this path must be removed, and usually is if the uninstall works properly all the way)
- Now, upload the plugin through "Magento Connect Manager"

Settings and Configurations

The following are the admin settings in the Resurs Bank Magento Extension that are required to configure by Merchant/ Admin.

Those settings can be accessed from Back end System -> Configuration -> Payment Methods

Setting	Description
Section - Common settings	
Server	Select the payment environment whether "Test" or "Live" mode. Server URLs are dynamically configured with the two form fields for production and test.
BaseUrl Live/Test	Base URLs for our production- and test environment
Activated countries	Select the country to which this payment method should be available
Payment fee tax class	The fee rates that should be applicable for payment method fee
New order status	Select the new order status that should be applicable for all new orders through this Payment gateway
Status for orders that will not be accepted	Select the not accepted order status that should be applicable for all Not accepted or failed orders through this Payment gateway
Freeze status	The status for an order that is frozen.
Unfreeze/thaw status	Status that an order will be set to, when it's granted
Annulment order status	Order status to use on annulments (Normally "Canceled" could be used here)
Pending payment status	Select the pending payment status that should be applicable for all Pending orders through this Payment gateway
Signing status	A status for an order that are waiting for signing (normally used for credit cards). If this status is the last in order view, it will indicate that the customer has not fully completed an order.
Redirect location in Magento when signing fails	When signing fails a redirect call is being made. Normally the customer will be thrown back to the shopping cart, but sometimes merchants prefer to send the customer back to the payment-method section
Custom error text message	A custom error message when errors occur during order failures or cancellations
API	
How to handle paymentspecs <i>(Experimental setting! Only change this if you know what you are doing)</i>	This configuration value sets how the plugin should behave when sending information about the order to paymentadmin (speclines). <ul style="list-style-type: none"> The normal behaviour (Regular flow) is to send each product, as is. This means that the plugin will not check bundled packages and its content, and the primary article is sent as specline. The second behaviour (Adding products from bundles into speclines) will analyze bundled packages and send each article in a bundle through the flow Warning: Finalize/creditpayment/annulpayment is not supported in this mode)
Automatic finalizations, credits and annulments	Activates finalizing of orders. Default is disabled. Finalizing is normally made from backend, but when it comes to orders containing items that cannot ship from Magento, finalizing will occur immediately when the customer chooses to place an order (and only if the order contains non-shippable items).
Invoice numbering on debits, credits and annulments	The only option for the moment is <i>Let Resurs Bank handle invoice numbers</i> , meaning that finalizing an order will use Resurs Bank invoice ID's rather than Magentos internal.
Store Credentials	
Customer / Representative ID	The Resurs Bank Merchant ID gotten from Resurs Bank
Customer / Representative Password	Resurs Bank Merchant Password
Payment method settings (<i>The following are the settings that will be listed for each payment method that are available for this Merchant</i>)	
Payment method status	Enable or Disable the payment method
Title	Payment title that should be displayed for customer in checkout page
Sort order	Payment method display order in the front end.
Payment method logotype	Defines what logotype the payment method should have in a checkout. If no URI is given here, a default logotype will be used

Payment fee	Payment fee to each method can be set here for this specific Payment method.
Payment fee description	This value will be sent to Resurs Bank API Payment fee product line.
Payment method description	This value will be displayed in the front end under each payment method.

Updating payment methods

Updating payment methods in a multi-channel store (a store with multiple shops) should always be done in the top level of the store (using "website name"), since settings are inherited through the scopes. To completely save payment methods, you also have to save settings after updating them. Since version 1.2.8, salt keys for the callbacks are updated each time an update or a save occurs.



Updating the base URL and callbacks

If you for some reason change the base URL you must reload and save the payment methods again, this is so the plugin will re-register the callbacks with the new URL.

Magento Plugin 1.x - Changelog

Changelog for Resurs Bank Magento plugin

Deprecated versions can be [viewed here](#)

1.4.1 (Unreleased)

Internal issue ID	Description of what's fixed	Magento Version (Default: All)
57220	Old forms when switching between payment methods are cleaned up as an extra precaution	
55645	List of payment methods are now included in the "Save"-action (admin) instead of page reloading	
46563	Logotype images are replaced with new	
58206	Canceling payments sometimes interferes with other payment methods	
62183	Finalization with or without type INVOICE (has hotfix)	
60194	When submitLimitApplication returns unexpected line breaks in address data, the JS fails	

1.4.0 (Unreleased)

Internal issue ID	Description of what's fixed	Magento versions (Default all)
44045	Streamline release (hiding fields that are not necessary, and should be inherited from Magento's own form fields)	
46599	Information about changing the invoice sequence numbers added	
44397	Added row in admin that shows payment method thresholds (min - max)	
49811	Payment method thresholds are not handled properly in checkout (exceeding min or max will hide method)	
47333	getCostPopup in standardcheckout misbehaved and did not pop up properly. Fixed.	
47492	Checkout problems for logged in users	
48000	Too long article numbers shortened down to the database limit of max 100 characters, if length exceeded	
48025	Fixed minor cosmetic defect for Magento 1.9	1.9
49407	LEGAL address-forcing is not handled properly in confirmation mail (firecheckout patch included)	
49782	Magento 1.9 still using a deprecated getCostPopup method	1.9
55041	Conflicting jQuery, updated framework to comply with 1.9.1 and 1.9.2 requirements Also #54763	1.9.1
55290	Dynamic forms, issues with phone elements fixed	
50564	Fix for undefined indexes in Block/Checkout (Only visible in development environments)	
50036	Environment variables RESURS_CLEAN_METHODS_CACHE and RESURS_CLEAN_METHODS_CACHE_EXTEND added (Also related to #49811 and #48959) Also #49949	
56088	<i>Refactoring of form.phtml, scripts and dynamicForm. Stricter element checking on blurs, reload-issues and recapturing of entered values. Moving out functions out of controller</i>	
56988	MagentoFieldObject in MagentoResTransField is sometimes catching null. Now using typeof-checking.	
55637	Language updates	
55646	Prepared package for Magento up to 1.9.2	
55775	Loader lockups (Related to #56088) - Primary errors caused when AJAX-calls are timing out.	
55920	The plugin now ignores company fields, due to elements sort order in checkouts	
46777	Compatibility issues between different jQuery version (notice) - where prop and attr are acting different based on which version we are running	

57211	Sometimes tax is rounded with decimals in a way that ecom does not accept (at least for .fi)	
55746	More elements to remember	
57220	Forms when switching between payment methods are not cleaned up anymore	

1.3.8.1+1.3.8

Internal issue ID	Description of what's fixed	Magento versions (Default all)
46275	URL to correct test-environement is finally fixed	
44698	setInvoiceSequence-support added for finalizer, will set during "save"-click in admin	
46407	Products that should be tax-free are not accepted by ecommerce due to null values	
44711	jQuery v1.9 conflicts with RB distributed jQuery v1.5 on Vanillacheckout	
44976	QuantityCounterBehaviour in extended handler for bundles differs to the simple handler Also #44692	
45039	Corrected saltkey-generating in configcontroller (the order which the save-method is handling everything has been changed)	
44748	Removed confusing payment method from error messages in admin-view (getPayment)	
44448	"Read more" window opening has annoying behaviour. Fixed.	
44928	ucwords() changed to strtoupper() for country codes set from submitlimitapplication. Some instances does not like the ucwords()-handler	
1.3.8		
44509	Strict validations, after-effects fixed (Undefined properties) Also 44455 and 44493	
44110	Wrong error message started to show up on NO_DECISION and DENIED	
44698	setInvoiceSequence-support added for finalizer	
43777	Changed magento order admin view to check used payment method against getPayment for an order instead of internal call, to avoid problems with the RB-view	
44448	Read more for payment methods now always calling the same window to keep it topmost	
44149	Tax issues fixed from 1.3.5 Also #44364	
27148	Magento orderadmin view adjustments Also 27252, 43995 and 44657	

Plugin Changelog (deprecated versions)

1.4.2

Patch for SEKKI

1.4.1

[Magento 1.x:#57220] Old forms when switching between payment methods are cleaned up as an extra precaution
[Magento 1.x:#55645] List of payment methods are now included in the "Save"-action (admin) instead of page reloading
[Magento 1.x:#46563] Logotype images are replaced with new
[Magento 1.x:#58206] Canceling payments sometimes interferes with other payment methods
[Magento 1.x:#62183] Finalization with or without type INVOICE (has hotfix)
[Magento 1.x:#60194] When submitLimitApplication returns unexpected line breaks in address data, the JS fails

1.4.0

[Magento 1.x:#44045] Streamline release (hiding fields that are not necessary, and should be inherited from Magento's own form fields)
[Magento 1.x:#46599] Information about changing the invoice sequence numbers added
[Magento 1.x:#44397] Added row in admin that shows payment method thresholds (min - max)
[Magento 1.x:#49811] Payment method thresholds are not handled properly in checkout (exceeding min or max will hide method)
[Magento 1.x:#47333] getCostPopup in standardcheckout misbehaved and did not pop up properly. Fixed.
[Magento 1.x:#47492] Checkout problems for logged in users
[Magento 1.x:#48000] Too long article numbers shortened down to the database limit of max 100 characters, if length exceeded
[Magento 1.9:#48025] Fixed minor cosmetic defect for Magento 1.9
[Magento 1.x:#49407] LEGAL address-forcing is not handled properly in confirmation mail (firecheckout patch included)
[Magento 1.9:#49782] Magento 1.9 still using a deprecated getCostPopup method
[Magento 1.9.1:#55041/#54763] Conflicting jQuery, updated framework to comply with 1.9.1 and 1.9.2 requirements
[Magento 1.x:#55290] Dynamic forms, issues with phone elements fixed
[Magento 1.x:#50564] Fix for undefined indexes in Block/Checkout (Only visible in development environments)
[Magento 1.x:#50036/#49949] Environment variables RESURS_CLEAN_METHODS_CACHE and RESURS_CLEAN_METHODS_CACHE_EXTEND added (Also related to #49811 and #48959)
[Magento 1.x:#56088] Refactoring of form.phtml, scripts and dynamicForm. Stricter element checking on blurs, reload-issues and recapturing of entered values. Moving out functions out of controller
[Magento 1.x:#56988] MagentoFieldObject in MagentoResTransField is sometimes catching null. Now using typeof-checking.
[Magento 1.x:#55637] Language updates
[Magento 1.x:#55775] Loader lockups (Related to #56088) - Primary errors caused when AJAX-calls are timing out.
[Magento 1.x:#55920] The plugin now ignores company fields, due to elements sort order in checkouts
[Magento 1.x:#46777] Compatibility issues between different jQuery version (notice) - where prop and attr are acting different based on which version we are running
[Magento 1.x:#57211] Sometimes tax is rounded with decimals in a way that ecom does not accept (at least for .fi)
[Magento 1.x:#55746] More elements to remember
[Magento 1.x:#57220] Forms when switching between payment methods are not cleaned up anymore

1.3.9

Internal testing version only

1.3.8.1

[Magento 1.x:#46275] URL to correct test-environment is finally fixed
[Magento 1.x:#44698] setInvoiceSequence-support added for finalizer, will set during "save"-click in admin
[Magento 1.x:#46407] Products that should be tax-free are not accepted by ecommerce due to null values
[Magento 1.9:#44711] jQuery v1.9 conflicts with RB distributed jQuery v1.5 on Vanillacheckout
[Magento 1.x:#44976/#44692] QuantityCounterBehaviour in extended handler for bundles differs to the simple handler
[Magento 1.x:#45039] Corrected saltkey-generating in configcontroller (the order which the save-method is handling everything has been changed)
[Magento 1.x:#44748] Removed confusing payment method from error messages in admin-view (getPayment)
[Magento 1.x:#44448] "Read more" window opening has annoying behaviour. Fixed.
[Magento 1.x:#44928] ucwords() changed to strtoupper() for country codes set from submitlimitapplication. Some instances does not like the ucwords()-handler

1.3.7

[Magento 1.x:#43754] Complete finalize/credit/annul for extended bundles

1.3.6

[Magento 1.x:#42486] Implementation - Automated Finalizepayment (Works for the non-extended-bundle-handler only)
[Magento 1.x:#42940] Implementation - Automated Creditpayment/Annulpayment (Works for the non-extended-bundle-handler only)
[Magento 1.x:#42986] Language updates related to #42486, #42490, #41754, #43254 and others
[Magento 1.x:#43145/#43974] Strict validation from ecommerce gives error at place order
[Magento 1.x:#43053] Description-lines in special rules (fee, etc) is needed in paymentspec, for successful finalizings (Related to #43145)

[Magento 1.x:#41872] Replaced some isNaN with getNaN due to some phone-numbering issues
[Magento 1.x:#40492] Fixed a hard-coded "Close"-button from getCostOfPurchase
[Magento 1.x:#41726] Prevention against disabled payment methods has been improved (when having 2 methods and one of them gets disabled)

1.3.5

[Magento 1.x:#41851] Completion of bundle handler (Major again)
[Magento 1.x:#41412] Charset issues in language files fixed by changing to UTF8
[Magento 1.x:#32291] Unitmeasure issue for norway (path to language files nb_NO should be nb_NO)
[Magento 1.x:#40485] Send plugin version through metadata at bookpayment (Has been reverted)

1.3.4

[Magento 1.x:#40424/#39220/#39221] Major fixes for bundled articles and how paymentspecs are created for the Api (including tax and discount issues)
[Magento 1.x:#39714] Problems with reverse tax calculations in when using "apply discount after tax"
[Magento 1.x:#38100] js-fixes for checking phone numbers
[Magento 1.x:#39218] Issues with multiplechannel scopes, payment methods, callbacks and salt keys, now pointing back to a static place for this to not conflict with older versions/environments
[Magento 1.x:#39217] Issues with order-view in admin where not all payment methods shows up depending on which scope and store the administrators is looking at

1.3.3

[Magento 1.x:#38447] Fixes in Magento Order-admin-view
[Magento 1.x:#38758/#37556] Charset problems with getCostOfPurchase
[Magento 1.9:#34018] Broken drop-downs in Magento 1.9 caused by different jQuery-libraries

1.3.2

[Magento 1.x:#38421] If there is only one payment method, the primary configuration in Magento-admin disappears
[Magento 1.x:#33900] Contact phone information added to "denial"-messages
[Magento 1.x:#38623] When combining bundled products with regular products product id's are sometimes colliding in specrows (if the bundle contains the same items as a regular)
[Magento 1.x:#32291] Unit measure that is sent to betaladmin is now dynamically set from language files (The "pcs" translation)
[Magento 1.x:#37531/#37426] Language issues

1.3.1

[Magento 1.x:#37555] Related to #36137 - fixed a static view in css for CostOfPurchaseHtml
[Magento 1.x:#37663] Fixed a conflict with callback urls in a multichannel
[Magento 1.x:#37664] Fixed a conflict with payment methods in a multichannel
[Magento 1.x:#38011] Payment method-backwards compatibility in order-admin-view from Magento. Older methods in order view fails if they mismatch with new naming method.
[Magento 1.x:#36830] When payment methods fail due to denial in submitlimitapplication or similar, payment methods should only be unselected, not disabled.
[Magento 1.x:#37211] Payment methods that has been made unavailable (expired, disabled, etc) from Resurs Bank should be checked on fly if errors occurs during payment
[Magento 1.x:#6629] Autofilling forms when phone/cellphone-numbers are changed under some circumstances
[Magento 1.x:#38096] Try-catch-fixes for getAddress, if services is down

1.3.0

[Magento 1.x:#37138] No feedback when getAddress fails
[Magento 1.x:#6629] Autofilling between magento forms and dynamic forms improved
[Magento 1.x:#36853] Error messages are not shown on denied checkouts, user just "jumping back" to last form
[Magento 1.x:#36137] Beautified CostOfPurchaseHtml by adding css and changed popup method for it (window.open instead of a hidden div)
[Magento 1.x:#36671/#27444] Bug: Address is not overridden properly in order confirms (Related to #36118)

1.2.9

[Magento 1.x:#36127] Form for getAddress removed from paymentMethods popup. Replaced with code examples, so each representative can put the getAddress (if needed) in the right place of checkout

1.2.8

[Magento 1.x:#35796] Changed error messages for failed signings
[Magento 1.x:#35971] Hide "Read more" for all payment method types "CARD", since there's normally no information present there
[Magento 1.x:#36032] Payment method country issue, where methods disappear when a country like "Se" is mismatched with an array containing "SE".

[Magento 1.x:#36118] Addressdata is forced into webforms even if submitlimitapplication fails or is not granted (Needs patch to work in Fire checkout!!)
[Magento 1.x:#25487] Removed buttons for hashkeys in admin, hashes are now updated at ecommerce on each save

1.2.7

[Magento 1.x:#33493] Optional choice for redirects on failed signings - normally orders are annuled and sent back to cart. Now, redirect to checkout are added.
[Magento 1.x:#32349] New state for signing addded, to identify aborted signings (where user for example used back button during a signing process, which will empty the shopping cart even if the order is not completed)
[Magento 1.x:#35332] Fix for bundled products when using fixed prices

1.2.6

[Magento 1.x:#35108] Hotfix, update for making older internal methods match with current requirements.

1.2.5

[Magento 1.x] Forcing address-data from Resurs Bank is now allowed in Norway. Fix has been applied.
[Magento 1.x:#32396] Method ids are from now not called Resurspayment<methodid>, instead their new name is Resurspayment<representativeid><methodid>, to make multichannels etc less sensitive against colissions.
[Magento 1.x:#34956] Clarified description of the "getAddress"-form (Text: "Verify your social security number") with correct SSN Format. Sweden finished (still required for NO, DK, FI)

1.2.4:

[Magento 1.x] Remarked code passwordexpiration removed (expiration is no longer used by Resurs Bank)
[Magento 1.x] Fix for payment methods abstract files in case of non numeric methods
[Magento 1.x:#28763] Detection of SSL-support (From API, so that representatives without SSL-supported webservers may get warnings when not available)
[Magento 1.x:#323251] Autoloader sometimes fails loading methoddata in mage-devel on view orders in admin
[Magento 1.x:#33102] Continued fix for #31614 which affected errors in submitLimitApplication in the same procedure.
[Magento 1.x:#32613] Changed patchfiles for Firecheckout and Onestep plugins to give more details on submitLimitApplication. Language updated
Added patchfiles for Firecheckout 2.5.7 (including patchcode if you have other versions than 2.4.4 or 2.5.7), and changed filenames for the controllerfiles in the path where patches can be found
[Magento 1.x:#32579] Onepagecontroller has undefined variables (trigged by old test environment where address data was not returned properly)

1.2.3:

[Magento 1.x:#32403] Unavailable payment methods translation change
[Magento 1.x:#32267] Cookie throttling on payment-errors removed
[Magento 1.x] Removed method generateSecurePassword which is no more in use (and mis-spelled) and cleaned up old code
[Magento 1.x:#28814] Bundled products will now split up and be listed separately in betaladmin (but still be marked as bundled products)
[Magento 1.x:#32085] Errors occurring during test sessions, when restoration of backups causes mismatches in incremental order id's
[Magento 1.x:#32050] Headers already sent caused by dynamicformAction in IndexController.php
[Magento 1.x:#31642] Information about government-id's (personnummer) removed from admins order view
[Magento 1.x:#31614] In rare cases, other payment methods won't play together with our plugin
[Magento 1.x:#28548] Environment on a default installation now always pointing to test instead of production
[Magento 1.x:#28063] Logotype branding based on payment method
[Magento 1.x:#6628] Overlapping text failures, needed extended fix for some linebreaks after selection
[Magento 1.x] Changed to a safer method to pick up current scope/channel for payment methods in admin
[Magento 1.x:#27865] Payment methods that does not belong to a representative account are now hidden
[Magento 1.x:#25691] The box "payment method example" containing urls that does not open in a new window. Target set to _blank by a regexp-replace, since html is serverside

1.2.2:

Password-expire-email-template removed from config.xml
[Magento 1.x:#6628] CSS Fix for Firecheckout that affects the position of "read more"
[Magento 1.x:#27445] Extra errorhandlers logging plugin callbacks
[Magento 1.x:#27445] If statuses set to complete in a procedure as unfreeze, Magento's rule is not set to complete.

1.2.1:

[Magento 1.x:#25652] Use of customized logotypes for Resurs Bank payment methods (If the url begins with /, we're always defaulting to the path of Magento's skin directory)
[Magento 1.8:#25801] Billing address are not forced from SSN in Vanilla onepage. Changes made for 1.8 only.

[Magento 1.8:#26313] Shipping address are not forced from SSN in Onestepcheckout. If same as billing-address, the force of address should invoke too.
Changes made in the method so this affects both 1.7 and 1.8 (even if it already works in 1.7)
[Magento 1.8:#26326] Customer's full name are displayed in an improper way (randomly): "customer->address->fullname?>" instead of the correct value
[Magento 1.x:#26450] Field for digest regarding automatic_fraud_control should be empty like the other fields on "first time installations"
[Magento 1.x:#27181] Payment info (Magento Orderadmin) has been more sensitive in scoped environment. The blue "resurs box" only showed up properly in main site, due to different scopes and some data missing.

1.2.0

Added support for multiple websites, possible to use one representative per webstore (scopes)
Behaviour of saltkey fields changed (scopes)
Callbacks changed to support multiple websites (scopes)

1.1.39:

Backups may not warn on screen when there are issues with writing in the backup paths. They are therefore logged from now on (Payment methods and config controllers)
registerCallbackEvent fixed and should work again

1.1.38:

GetAddress issues for vanilla fixed. Application-Government-Id now inherits data from the getaddress-field (#11678)

1.1.37:

Disabled cache-flushing each time salt keys are updated in control panel since it delays too much (and still, you need to flush the cache on other changes anyway, if it is active)
Function updatecallbackregistrations in indexcontroller upgraded - on password or username changes callbacks will be updated automatically
Cron.php sometimes generates unwelcome errors when payment methods are not available
Throttle of payment methods implemented in onestepcheckout and vanilla (Not located in the plugin: Check the patch-directory)
Current cookiethrottler had some fixes

1.1.36:

Removed "&" from password-generator since this character are translated as a html entity in the soapcall
Fixed a copy problem in Model/Cron.php
Payment methods does not load properly on first click (Onestepcheckout)
Some payment-methods generate errors when \$length in the form is not defined (Affecting both Onestep and vanilla-magentos)

1.1.35:

Extra complexity in mkpass() for password_expire
Letting Resurs Bank generate a session id instead of using microtime() or uniqid() for doing this
Patches for Firecheckout should only be delivered via patch-files, not through the installation package
Secured Api.php against undefined variable errors (during debug mode)
Custom statuses works better with magento's states
Changed supported countries from multiselect to single selection
Password expiration callback throttler added (If too many requests are received in the same time, throttle password-changes and only accept the first received request)
form.phtml in design/base vs design/default resynced since they differ
Throttle of payment methods implemented in the prepareSigning-method (on failed signings)

1.1.34:

Checking count of methods instead of returning unreadable soap error in cron.php
password_expiration and identifier, safer digest controls for multiple passwords in indexcontroller
Promotion codes should not be rounded (Tax problem)
Another tax-problem on while using promotion codes (depending on when the discount should be applied - before or after tax) are fixed
Added extra exception for some javascript-calls in IndexController (regarding sh_a_Govld and sh_a_Mob)
First prevention of headers already sent from indexcontroller by changing \$dynamicFormHtml \$rowdata (placing the variable on one row seem to solve problems)
Fixing a bunch of variables that needs to be defined to be used properly (if not, they are generating "undefined variable"-errors)
Firecheckout patch for submitlimitapplication: Empty variables returned gives us problems with undefined variables (found on addressrow2)
htmlentities() removed from id and artNo in speclines (Model/Api.php) due to conflicting long names on encoding

OldFlow Plugin - Coding examples for getAddress

Validation of Social Security Numbers (Personnummer)

Example to enable getAddress to autofill address-forms in Firecheckout, Onestepcheckout and Vanilla checkout (the standard checkout). This is enabled from v1.2.9

Source code:

getAddress

```
<?php if (!$_this->isCustomerLoggedIn()) : ?>
<p class="ssn-label">
    <?php echo $_this->__('Verify your social security number') ?><br>
    <?php echo $_this->__('SSN Format') ?>
</p>
<p class="ssn-input-controls">
    <input name="resurs-ssn" id="resurs-ssn" type="text" class="ssn-input-text" />
    <!-- Observe: getAddress is using "natural" as default -->
    <input name="resurs-ssn" id="resurs-ssn-button" type="button" class="resurs-ssn-input-button" value="
<?php echo $_this->__('Get Address') ?>" onclick="getAddress('<?php echo $_this->getUrl('resursbank/index
/getAddress')?>','NATURAL','<?php echo $_this->getSkinUrl('images/opc-ajax-loader.gif')?>')" />
</p>
<div class="resurs-ssn-address-container" id="resurs-ssn-address-container"></div>
<?php endif; ?>
```

Fire Checkout (app/design/frontend/base/default/template/PATH-TO-FIRECHECKOUT-TEMPLATES/checkout/billing.phtml):

Around line 96, just above the li-tag for "billing-new-address-form".

OneStep Checkout (app/design/frontend/base/default/template/PATH-TO-ONESTEPCHECKOUT-TEMPLATES/checkout.phtml):

Around line 53, just after the first p-tag for "onestepcheckout-numbers onestepcheckout-numbers"

Vanilla Checkout (app/design/frontend/base/default/template/persistent/checkout/onepage/billing.phtml and/or app/design/frontend/base/default/template/checkout/onepage/billing.phtml)

Around line 39, just after the first endif tag (above li for "billing-new-address-form")

For Magento 1.9 you may also want to look at app/design/frontend/rwd/default/template/persistent/checkout/onepage/billing.phtml, in case of different templates.

Notes:

Address lookups is set to NATURAL (private customer) in this code example, since "Company" is normally not filled when entering the checkout. To make getAddress use of company-lookup (LEGAL) instead of private, the company field has to be filled in before doing the getAddress-call. In that case, getAddress will automatically switch to LEGAL.

Manually patching Firecheckout

1. In the folder firecheckout/controllers, edit IndexController.php, and find saveOrderAction() (In firecheckout 2.57 it's around row 1244):

```
saveOrderAction()  
  
public function saveOrderAction()  
{
```

2. Put the code from [MagentoFC_Patch.txt](#) (last update 2015-09-10) right after this code:

```
expireAjax  
  
if ($this->_expireAjax()) {  
    return;  
}
```

3. Save the file

What was changed?

2015-09-10

In some exclusive cases 'customerinfo' fails (mostly if logging to display is active), so we've changed this row (in our patchfile located at row 1307 - may differ depending on how your set up looks):

```
$customerinfo = (isset($limitResult['customerinfo'])) ? $limitResult['customerinfo'] : array();
```

```
$customerinfo = (isset($limitResult) && !empty($limitResult) && isset($limitResult['customerinfo']) && !empty  
($limitResult)) ? $limitResult['customerinfo'] : array();
```

Manually patching Onestep checkout

1. In app/design/frontend/base/default/template/onestepcheckout/checkout.phtml, Look around line 181, After ...

Payment method errors

```
<?php if(isset($this->formErrors['payment_method'])) && $this->formErrors['payment_method']: ?>
    <div class="onestepcheckout-error onestepcheckout-payment-method-error">
        <?php echo $this->__('Please choose a payment method.');?>?
    </div>
<?php else: ?>
```

Replace this code

```
<?php if(isset($this->formErrors['payment_method_error'])): ?>
    <div class="onestepcheckout-error onestepcheckout-payment-method-error">
        <?php echo $this->__('Please enter valid details below.');?>?
    </div>
<?php endif; ?>
```

... with this code ...

```
<!-- Begin display Resursbank Error -->
<?php if(isset($this->formErrors['payment_method_error']) && !empty($this->formErrors['resursbank_error'])): ?>
    <div class="onestepcheckout-error onestepcheckout-payment-method-error">
        <?php echo $this->formErrors['resursbank_error'];?>?
    </div>
<?php elseif(isset($this->formErrors['payment_method_error'])): ?>
    <div class="onestepcheckout-error onestepcheckout-payment-method-error">
        <?php echo $this->__('Please enter valid details below.');?>?
    </div>
<?php endif; ?>
<!-- End display Resursbank Error -->
```

2. In the bottom of the file, add the code below. In this example disabling of the payment method is remarked, so the customer may choose the payment method again, immediately after a payment method error.

Resurs Bank Code Block

```
<!-- Begin Resursbank Block -->
<?php
    if($paymentId = Mage::getSingleton('checkout/session')->getDisablePaymentMethodId())
    {
        echo "<script type='text/javascript'>
            \$('#method_resurspayment$paymentId').checked = false;
        </script>";
        Mage::getSingleton('checkout/session')->setDisablePaymentMethodId('');
    }
?>
<!-- End Resursbank Block -->
```

Plugin Hotfixes

This page contains hotfixes for the OldFlow release.

2016-02-11

A problem discovered in the finalization process that affects payment methods that is not of the type "INVOICE". This can be manually fixed by editing the file **./app/code/lcoal/Resursbank/Resursbank/Model/Api.php**:

Look up:

```
public function finalizePayment(Varien_Object $payment)
```

Find this code:

finalizePaymentArray

```
$finalizePayment = array(
    'paymentId' => $additionalData,
    'preferredTransactionId' => $orderId,
    'partPaymentSpec' => $paymentSpec,
    'createdBy' => 'SHOP_FLOW',
    'orderId' => $orderId,
    'orderDate' => date('Y-m-d', time()),
    'invoiceId' => $invoiceId,
    'invoiceDate' => date('Y-m-d', time()),
    'ourReference' => utf8_encode($order->getStore()->getWebsite()->getName()),
    'yourReference' => utf8_encode($billing->getFirstname() . " " . $billing->getLastname())
);
```

Replace with this code:

finalizePaymentArray-Fix

```
/* #62183 - Begin - Finalization with or without type INVOICE */
$isInvoice = false;
$methodType = null;
try {
    $methodCode = $payment->getMethodInstance()->getCode();
    $methodInformation = Mage::getModel('resursbank/methods')->getCollection();
    $methodInformation->addFieldToFilter('payment_id', str_replace('resurspayment', '', $methodCode));
    foreach ($methodInformation as $method) {
        $methodType = $method->getPaymentType();
        if (!empty($methodType)) {break;}
    }
} catch (Exception $e) {}
if (strtoupper($methodType) === "INVOICE") { $isInvoice = true; }
$finalizePayment = array(
    'paymentId' => $additionalData,
    'preferredTransactionId' => $orderId,
    'partPaymentSpec' => $paymentSpec,
    'createdBy' => 'SHOP_FLOW',
    'orderId' => $orderId,
    'ourReference' => utf8_encode($order->getStore()->getWebsite()->getName()),
    'yourReference' => utf8_encode($billing->getFirstname() . " " . $billing->getLastname())
);
if ($isInvoice) {
    $finalizePayment['orderDate'] = date('Y-m-d', time());
    $finalizePayment['invoiceId'] = $invoiceId;
    $finalizePayment['invoiceDate'] = date('Y-m-d', time());
}
/* #62183 - Finish */
```

2015-02-26

Some Magento-instances (primary the one step checkout) doesn't accept the ucwords()-translation that belongs to the strict validation, so from now on they are set to uppercase in submitlimitapplication. Checkout.php and OnepageController.php has been patched and will be released with v1.3.8.1 (find the files below)

2015-01-19

Stricter validation from ecommerce generates errors on "place order". This is fixed from version 1.3.6, but is not yet released. Currently, test environment are affected.

For Firecheckout, [patch files are updated here](#).

For Vanilla and the Onestep-checkout, edit app/code/local/Resursbank/Resursbank/Block/Checkout.php around line 36, find following code:

Original code < 1.3.6

```
$slaAddress = array(
    'firstName' => (isset($_POST['billing']['firstname']) ? 
$_POST['billing']['firstname'] : ""),
    'lastName' => (isset($_POST['billing']['lastname']) ? 
$_POST['billing']['lastname'] : ""),
    'addressRow1' => (isset($_POST['billing']['street'][0]) ?
$_POST['billing']['street'][0]),
    'addressRow2' => (isset($_POST['billing']['street'][1]) ?
$_POST['billing']['street'][1]),
    'postalCode' => (isset($_POST['billing']['postcode']) ? 
$_POST['billing']['postcode'],
    'postalArea' => (isset($_POST['billing']['city']) ? 
$_POST['billing']['city'],
    'country' => (isset($_POST['billing']['country_id']) ? 
$_POST['billing']['country_id'])
);
```

Replace with:

Changed code >= 1.3.6

```
$slaAddress = array();
if (isset($_POST['billing']['firstname']) && trim($_POST['billing']['firstname']) != "") {
    $slaAddress['firstName'] = $_POST['billing']['firstname'];
    if (isset($_POST['billing']['lastname']) && trim($_POST['billing']['lastname']) != "") {
        $slaAddress['lastName'] = $_POST['billing']['lastname'];
        if (isset($_POST['billing']['street'][0]) && trim($_POST['billing']['street'][0]) != "") {
            $slaAddress['addressRow1'] = $_POST['billing']['street'][0];
            if (isset($_POST['billing']['street'][1]) && trim($_POST['billing']['street'][1]) != "") {
                $slaAddress['addressRow2'] = $_POST['billing']['street'][1];
                if (isset($_POST['billing']['postcode']) && trim($_POST['billing']['postcode']) != "") {
                    $slaAddress['postalCode'] = $_POST['billing']['postcode'];
                    if (isset($_POST['billing']['city']) && trim($_POST['billing']['city']) != "") {
                        $slaAddress['postalArea'] = $_POST['billing']['city'];
                        if (isset($_POST['billing']['country_id']) && trim($_POST['billing']['country_id']) != "") {
                            $slaAddress['country'] = $_POST['billing']['country_id'];
                        }
                    }
                }
            }
        }
    }
}
```

Do the same for app/code/local/Resursbank/Resursbank/controllers/Checkout/OnepageController.php (around line 259)

You can also download the updated files [here \(Checkout.php\)](#) and [here \(OnepageController.php\)](#)

Release- and installation notes, error logging and development for Magento OldFlow

Developing Magento

Cache

Make sure that as much as possible of Magento's caching features are disabled. Old caches may be heavy showstoppers when things normally works ok and suddenly stops. This is especially occurring when many things are changing shortly after each other. A good thing otherwise, is to clear caches often.

After installation notices

Older plugin releases and test environments

For older releases, make sure that test environment are pointed to <https://test.resurs.com/ecommerce-test/ws/V4/>

Test versus production

Data in test environment may differ from data that can be found in production environment. For example, `getCostOfPurchaseHtml` may present more information on payments in production, depending on chosen payment method, than it normally does in test.

Configuration

Don't forget to check out settings and configurations below, since there are things that has to be done before the plugin works properly, like setting up merchant credentials.

Make sure that you are really setting this to what is preferred usage for your store.

The screenshot shows the 'Viktiga inställningar' (Important Settings) configuration page in the Magento Admin. It includes fields for:

- Server: Environment URL (Test, Production URL: https://test.resurs.com/ecommerce-test/ws/V4/)
- Resurs Bank WebServices ProduktionsURL: https://ecommerce.resurs.com/ws/V4/
- Resurs Direct WebServices Testurl: https://test.resurs.com/ws/V4/
- Addresscode filer:
- Payment metoder:
- Orderstatus för ny order:
- Status för annan order vid godkänt:
- Om beställningen är pågående: Test, Be summary om beställningen är pågående, Mötte finns i den ska lägg upp
- Flyt uppgradera upp till status:
- Aktiverat order status:
- Omberäknas för pågående order:
- Begagnat:
- Omräkningspris i Magento när sätter pris:

Error handling in production environments

For a production environment: Make sure that `MAGE_IS_DEVELOPER_MODE = "true"` is disabled (in .htaccess) if enabled, together with `display_errors` (php.ini) - unless you don't want it to be very verbose. Other plugins (including this) may generate errors that makes the plugin stop working properly, since the AJAX routines are depending on correct returned values. For example, plugins that contains pre-undefined variables, may create log-notices this way, that may halt scripts before they are finished. Those settings may also cause same kind of errors in a test environment.

Release notes

Version 1.2.9

The form for which the customers can enter their own SSN to confirm their home address has been removed and is now put into code examples which can be found here

Version 1.2.5

This version contains a major change of how payment-methods are saved in configuration. When upgrading a reconfiguration of your old payment methods may be needed. We've also added patch files to manually patch up newer versions of Firecheckout (look in the patch-path in the zip-file). **The new format (in the Model-path) is Resurspayment<representativId><paymentMethodId>.php**

Bugs, etc

Pricing bugs

Discover dates:

2014-02-03, Klarna
2015-02-23, Payson

There may be problems with shopping cart price rules, when running our plugin together with Klarna, Payson and possibly more plugins, due to misconfiguration. Especially shipping and order sums may fail in the final order.

How to fix:

Make sure etc/config.xml (in Klarna's plugin) have shipping included in the <before>-tag under <sales><quote><totals><klarna_fee>. Likewise, there may be problems with the shopping cart information (confirmed with Magento 1.9) when working together with Payson plugin (Discovered 2015-02-23).

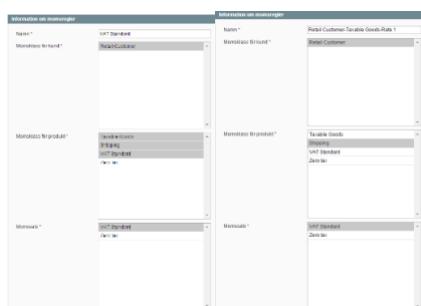
Add shipping subtotal at the <before>-tag at <sales><quote><totals><payson_invoice> (etc/config.xml in Payson's plugin)

There is however not a guarantee that the fix is completely solved this way; problems with other miscalculations (like double taxes) has been reported to us.

Technical issues on tax classes

Found 2015-03-24

Some tax classes may cause technical issues with webservices, and some settings in Magento's own configuration may give double taxes. Currently, there is no fix for this, since it's configurable outside the space of Resurs Bank.



Payment method thresholds, min and max boundaries

Technical issues should normally not occur, when a price is too low or too high for a payment method, since the payment method automatically checks the boundaries during the payment. However, if it happens, it may be caused by an older cached payment method. To fix this, you need database access, so you can remove the misconfigured payment methods from the table **resursbank_payment_methods**.

<u>id</u>	<u>payment_id</u>	<u>description</u>	<u>legal_info_links</u>	<u>min_limit</u>	<u>max_limit</u>	<u>payment_type</u>	<u>status</u>	<u>user_id</u>
30	magfire109	Företagsfaktura Magento firecheckout		10	50000	INVOICE	0	magfire
31	magfire110	Befintigt kort Magento firecheckout		0	2147483647	CARD	0	magfire
32	magfire111	Nytt kort Magento FireCheckout		1000	50000	REVOLVING_CREDIT	0	magfire
33	magfire112	Faktura privatperson Magento FireCheckout		10	50000	INVOICE	0	magfire
34	magentosubfire111SUBFC	Nytt kort Magento SubFireCheckout		1	50000	REVOLVING_CREDIT	0	magentosubfire
35	magentosubfire112SUBFC	Faktura privatperson Magento SubFireCheckout		10	50000	INVOICE	0	magentosubfire
36	magfire113	Faktura utan signering		10	50000	INVOICE	0	magfire
37	magfire114	Delbetalning		10	50000	REVOLVING_CREDIT	0	magfire
38	magfire115	Signering 10		1	50000	REVOLVING_CREDIT	0	magfire
39	magfire116	Uppskj fakt eft kampanj		10	50000	INVOICE	0	magfire
40	magfire117	Delbetalning		10	50000	REVOLVING_CREDIT	0	magfire
41	magentosubfire112SUBFCFAKTURA	Faktura privatperson Magento SubFireCheckout		10	50000	INVOICE	0	magentosubfire
42	magfireTEST	test		1	50000	INVOICE	0	magfire

Magento: Finalize-, credit- and annulpayment details (v1.3.8.1)

From version 1.3.8.1, automated finalizepayment, creditpayment and annulpayment are supported. Functionality related to the aftershopflow is by default disabled, since the new flow also affects the booking-stage for some product types. The settings can however be activated from the administration panel in Magento (in the payment methods section).

It is also very important to use the "regular flow", since the extended handler for bundled products is not yet fully supported.

By default, the settings looks like this:

The screenshot shows the 'API' section of the Magento Admin. Under 'Hantering av paymentspecs', there is a warning: 'Varning: Experimentella inställningar! Ändra dem bara om du vet vad du sysslar med'. Below it is a dropdown menu set to 'Use a regular flow to add specines (Default an)'. Under 'Automatiska debiteringar, krediteringar och annulleringar', another dropdown menu is set to 'Hantera inte automatiskt'. Under 'Fakturanummer-hantering vid debiteringar, krediteringar och annulleringar', a third dropdown menu is set to 'Låt Resurs Bank hantera fakturanummer'.

As you also can see at the image, this flow currently only accepts that invoice ID's are handled by Resurs Bank and can not be switched over to Magento-handling.

As soon as the flow is activated the shopflow will start working as follows:

- Finalizepayment/debit are only invoked when the order is fully shipped to the customer. This means that partial debits are currently not possible.
- Finalizepayment/debit will invoke immediately after booking, if the order only contains virtual or downloadable products, since they are not able to ship through Magento
- Creditpayment and annulpayment are invoked on order cancellations and the plugin automatically detects, depending on the order, which method it will use

Where and when it happens

Finalizing, credits and annulments are all methods controlled from Magento Orderview, and are activated when you begin your shipping process.

Finalize/Debit order

The screenshot shows the 'Order View' page for Order # 13400001052. At the top, there are buttons for Back, Edit, Cancel, Send Email, Hold, Invoice, and Ship. The 'Ship' button is highlighted with a pink box. Below the buttons, there is a section titled 'Information från Resurs Bank' with instructions about address matching and invoice handling. A note at the bottom says 'Tänk på att fakturan måste sändas innan ändringar kan göras på ordern.' (Remember that the invoice must be sent before changes can be made to the order.)

Currently, no finalization are envoked before the full order has been shipped. If you are only partially delivering your order, nothing is going to happen.

As you are submitting your shipment in full, Betalningsadmin is also updated together with your order.

Information från Resurs Bank										
Detta är adressen Resurs Bank ger tillbaka från kreditupplysning. Se till att matcha informationen i ordning.										
Fakturan är skickad: 2015-02-12										
Betalningsid										13400001052
Ordersumma										32109
Kreditgräns										32109
Bedräget?										No
Fryst?										No
Kundens namn										Bo Börje Bertilsson
Leveransadress										Ekslingen 7 Helsingborg SE - 25467
Kundens telefonnummer										

Credit and annulments

Crediting and annulling orders are made through Magento's cancellation button, and depending on the current order it will either annul or credit the payment.



In the above example, a creditpayment will invoke since the order has been debited (shipped). If it has not been shipped, the order will instead get annulled.

Orderspecifikation										
= Auktoriserat = Annulerat = Debiterat = Krediterat										
Artikelnummer	Beskrivning	Styckpris exkl. moms	% moms	Summa	Varav moms	Auktoriserat	Annulerat	Debiterat	Krediterat	
DenBilligaBilen	Den billiga bilen Tesla - Too cheap for you	25500,00	25	31875,00	6375,00				1 st	
SKU_Skyddshandskar	Skyddshandskar	95,20	25	119,00	23,80				1 st	
shippingfee	Fraktvagn	90,00	0	90,00	0,00				1 st	
paymentfee	paymentfee	20,00	25	25,00	5,00				1 st	

Orders are also, always, logged in the order history of Magento

- 12 feb 2015 10.21.18 | Canceled**
Customer Notification Not Applicable
- 12 feb 2015 10.21.18 | Complete**
Customer Not Notified
Betalningen har krediterats
- 12 feb 2015 10.17.05 | Complete**
Customer Not Notified
[13400001052] Betalningen har debiterats
- 12 feb 2015 10.14.13 | Complete**
Customer Not Notified
- 12 feb 2015 10.13.44 | Processing**
Customer Not Notified
Lyckades på Resurs Banks sida
- 12 feb 2015 10.13.43 | Payment signing**
Customer Not Notified
Resurs Bank pågående signering
- 12 feb 2015 10.13.42 | New**
Customer Notification Not Applicable
Order created
- 12 feb 2015 10.13.42 | New**
Customer Notified ✓

MagentoPlugin - Important notices, features, information

Tax calculation methods (and discounts)

Issues with tax calculation, especially when applying tax to products with giftcards/discount, has been reported to magentocommerce. [Magento themselves](#) strongly recommends that you - from administration - apply tax **after discount** and using tax on catalog prices to avoid calculation issues.

My payment methods are showing up, even when the allowed amount are too high or too low

Problem: When you are trying to check out with an amount - higher or lower than the payment method normally accepts - and the payment still shows up in the checkout.

Cause: Starts in the table *resursbank_payment_methods* where all payment methods are stored. They sometimes need to be emptied to sync properly, especially in test environments where min/max amount are changed often.

Solution: Running apache? Add this row to your .htaccess-file:

```
SetEnv RESURS_CLEAN_METHODS_CACHE "true"
```

Then update your payment methods again. By doing this, you will clean up the payment methods cache and add the new proper settings. This variable is limited to the current representative id, so if you have more than one store, the cleanup will only occur for the current representative. If you want to do a full clean-up with this method, in all (if you have this configured) stores, you should also add the variable **RESURS_CLEAN_METHODS_CACHE_EXTEND** to .htaccess

Credit cards and payment methods that requires payment signing

If your store needs payment signing in some of the payment steps, make sure you assign "Signing status" correct status. This is made by creating a custom order status in magento like this:

New Order Status

Orderstatus information	
Statuskod *	paymentsigning
Status Label *	Payment signing

When the status is created, you have to assign it to a state. In our example, we use "Hold/Holded" ("Parkerad/Pausad") like this:

Tilldela orderstatus till state

Tilldelningsinformation	
Status för beställning *	Payment signing
Order State *	Parkerad/Pausad
Använd orderstatus som default	<input type="checkbox"/>

When the status has been created, assign it to the assignment rules in our payment method configuration:

Orderstatus för pågående order	Behandlas
Signing status	Payment signing
Omdirigeringsplats i Magento när signeringar misslyckas	-- Vänligen välj -- Ny Klarna Reserved Behandlas Slutförd Stängd Avbruten Parkerad/Pausad Payment signing
Visa detta textmeddelande när fel inträffar	

The reason is to mark the order to indicate that it is in a "Payment signing"-state if something goes wrong during the payment signing.

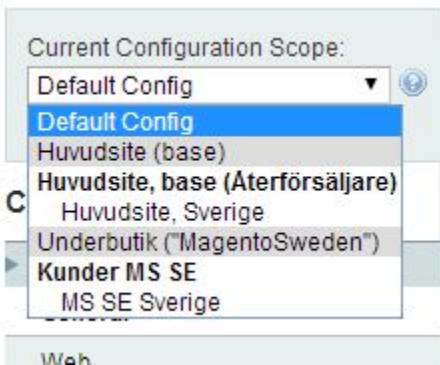
Multiple stores with the plugin

Since Resurs Bank plugin v1.2.0 the behaviour has changed to support more than one scope/channel/store - here is some information on how it works.

In this example, we have set up one main store and one substore.

Manage Stores		
+ Create Website + Create Store + Create Store View		
Page <input type="button" value="1"/> of 1 pages View <input type="button" value="20"/> per page Total 2 records found		Reset Filter Search
Website Name	Store Name	Store View Name
Huvudsite (base)	Huvudsite, base (Återförsäljare)	Huvudsite, Sverige
Underbutik ("MagentoSweden")	Kunder MS SE	MS SE Sverige

From now, all payment methods are shown from all sites configured from the store-editor, so it is important that you change your configuration for each store (each store view actually), since each store probably have their own payment methods and Resurs Bank only supports one representative per site. When configuring representatives you should loop through all your sites and fix them, starting with the default/main config.



Important note:

It is highly important that each store has its own environment, callback credentials and representative data configured, so make sure you check all settings for each store you are configuring!

Vanliga inställningar		
Server	<input type="text" value="Test"/>	[STORE VIEW]
Resurs Bank Webservices Produktionsmiljö	<input type="text" value="https://ecommerce.resurs.com/ws/V4/"/>	[STORE VIEW]
Resurs Bank Webservices Testmiljö	<input type="text" value="https://test.resurs.com/ecommerce/ws/V4/"/>	[STORE VIEW]
Lagra inloggningsuppgifter		
Kund / Representant ID	<input type="text"/>	[STORE VIEW]
Lösenord för kund / representant	<input type="password"/>	[STORE VIEW]

And: Don't forget to save configuration! Updating payment methods is not enough.



Payment methods

In the main configuration for the plugin, the settings should look something like this. In the illustration below the "SubFirecheckout"-method is a payment method that does not belong to the representative in this store. Therefore, we set that to "No" (inactive). This should be fixed in the sub-store-views after handling the main configuration. When everything needed is set here, it should be okay to go on with the next store.

Faktura privatperson Magento FireCheckout

Betalningsstatusen är aktiv?	<input type="text" value="Yes"/>
Title	<input type="text" value="Faktura privatperson FC"/>
Sorteringsordning faktura	<input type="text"/> ▲ Vikt av valet. Lägre värde innebär högre placering.
Avgift för att använda betalningsalternativet	<input type="text"/>
Beskrivning för betalningsalternativets avgift (kommer visas på orderraden)	<input type="text" value="Beskrivning för betalningsalternativets avgift"/>
Beskrivning för betalmetoden (kommer visas i kassan när betalmetoden väljs)	<input type="text"/> Om du väljer faktura från Resurs Bank så kommer din folkbokföringsadress att sättas till orders faktureringsadress (billing address).

▲ The text to be displayed under the payment method.

Nytt kort Magento SubFirecheckout

Betalningsstatusen är aktiv?	<input type="text" value="No"/>
Title	<input type="text" value="SubFire Nytt kort"/>
Sorteringsordning faktura	<input type="text"/> ▲ Vikt av valet. Lägre värde innebär högre placering.
Avgift för att använda betalningsalternativet	<input type="text"/>

In the sub-view, the settings look a little bit different. Now, you have checkboxes to the right in all forms. Uncheck those, that need to be different, like payment methods and representative data (including salt keys).

You should now disable all payment methods that do not belong to the representative and enable the rest, that you want to use. Don't forget salt keys, representative data, the environment and save!

Faktura privatperson Magento FireCheckout			
Betalningsstatusen är aktiv?	No	<input type="checkbox"/> Use Website [STORE VIEW]	<input checked="" type="checkbox"/> Use Website [STORE VIEW]
Title	Faktura privatperson FC	<input checked="" type="checkbox"/> Use Website [STORE VIEW]	<input checked="" type="checkbox"/> Use Website [STORE VIEW]
Sorteringsordning faktura		▲ Vikt av valet. Lägre värde innebär högre placering.	
Avgift för att använda betalningsalternativet		<input checked="" type="checkbox"/> Use Website [STORE VIEW]	
Beskrivning för betalningsalternativets avgift (kommer visas på orderraden)	Beskrivning för betalningsalternativets avgift	<input type="checkbox"/> Use Website [STORE VIEW]	
Beskrivning för betalmetoden (kommer visas i kassan när betalmetoden väljs)	Om du väljer faktura från Resurs Bank så kommer din folkbokföringsadress att sättas till orders faktureringsadress (billing address).	<input type="checkbox"/> Use Website [STORE VIEW]	
▲ The text to be displayed under the payment method.			
Nytt kort Magento SubFirecheckout			
Betalningsstatusen är aktiv?	Yes	<input type="checkbox"/> Use Website [STORE VIEW]	<input checked="" type="checkbox"/> Use Website [STORE VIEW]
Title	SubFire Nytt kort	<input checked="" type="checkbox"/> Use Website [STORE VIEW]	<input checked="" type="checkbox"/> Use Website [STORE VIEW]
Sorteringsordning faktura		▲ Vikt av valet. Lägre värde innebär högre placering.	
Avgift för att använda betalningsalternativet		<input checked="" type="checkbox"/> Use Website [STORE VIEW]	
Beskrivning för betalningsalternativets avgift (kommer visas på orderraden)		<input checked="" type="checkbox"/> Use Website [STORE VIEW]	

Resurs Bank Plugin for Magento: Coming soon

Deprecated

See [Magento Plugin 1.x - Changelog](#) instead where both released and unreleased updates are being made.

Magento Plugin Releases (Download)

Since we do have more than one Magento-plugin (depending on which Magento and flow you need), all plugin releases are documented in the primary space of [Magento modules](#).

Older releases of the deprecatedFlow plugin

As of january 2017, we have completely switched over to Bitbucket, but there is still an older archive available with prior releases of Magento 1x-deprecated flow located at <https://test.resurs.com/autodocs/magento-oldflow-archive/> if you really need to look at them.

Resurs Bank Magento 2.4+ Installation Instruction

Contents

- [System Requirements](#)
- [Packages](#)
 - [Core](#)
 - [Order Management](#)
 - [Simplified](#)
 - [RCO](#)
 - [Part Payment](#)
- [Installation Procedure](#)

System Requirements

Outside of Magento's own requirements you will need the PHP SOAP and JSON extensions. The iframe based checkout solution (RCO) will not function properly without SSL.

Packages

The module consists of five separate packages, which can all be installed at once through the metapackage **resursbank/magento-all**, or installed separately if preferred. Resurs Bank recommends you install the module using the supplied metapackage. Each of the packages are briefly described below.

Core

resursbank/magento-core

Basic functionality required by all other packages such as API libraries, payment method specifications and functionality to synchronize metadata from the API to your local database. Please note that this is the only required package, all other packages are optional but recommended.

Order Management

resursbank/magento-ordermanagement

Command integration against the API to capture, refund and void payments. Supports partial refunds / capture. Integrates remote callbacks to modify order status and perform other actions as the payment changes state. Includes a feature to track events related to the payment.

Simplified

resursbank/magento-simplified

Provides integration with Magento's native checkout process. This module enables payment methods from Resurs Bank to be displayed and utilized in the checkout process, it also enables a widget to fetch the customer address information from a remote service.

RCO

resursbank/magento-rco

Provides a custom single-page checkout solution based on an iframe for quicker conversions. This module replaces the native checkout page.

Part Payment

resursbank/magento-partpayment

Implements a widget on product pages to calculate estimated part payment prices

Installation Procedure

Download extensions from [Magento marketplace](#).

Assuming you are installing the metapackage as recommended, the complete installation procedure is as follows.

```
cd [magento_root_directory]
composer require resursbank/magento-all
bin/magento module:enable Resursbank_Core Resursbank_Simplified Resursbank_Rco Resursbank_Ordermanagement
Resursbank_Partpayment
bin/magento setup:upgrade
bin/magento setup:di:compile
bin/magento cache:flush
```

Open the administration panel and navigate to **Stores -> Configuration** followed by **Sales -> Payment Methods -> Other Payment Methods** and click on the **Open** button displayed on the **Resurs Bank** section.

The screenshot shows the Magento 2 Admin Configuration interface. On the left, there's a sidebar with various menu items like Dashboard, Sales, Catalog, Customers, Marketing, Content, Reports, Stores, System, and Find Partners & Extensions. The main area is titled 'Configuration' and has a 'Save Config' button in the top right. Under 'OTHER PAYPAL PAYMENT SOLUTIONS:', there's a link to 'OTHER PAYMENT METHODS'. Below that, there's a section for 'Resurs Bank' with a logo and a brief description: 'The easier and more flexible it is for your customers to shop, the more you sell. With Resurs checkout, you increase the possibility of more purchases and more satisfied customers.' A 'Close' button is next to the description. Under 'Payment Methods', there's a section for 'Resurs Bank API' which is currently selected. It contains fields for 'Checkout type' (set to 'OnePage iFrame based Resurs Checkout'), 'Environment' (set to 'Test'), 'Username' (empty), 'Password' (empty), 'Sync data' (with a note about syncing payment methods and annuity factors), 'Order Management' (set to 'Yes'), and 'Automatically sync data' (set to 'No').

Select your preferred **Checkout type** and supply your **Username** and **Password** for the API. Note that if you do not have these credentials you will need to contact Resurs Bank to have an account setup for you before you can proceed any further.

Save the configuration and then proceed by clicking on the **Sync data** button to sync payment methods and annuity factors from the API (annuity factors are metadata for the part payment package and will only be synchronized if you've installed said package).

When the synchronization process has completed your payment methods will be listed under the **Payment Methods** section as illustrated by the image below.



The easier and more flexible it is for your customers to shop, the more you sell. With Resurs checkout, you increase the possibility of more purchases and more satisfied customers.

[Close](#)

[Resurs Bank API](#)

[Payment Methods](#)

[Synced payment methods](#)

Name	Min total	Max total
Swish		
Faktura	SEK 1.00SEK	50,000.00
Delbetalning	SEK 1.00SEK	50,000.00
Bankkort Visa/Mastercard		
Kreditkort Visa/Mastercard		
Nytt revolverande kort	SEK 1.00SEK	50,000.00
Befintligt revolverande kort		
Företagsfaktura	SEK 10.00SEK	50,000.00
Befintligt revolverande konto		
Nytt revolverande konto	SEK 1.00SEK	50,000.00
Trustly		

Finally, expand the **Callback Settings** section (only available if you've installed the **Order Management** package) and click on the **Update** button to register callback URLs. We also recommended that you then click on the **Perform test** button to ensure that the API can reach your website. Please note that in order for any callbacks to function at all, including the test callback, your website **must be exposed to the Internet**, otherwise the API can't reach it.

Callback Settings

Callback registration Update

Test callbacks Perform test

Test last triggered at

Test last received at

No registered callbacks found. Update callbacks, if you don't get a list of "Registered Callbacks", please verify that the provided username and password are accurate.

Upgrading from the resursbank/checkout module

When upgrading from the resursbank/checkout module you should remove the old module before installing the new one.

You may encounter an SQL error when executing the `setup:upgrade` command, a constraint error caused by a duplicate foreign key in the `resursbank_checkout_account_method_annuity` table. This can be resolved by executing the following query:

```
alter table [DATABASE].resursbank_checkout_account_method_annuity drop foreign key [FOREIGN_KEY];
```

Replace `[DATABASE]` with the name of your database.

Replace `[FOREIGN_KEY]` with the name of the present foreign key as supplied by the error message (normally this would be `method_id`)."

You can then re-run the `setup:upgrade` command to proceed with the upgrade.

Please remember to always backup your database before executing any queries manually!

Resurs Bank Magento 2 CE 2.4+ payment gateway documentation

Contents

- Installation
- Settings
 - Resurs Bank API
 - Checkout type
 - Environment
 - Username
 - Password
 - Sync Data
 - Order management
 - Automatically sync data
 - Payment Methods
 - Callback Settings
 - Callback registration
 - Test callbacks
 - Test last triggered at
 - Test last received at
 - Part Payment Widget
 - Active
 - Payment method
 - Annuity
 - Advanced Settings
 - General
 - Logging
 - Log Level
 - Delete aborted orders
 - Round item tax
 - Onepage iFrame based Resurs Checkout
 - Alert customers when shipping methods update
 - Two step Magento Checkout with Resurs payment methods
 - Swish maximum order total
 - Wait for fraud control
 - Cancel automatically
 - Debit automatically
- Simplified Flow
- Resurs Bank Checkout (RCO)
- Part Payment widget
- Order management
 - Payment information
 - Payment History
 - Callbacks
 - AfterShop API integration
- Order confirmation email
- Problems, debugging and support
 - Orders
 - Discount tax
 - Contacting the support staff
 - Logs
 - Disable unnecessary modules
 - Callbacks
 - Payment methods
 - Information to include with problem reports to the support staff

Installation

Start by reading through the separate document detailing our [installation instructions](#).

This document also covers system requirements, a brief description of each individual module package and instructions for initial settings.

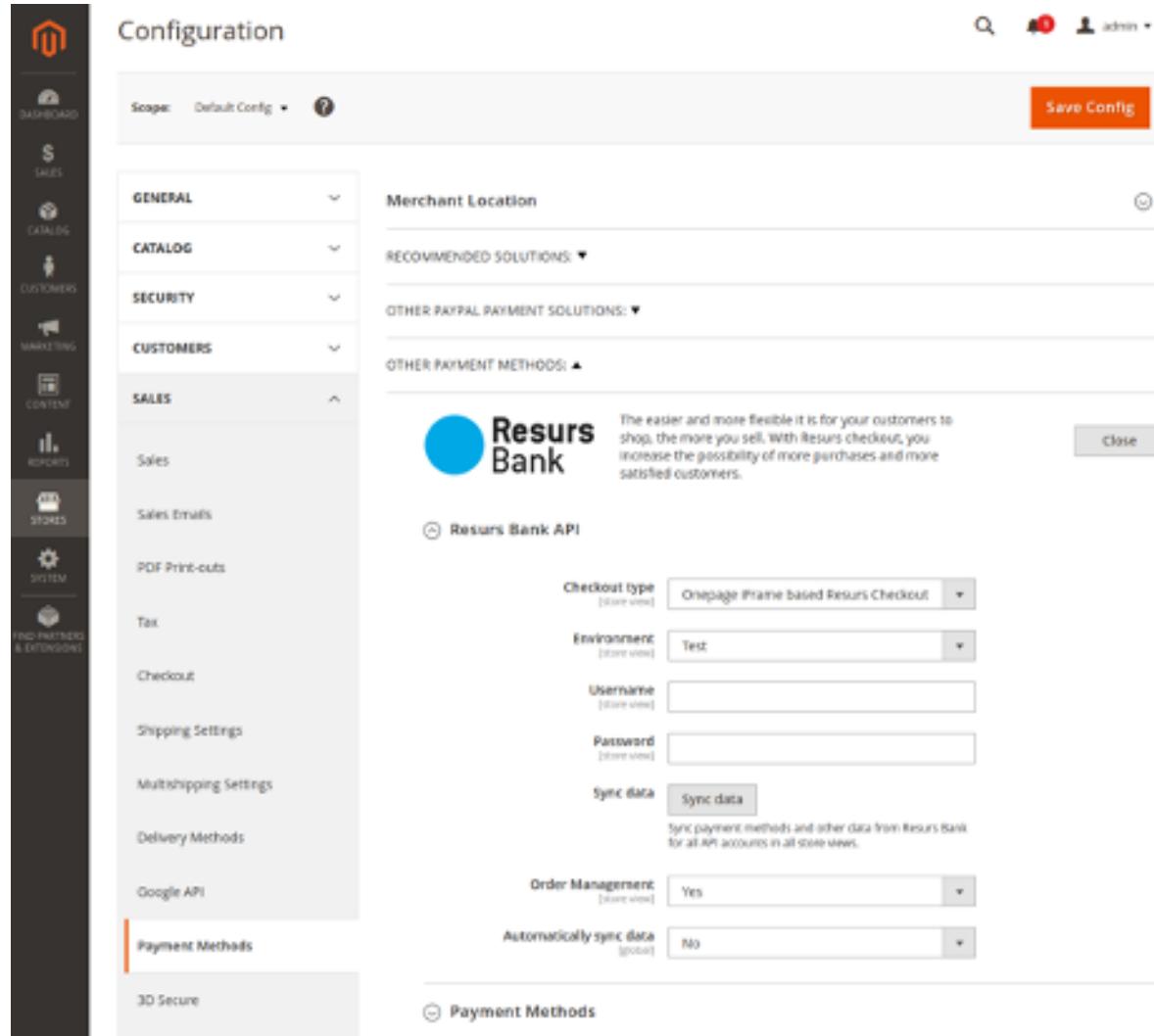
Please keep in mind that if you are missing any settings or features described in this document it's likely because your developer(s) have not installed the appropriate packages, all of which are freely available and can be installed by following the separate installation guide.

Settings

All our settings are located in the **Payment Methods** section of the configuration, under the subsection labeled **Other Payment Methods**. The various packages our module is composed of provide their own settings and may affect settings specified by other packages. For example, the **Simplified** and **RCO** modules will both append a **Checkout type** selection value to specify the respective API implementation. Below we will cover all settings provided by all packages. The illustrations are supplied from an environment which has all packages enabled.

Resurs Bank API

These settings are supplied by the **core** module and provide integration with the API.



The screenshot shows the Magento 2 Admin Configuration interface. The left sidebar menu is visible, showing various sections like Dashboard, Sales, Catalog, Customers, Marketing, Content, Reports, Status, System, and Find Partners & Extensions. The main content area is titled "Configuration" and shows the "Scope: Default Config". On the right, there is a "Save Config" button. The configuration page is divided into sections: "GENERAL", "CATALOG", "SECURITY", "CUSTOMERS", "SALES", and "PAYMENT METHODS". Under "SALES", "Sales", "Sales Emails", "PDF Print-outs", "Tax", "Checkout", "Shipping Settings", "Multishipping Settings", "Delivery Methods", "Google API", and "Payment Methods" are listed. "Payment Methods" is currently selected, indicated by an orange border. Under "Payment Methods", there is a "Merchant Location" section with "RECOMMENDED SOLUTIONS" and "OTHER PAYPAL PAYMENT SOLUTIONS". Below this is a large "Resurs Bank" logo with a blue circle. To the right of the logo, a descriptive text states: "The easier and more flexible it is for your customers to shop, the more you sell. With Resurs checkout, you increase the possibility of more purchases and more satisfied customers." A "Close" button is located next to this text. Further down, there are sections for "Resurs Bank API" and "Payment Methods". The "Resurs Bank API" section contains fields for "Checkout type" (set to "Onepage Iframe based Resurs Checkout"), "Environment" (set to "Test"), "Username" (empty), "Password" (empty), and a "Sync data" button. A tooltip for "Sync data" says: "Sync payment methods and other data from Resurs Bank for all API accounts in all store views." Below this are "Order Management" (set to "Yes") and "Automatically sync data" (set to "No"). The "Payment Methods" section is partially visible at the bottom.

Checkout type

Which API integration to utilize. The values are supplied by the **Simplified** and **RCO** modules, and thus one of these is required for a checkout integration. The various checkout types are described in more detail below in a later section.

Environment

A choice between **Test** and **Production**.

Username

API account username, supplied by Resurs Bank. There is a separate username field for each **Environment** value, we display the related one depending on your selection. This allows you to quickly swap between environments upon need.

Password

API account password, supplied by Resurs Bank. There is a separate password field for each **Environment** value, we display the related one depending on your selection. This allows you to quickly swap between environments upon need.

Please note that the value will become encrypted when you save the configuration.

Sync Data

When you click this button the module will:

1. Disable all payment methods recorded in the **resursbank_checkout_account_method** table.
2. Fetch payment methods from the API (and all associated metadata).
3. Record this data to the **resursbank_checkout_account_method** table.
4. Enable the payment methods that were available at Resurs Bank (leaving methods which no longer exist disabled).

Please note that:

- This process is executed once for each API account you have specified in your various websites / stores / store views.
- This process will also synchronize your part payment metadata if the **Part Payment** package has been installed.
- It's advisable to refresh your cache after this procedure has been completed.

Order management

Requires the Order Management module package (resursbank/magento-ordermanagement).

Automatically sync data

Whether or not to automatically execute the procedure to synchronize payment methods and their metadata at set intervals (defined by your cronjob). By default the module specifies to execute this procedure once a day at **00:00** (12 AM).

Payment Methods

This section contains a list of all **active** payment methods currently recorded in your local database. The list also details the **min** / **max** purchase value for each payment method.



The easier and more flexible it is for your customers to shop, the more you sell. With Resurs checkout, you increase the possibility of more purchases and more satisfied customers.

[Close](#)

ⓘ Resurs Bank API

ⓘ Payment Methods

Synced payment methods

Name	Min total	Max total
Swish		
Faktura	SEK 1.00SEK 50,000.00	
Delbetalning	SEK 1.00SEK 50,000.00	
Bankkort Visa/Mastercard		
Kreditkort Visa/Mastercard		
Nytt revolverande kort	SEK 1.00SEK 50,000.00	
Befintligt revolverande kort		
Företagsfaktura	SEK 10.00SEK 50,000.00	
Befintligt revolverande konto		
Nytt revolverande konto	SEK 1.00SEK 50,000.00	
Trustly		

Callback Settings

Requires the Order Management module package (`resursbank/magento-ordermanagement`).

This section provides buttons to register and test your callbacks. The section also details when a test was last dispatched / completed as well as a list of all registered callback URLs.

Please note that the illustration below is intentionally cleared of all data since some of the listed values should be considered private, but not sensitive.

Callback Settings

Callback registration

Update

Test callbacks

Perform test

Test last triggered at

Test last received at

No registered callbacks found. Update callbacks, if you don't get a list of "Registered Callbacks", please verify that the provided username and password are accurate.

Callback registration

Clicking the **update** button will execute a process to update the registered callback URLs at Resurs Bank for the configured API account. Note that unlike syncing data from the API this action will only affect the contextual API account and not all API accounts specified in your installation.

The module registers the following callback types (their functionality is described in detail in a later section of this document).

- UNFREEZE
- BOOKED
- UPDATE
- TEST

While there are more callback types available, UPDATE will be executed alongside them, and thus these other callbacks are considered needless overhead.

Test callbacks

Clicking this button will trigger an API call that forces Resurs Bank to execute the **TEST** callback. Essentially this means Resurs Bank will submit a request to the **TEST** callback registered for your configured API account.

Test last triggered at

Displays the time when you last initiated a test.

Test last received at

Displays the time when the last **TEST** callback URL was requested by Resurs Bank.

Part Payment Widget

Requires the Part Payment module package (`resursbank/magento-partpayment`).

This section contains settings affecting the behavior of the part payment widget. This widget itself is described in detail later in this document.

⌚ Part Payment Widget

Active [store view]	<input type="text" value="No"/> ▼
Payment method [store view]	<input type="text" value="Please select"/> ▼
Annuity [store view]	<input type="text" value="Please select"/> ▼
Annuity can only be selected if a payment method has first been selected.	

Active

Whether to display the widget on your product pages.

Payment method

Method to extract annuity factors from (these are listed in the **Annuity** setting described below).

Annuity

Annuity factor (payment interval) to utilize as a basis for part payment price calculations.

Advanced Settings

General

This section contains useful though not necessary settings which may be applied at will to suit your specific needs.

⌚ Advanced Settings

⌚ General

Logging [global]	<input type="text" value="Yes"/> ▼
Log Level [store view]	<input type="text" value="INFO"/> ▼
Delete aborted orders [store view]	<input type="text" value="Yes"/> ▼
Delete order when payment is aborted, to avoid orders with pending payment.	
Round item tax [store view]	<input type="text" value="No"/> ▼
Round item tax percentage to nearest integer. Only use this when your configuration results in floating tax percentage values, for instance when using percentage based shipping fees in combination with currency conversion.	

Logging

Whether or not to enable logging. There is a separate log for each package, and that package will only write information to its own log. The logs contain metadata, errors and any information relevant to debug problems.

The log files are named with the following scheme: **resursbank_[package].log** for example, **resursbank_core.log** or **resursbank_ordermanagement.log**

Some packages may produce additional log files. Log files are detailed in their own section later in this document.

We recommend you leave logging enabled but adjust the `Log Level` setting to fit your requirements (e.g. only log `ERROR` and `EXCEPTION` messages in production).

It is also advisable to rotate the log files regularly to avoid disk space issues developing over time, especially if `Log Level` is set to `INFO` or `DEBUG`.

Log Level

Sets the lowest level of message that will be logged. So for example if set to the default `INFO` level then `DEBUG` level messages will not be logged while `INFO`, `WARNING`, `ERROR` and `EXCEPTION` messages are logged while if set to `ERROR` no messages with a level of `DEBUG`, `INFO` and `WARNING` will be logged.

Delete aborted orders

Magento will produce an order prior to payments being completed, which may leave incomplete orders in your system if the payment fails for any reason. The natural events are as follows:

1. Magento creates the order.
2. Magento redirects the customer to Resurs Bank to fulfill their payment.
3. The client completes / fails to complete the payment (payment might for example fail if the customer lacks money in their account, enters a code incorrectly, or is otherwise not eligible for their choice of payment method).
4. The client is returned to Magento (either the success or failure page, depending on the outcome of the payment process).

In the event the client reaches the failure page the order will become canceled and the client will be redirected back to the checkout page where they are allowed to attempt payment once again. We automatically re-create their cart and supply all information we can based on their previous attempt (such as addresses, shipping method etc.).

Sometimes a client will try over and over to complete an order using a payment method they are not eligible for, as a result you may naturally end up with many canceled orders in your system.

By enabling this setting you can prevent this. We will delete the last order placed by the same client that wasn't completed / canceled, if any. This does of course not apply if the same client completes multiple orders in succession.

Round item tax

Depending on how you've configured your prices, taxes, products and shipping you may naturally end up with VAT values like **25.007%**

Resurs Banks API is very particular about what VAT values are allowed, and submitting an illegal value will cause an error resulting in a terminated purchase.

To avoid this, you can enable this setting which will round the values to the nearest integer.

Normally you should not need to use this setting, it's designed for complex setups utilizing irregular tax application procedures in combination with percentage based prices and discounts.

Note that the reason tax percentages appear accurate in Magento is due to how Magento rounds values in combination with its tax calculation, a procedure which is not matched by Resurs Banks API.

Onepage iFrame based Resurs Checkout

Requires the RCO module package (`resursbank/magento-rco`).

This section contains additional settings for the iframe based checkout solution (RCO).

Advanced Settings

General

Onepage iFrame based Resurs Checkout

Alert customers when shipping methods update
[store view]

No



Alert customers when shipping methods update

Shipping methods are not selected within the iframe itself, they are rendered in a separate list above the iframe on the checkout page.

Since customers often use a phone to place orders they will need to scroll through the checkout page, and since parameters such as address or method of payment may affect the list of available shipping methods, the list may update without the customer noticing (since they have scrolled passed the element), giving them an error as they attempt to place their order in case the shipping method they originally chose no longer is available when they attempt to complete their order.

By enabling this setting you will notify the customer when the list of shipping methods is updated, ensuring they do not miss important information.

Please note that this setting may annoy clients since it will produce a message every time the shipping method list is update. Therefore it should only be used if you've dynamic shipping methods that become available depending on parameters supplied through the iframe (such as delivery / billing address or method of payment).

Two step Magento Checkout with Resurs payment methods

Requires the Simplified Flow module package (resursbank/magento-simplified).

This section contains settings applicable to the native checkout procedure included in Magento.

Advanced Settings

General

Onepage iFrame based Resurs Checkout

Two step Magento Checkout with Resurs payment methods

Swish maximum order total [store view]

Maximum cart value allowed for Swish transactions. An empty value, or 0, will result in the utilisation of the value provided by the API.

Wait for fraud control [store view]

 Yes

Perform fraud controls before booking payments.
Consult Resurs Bank before activating.

Cancel automatically [store view]

 No

When the payment is assigned status FROZEN. Consult Resurs Bank before activating.

Debit automatically [store view]

 No

When the payment has been booked. Consult Resurs Bank before activating.

Swish maximum order total

The maximum cart value allowed to utilize the **Swish** payment method where applicable.

Wait for fraud control

Modifies the behavior of the procedure to handle payments at Resurs Bank. Simply put, by enabling this setting you ensure that *all* fraud checks and investigations are completed before accepting the payment. This grants higher security of payment, but it may also cause payments to take longer to process for natural reasons. Therefore it's advised this setting is only enabled after consulting Resurs Bank to discuss whether it's applicable for your specific circumstances.

Please note that while this setting is enabled in the illustration above it's actually disabled by default, it's been enabled in order to display the setting **Cancel automatically**, described below.

Cancel automatically

Cancel the payment at Resurs Bank automatically if it attains the status **FROZEN**. Please note that there are several legitimate reasons for a payment to become **FROZEN**, usually it means the payment requires further scrutiny to ensure it isn't fraudulent.

This setting requires **Wait for fraud control** to be activated (it won't be available otherwise).

Please note that this setting should only be applied after consultation with Resurs Bank.

Debit automatically

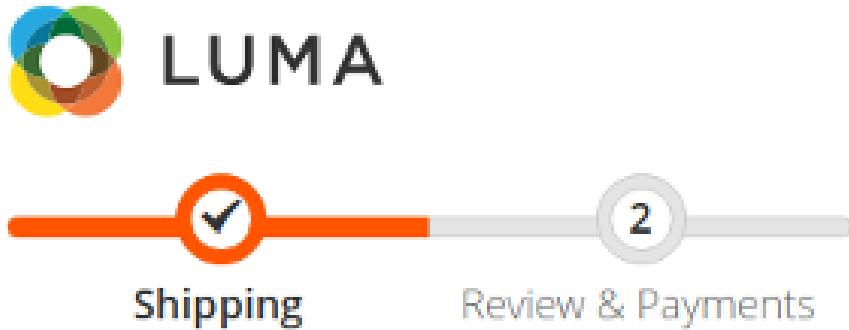
Whether or not to automatically debitize payments that have been accepted by Resurs Bank. Please note that this setting should only be applied after consulting Resurs Bank.

Simplified Flow

Requires the *Simplified Flow* module package (`resursbank/magento-simplified`).

Simplified Flow is one of the API integrations which you may utilize to communicate with Resurs Bank. This integration uses the native checkout process in Magento and you can thus combine this option with other payment solutions.

It prepends a widget to resolve customer address using their civic number or phone number (depending on country).



Shipping Address

Email Address *



You can create an account after checkout.

Private person Company

SSN (YYYYMMDDXXXX)

Fetch address

Phone number validation is also manipulated to make it match the expectation of the API. An inappropriately formatted phone number will cause the API requests, and ultimately the purchase, to fail.

Phone Number *

9999456456456



Please provide a valid phone number for your chosen country.

Shipping Methods

SEK 0.00

Free

Free Shipping

Payment methods function the same as for any other gateway.

Some methods require the customer to supply additional information (this depends on the method in question, and whether the customer is a company or private citizen). This is the case with invoices, as illustrated below.

The **Read More** link displayed with each payment method will open a dialog with additional information about the payment method.

LUMA Sign in

Shipping Review & Payments

Payment Method

Swish

Faktura

SSN (YYYYMMDDXXXX)
198001010001

[Read More](#)

My billing and shipping address are the same
Stella Eliasson
Makadamg 3
Göteborg, 416 55
Sweden
0708891234

[Place Order](#)

Order Summary

Cart Subtotal	SEK 70.00
Shipping	SEK 10.00
Tax	SEK 20.00
Order Total	SEK 100.00

2 Items In Cart

Ship To:

Stella Eliasson
Makadamg 3
Göteborg, 416 55
Sweden
0708891234

Shipping Method:

Flat Rate - Fixed

Delbetaling

Bankkort Visa/Mastercard

Kreditkort Visa/Mastercard

Resurs Bank Checkout (RCO)

Requires the RCO module package (`resursbank/magento-rco`).



Please note you must disable terms and conditions in Magento to utilize this checkout method.

Resurs Bank Checkout (shortened RCO) is an API integration which provides a custom iframe based checkout, replacing the native checkout process.

The screenshot shows the Resurs Bank Checkout (RCO) interface on a LUMA theme Magento store. The top navigation bar includes the LUMA logo and a 'Sign In' link. The main content area is divided into several sections:

- Select shipping method:** A table showing two options:
 - SEK 0.00 (Free) - Free Shipping
 - (checked) SEK 10.00 (Fixed) - Flat Rate
- Order Summary:** A table showing the breakdown of costs:

Cart Subtotal	SEK 70.00
Shipping Flat Rate - Fixed	SEK 10.00
Tax	SEK 20.00
Order Total	SEK 100.00
- Apply Discount Code:** A dropdown menu.
- Dina uppgifter:** Delivery information fields:

St... El... Mo... 3 ... 55 Gö... test@test.com 0708891234	198001010001 Inte du?
<input type="checkbox"/> Annan leveransadress	Ändra
- Betalsätt:** Payment method selection:

<input type="radio"/> Swish	
<input checked="" type="radio"/> Faktura Betala senare utan avgift	

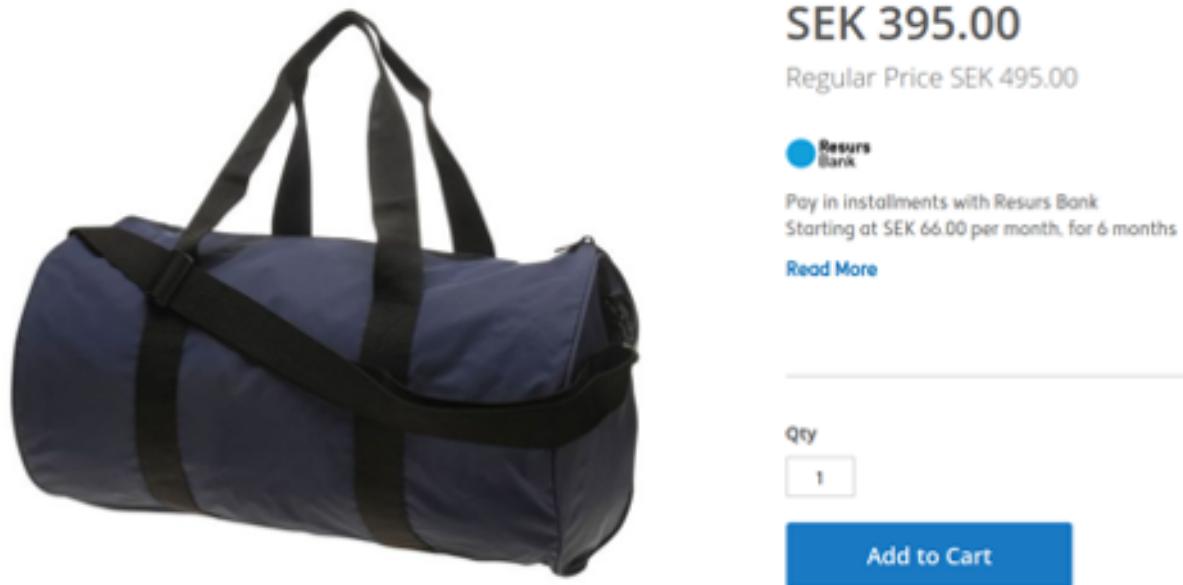
Få hem dina varor först, betala sen.
[Information och villkor](#)

Checkout agreements are supplied through Resurs Bank. **Disable** the native Magento checkout agreements can not coexist with RCO.

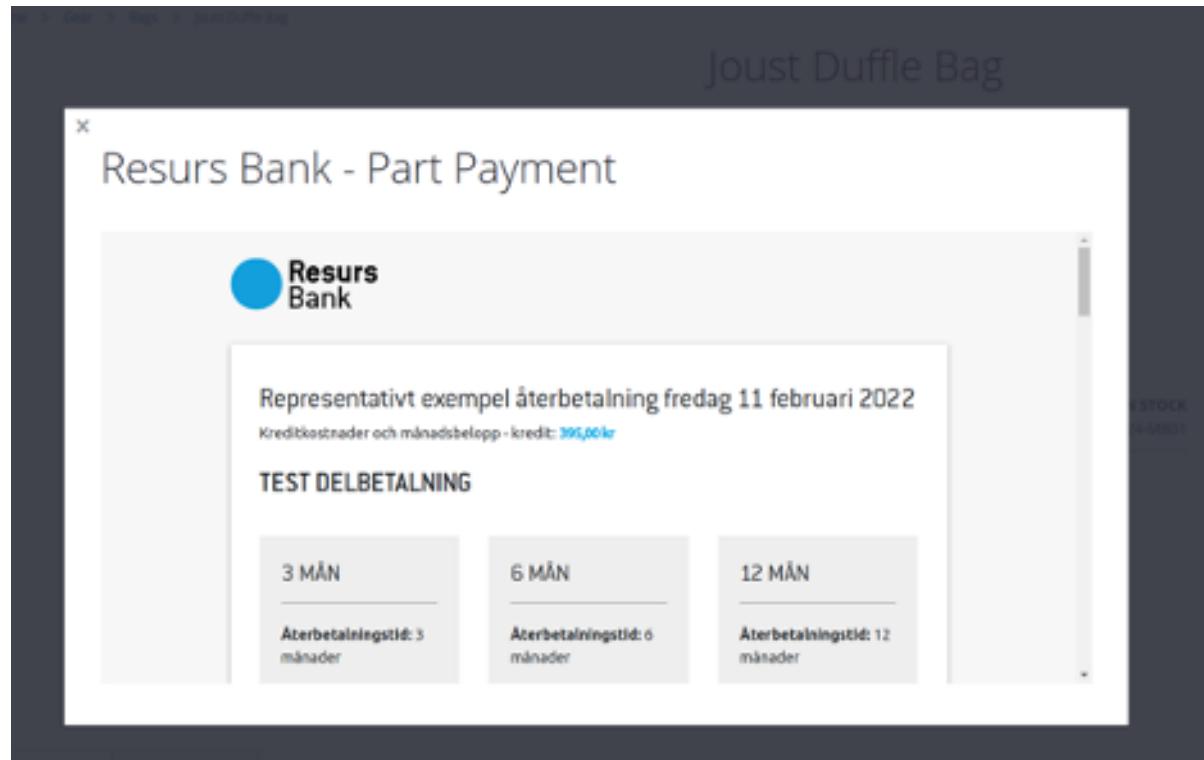
Part Payment widget

Requires the Part Payment module package (`resursbank/magento-partpayment`).

The widget is displayed on your product pages to reflect an estimated part payment price of the product. When it's displayed on a product page containing options that affect pricing (like configurable or bundled products), the price estimation is updated automatically as the customer modify their option selection.



Clicking the **Read More** link (illustrated in the image above) will open a dialog with information.

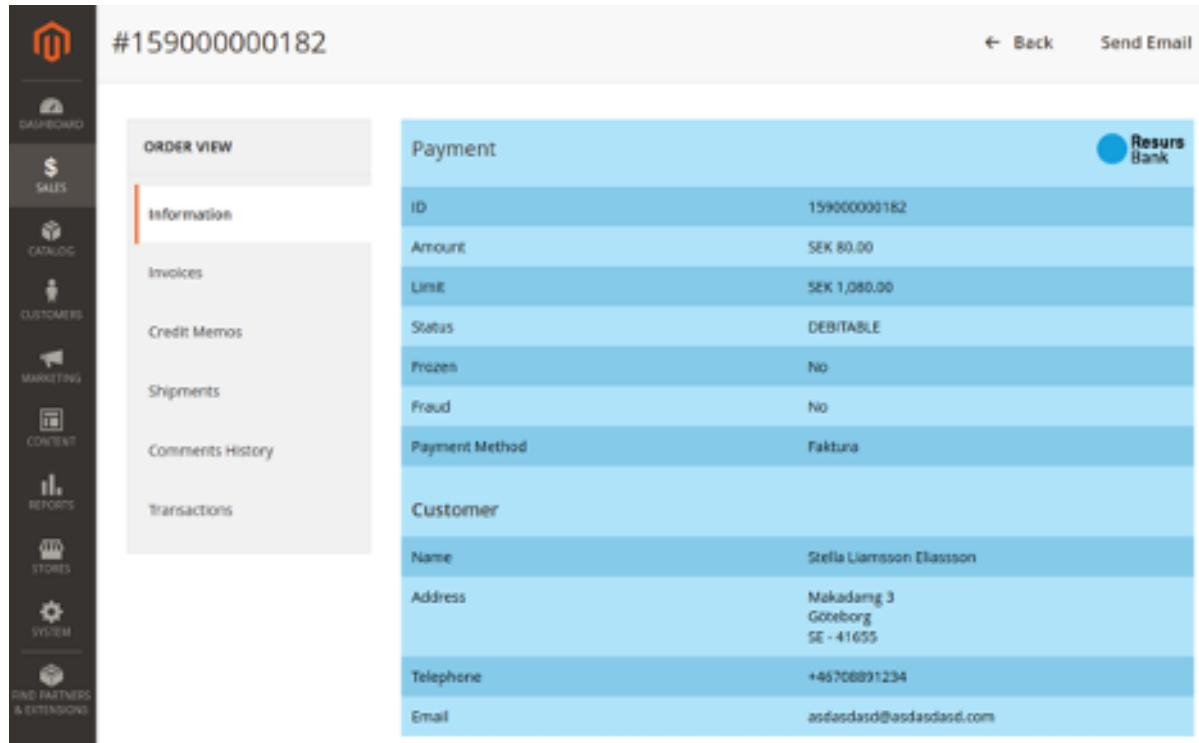


Order management

Requires the Order Management module package (resursbank/magento-ordermanagement).

Payment information

On the order, invoice and credit memo views, we append a block reflecting the current state of the payment at Resurs Bank.



The screenshot shows the 'Order View' page for an order with ID #159000000182. The left sidebar includes links for Dashboard, Sales, Catalog, Customers, Marketing, Content, Reports, Stores, System, and Find Partners & Extensions. The main content area has tabs for Information, Invoices, Credit Memos, Shipments, Comments History, and Transactions. The 'Information' tab is selected, displaying a 'Payment' section with the Resurs Bank logo. The payment details are as follows:

Payment	
ID	159000000182
Amount	SEK 80.00
Limit	SEK 1,080.00
Status	DEBITABLE
Frozen	No
Fraud	No
Payment Method	Faktura
Customer	
Name	Stella Liamsson Eliasson
Address	Makadam 3 Göteborg SE - 41655
Telephone	+46708891234
Email	asdasdasd@asdasdasd.com

Payment History

At the bottom of the order view you will find a button labeled **View Payment History**.

Submit Comment

Feb 7, 2022 10:10:05 AM | Pending Payment | Customer Not Notified

Authorized amount of SEK 80.00. Transaction ID:
"159000000182"

View Payment History

Clicking this button will open a dialog window displaying all the events that have occurred for the payment associated with the order.

#159000000182							
#159000000182 Payment History [Test]							
Event	Performed by	When	State From	State To	Status From	Status To	Additional information
Customer redirected to gateway.	Resurs Bank	2022-02-07 09:10:05					
Book signed payment API call initiated.	Resurs Bank	2022-02-07 09:10:20					
Book signed payment API call completed.	Resurs Bank	2022-02-07 09:10:21				FROZEN	
Client reached order success page.	Resurs Bank	2022-02-07 09:10:21	pending_payment	payment_review	pending_payment	resursbank_frozen	

Callbacks

Callbacks are essentially notifications submitted to your website from Resurs Bank whenever something happens to a payment, for example when it becomes debited or annulled.

The provided callback integration will update the state and status of the order in Magento to reflect the current state of the payment at Resurs Bank. It will also record the events in your logs and within the **payment history** (described above).

Note that if a payment is annulled at Resurs Bank the corresponding order in Magento will be canceled.

Callbacks are designed to be adaptable and your developers will easily be able to execute any number of custom actions whenever you receive these notifications without needing to modify the module itself.

AfterShop API integration

This API lets you manipulate payments at Resurs Bank. This module includes implementations to debit, credit and void payment at Resurs Bank when the corresponding action is performed in Magento. Please note the following.

- The module supports partial invoices / credit memos.
- If you wish to capture a credit memo remember you must create it from the **invoice view**, since using the button available on the order view will forcibly create an offline credit memo.
- If you edit an order the changes are not synchronized to the payment, you will need to modify it manually to match the changes you applied on the order.
- The actions performed against the payment are transcribed within **payment history** (described above).

Please note that this integration can be disabled in the configuration using the **Order Management** option in the **Resurs Bank API** section.

Order confirmation email

The **Order Management** module will affect how the order confirmation email is dispatched.

The module will prevent the email from being submitted during the checkout process (by setting the flag **can_send_new_email** to **false**). The module calls the method to submit the email when the callback **BOOKED** is received from Resurs Bank.

Under normal circumstances this callback will be submitted from Resurs Bank moments after the purchase has been accepted regardless of method of payment.

This may however be affected by the flags to modify the payment process (see the section **Advanced Settings - Two step Magento Checkout with Resurs payment methods** above for more details), or by settings applied on the payment methods at Resurs Bank.

When using **asynchronous** emails please be aware of the time limit imposed by Magento in **vendor/magento/module-sales/etc/di.xml**

Magento will dismiss order confirmations which have not been submitted within **24 hours** of order placement. This means that if the **BOOKED** callback would take longer than **24 hours** to be submitted the order confirmation emails would not be processed by Magento.

This is an unlikely scenario, but depending on your specific configuration and circumstances you may wish to increase this time limit. For your convenience we have included the code extract below which applies the limit explained above (from Magento 2.4.3).

```
<type name="Magento\Sales\Model\EmailSenderHandler">
    <arguments>
        <argument name="modifyStartFromDate" xsi:type="string">-1 day</argument>
    </arguments>
</type>
```

If your order confirmation emails do not appear to be submitting properly please review the information in the **debug** section below before initiating contact with our support team. As an emergency solution, you could disable the plugin responsible for preventing the original email submission: **vendor/resursbank/magento-ordermanagement/Plugin/Order/StopConfirmationEmail.php** (**resursbank_ordermanagement_stop_order_confirmation_email**) specified in **vendor/resursbank/magento-ordermanagement/etc/di.xml**

Problems, debugging and support

Orders

Before suspecting a problem with the module due to unexpected behavior with your orders you should consult the **payment history** (please see the **Order Management** section above).

This information will let you know what actions have been taken by the client and / or module. It will also detail when the order state and status have been modified by the module, and whether the module has canceled the order.

A few relevant notes regarding this information:

- If your order has been canceled, but it's not detailed in the payment history, then our module **did not** cancel the order. Remember that there are other systems which may cancel orders in Magento, for example the native cleaning process which automatically cancels orders that are left with the status **pending_payment** when they expire.
- If there is no information after the client was redirected to the gateway it's most likely because the client decided to close their browser window and thus not complete their purchase.
- If your order has obtained an unexpected state or status, but this change is not detailed in our payment history, then the module **did not apply** said state / status to your order directly.

Discount tax

There are multiple ways to define tax calculations in Magento and Resurs Bank supports most use cases. The only use case which isn't supported is a combination of **Apply customer tax** set to **After discount** in combination with **Catalog / Shipping Prices** set to **Excluding tax** (see the image below).

The screenshot shows the 'Configuration' section of the Magento Admin. On the left sidebar, under 'STORES', 'Tax' is selected. The main area displays 'Connection Settings' and 'Tax Classes'. A yellow callout box highlights the 'Tax Class for Shipping' dropdown set to 'Taxable Goods' and the 'Use system value' checkbox checked. Below it, 'Default Tax Class for Product' and 'Default Tax Class for Customer' are also set to 'Taxable Goods' with 'Use system value' checked. Under 'Calculation Settings', 'Tax Calculation Method Based On' is set to 'Total' with 'Use system value' checked. 'Tax Calculation Based On' is set to 'Shipping Address' with 'Use system value' checked. 'Catalog Prices' is set to 'Excluding Tax' with 'Use system value' checked. 'Shipping Prices' is set to 'Excluding Tax' with 'Use system value' checked. 'Apply Customer Tax' is set to 'After Discount' with 'Use system value' checked. 'Apply Discount On Prices' is set to 'Excluding Tax' with 'Use system value' checked. 'Apply Tax On' is set to 'Custom price if available' with 'Use system value' checked.

To understand why this doesn't work we must first consider that Resurs Bank will calculate all prices itself, using prices without VAT and a VAT percentage. Resurs Bank will calculate each row individually and then add them together. In the scenario illustrated above, Magento would subtract the discount **before** calculating the taxes, while Resurs Bank will do the opposite and there is no setting available at Resurs Bank to manipulate this behavior.

If you really must apply your settings in this way your only option is to create a plugin that modifies the data compiled for the payloads here: vendor/resursbank/magento-core/Model/Api/Payment/Converter/AbstractConverter.php :: getDiscountData()

Contacting the support staff

Before you contact our support staff please consider the following:

- Magento is a modular system and most setups have hundreds of modules and modifications to make the system fit your specific needs. Ask your developers if the problem you are experiencing could be caused by an incompatibility with another module or modification on your system. Even if we can apply a fix for it, knowing about it in advance saves a great deal of time for both parties, and if it's a customisation by your own development team we cannot provide a fix anyway since it's specific to your setup.
- While we certainly will help with an investigation if necessary it will take far more time compared to letting your own team handle it, because every setup is unique, and so is every server configuration. Only your own team has complete knowledge of the configuration for your server and

website, and they will likely have a local development copy of your website, letting them utilize tools to speed up the process which cannot be utilized by us since we would investigate the problem on your remote server.

- All of our modules are open source, and regardless of whether we can supply a solution to a problem within the module, your developers can apply a solution outside of it while investigating. This means you can have a solution implemented much faster while awaiting a potential patch from us. If no patch is made, for whatever reason, it's at least already fixed on your end.

For these reasons it's advisable you start by reporting problems to your development team and let them investigate the problem before they engage us. It's always better for us to discuss technical problems directly with a developer since it will speed up the process. The developer can then relay any relevant information back to you.

Below follows technical information for your developer(s) to assist them when researching problems, as well as guidelines for what information to include when contacting us to speed up the process.

Logs

For log files to be created you will need to enable the **Debug** setting in the module configuration (see the **Advanced Settings - General** described earlier in this document).

The module writes log files directly to **var/log** within the Magento directory, named according to the following scheme: **resursbank_[package_name].log**

For example, the **Simplified Flow (simplified)** module will create a log named **resursbank_simplified.log** containing all information produced specifically by the **Simplified Flow** module package.

Some packages will produce additional log files with information related to specific operations, at the time of writing the only implemented example of this can be found in the **Order Management** module which produces a separate log detailing incoming callbacks.

If you have the **Order Management** package installed we also track information about the payment process to the database (**resursbank_checkout_payment_history**) and you can later view this directly from the order view in the administration panel by clicking on the **View Payment History** button located at the bottom of the order view.

Remember to always consult your normal error logs since we do not / cannot log all exceptions, errors and other information directly in our own logs.

Disable unnecessary modules

A common mistake for Magento 2 developers is leaving modules enabled unknowingly if these include an option within their configuration to be disabled.

Remember that unless you disable a module using the Magento binary its files will still be loaded and can cause unexpected behavior. To properly disable a module, drop into your shell and execute the following commands:

```
cd [magento_root]
php bin/magento module:disable Vendor_Module
```

Callbacks

Callbacks are implemented through the **Order Management** package. These are URLs invoked by Resurs Bank whenever an event occurs for a payment. For example, when a payment is **booked** (accepted by Resurs Bank) the **BOOKED** callback URL will be triggered.

If you suspect callbacks aren't working properly for you please do the following:

- Check that your registered URLs are accurate through the corresponding section in the configuration (see **Callback Settings** for more information).
- Check the payment history using the button **View Payment History** from the order view. Every callback is tracked when it arrives and when it completes, so there should be two entries for each callback (please note that sometimes callbacks arrive at almost the same time, so two callbacks can technically be processed simultaneously).
- Check the **log** log file for detailed information about every incoming callback (if you suspect an error is occurring, please check your exception / system log using the timestamp from the payment history / callback log as reference).
- If no data exists to indicate callbacks are being received at all you can test that callbacks work using the **Test callbacks** button from the configuration.
- If the test callback doesn't seem to work either check with your server administrator that callbacks aren't being blocked by a firewall.

Whenever the module accepts a callback we manipulate the state / status of the order to match the status of the payment at Resurs Bank (please note you can see what state / status an order has obtained from any given callback through the **Payment History** feature described above).

Callbacks are meant to be adaptable, for example you might need to execute some action to manipulate data in an ERP system when something happens to the payment associated with an order. You can easily achieve this by creating plugins for **vendor/resursbank/magento-ordermanagement/Model/Callback.php :: unfreeze / update / booked**

Payment methods

Payment methods from Resurs Bank are dynamic, meaning that each separate API account has their own unique set of payment methods, each of which may specify a unique ruleset. This means great flexibility, but it also means that unlike most other gateways we cannot specify our payment methods

statically in an XML file. Instead, we fetch a list of all available payment methods and their metadata and store this information in your local database (our tables are prefixed `resursbank_`). We later inject our payment methods in places where payment methods are collected and utilized (see `vendor/resursbank/magento-core/Plugin/Payment/Helper/Data.php`).

It should be noted that our payment methods follow the new gateway design, specifying a command pool and value handlers in the `Core` and `Order Management` packages. The `Simplified Flow` package manipulates the `Authorize` command using a plugin (see `vendor/resursbank/magento-simplified/Plugin/Gateway/Command/Authorize.php`). Commands affecting the order post creation are defined in the `Order Management` package.

If payment methods do not appear to be functioning correctly you should first attempt to sync them and proceed by investigating the `resursbank_checkout_account_method` table. Please note the `activate` column, indicating whether a method is active and will appear in the checkout.

If you are using the iframe checkout (**RCO**) it's important to note that payment methods are listed by Resurs Bank and so they will be listed even if they aren't synced to your local database. In that event you will receive an exception when you attempt to place an order however, since Magento cannot find the payment method corresponding to the choice you made within the iframe.

Information to include with problem reports to the support staff

When reporting a problem to our support staff it's appreciated if you include the following.

- A description of the problem.
- Log files. Remove all log files from your log directory within Magento (var/log), reproduce the problem and then pack up the entire log directory in a tar / zip file. This is to avoid submitting us unrelated information which will needlessly slow us down.
- A guest account to your administration panel (alternatively screenshots of your configuration (both of the store view and default configuration)). If the problem concerns an order we would also appreciate screenshots from the order view of the payment information (the blue section at the top of the page) as well as the payment history (accessible through the **View Payment History** button at the bottom of the page).
- A list of modules you've installed that may affect your orders or checkout process (you may exclude modules which have been properly disabled using the Magento binary).

OpenCart

- Requirements
- Support & Help
- Installation
 - Downloading
 - Default method
 - Upgrading using default method
 - Installing and/or upgrading using developer mindset
- Payment method administration
- Order Totals Admin - Resurs Bank Fee
- Sales Order
- Get Address
- Log
- Not supported
 - Changing default checkout to Resurs Checkout
 - Usage
 - Adding shipping, gift cards etc on the checkout page
 - Giftvoucher

Requirements

- Working with localhost/127.0.0.1 or any local IP while testing will not work as Resurs Bank needs to be able to contact your OpenCart installation.
- You need an account with configured payment methods from Resurs Bank.
- The currency must match for current country, so for Sweden you must use SEK to enable the payment method in the cart. I.E. make sure that EUR is properly configured for the final order sum.
- SoapClient and curl must be installed on your server (i.e. ext-soap and ext-curl, etc).
- Your site has to be SSL/HTTPS ready.
- You need to have a valid SSL certificate for your OpenCart installation.

Support & Help

- Plugin intended to work with both Opencart 2.3+ and 3.x.
- Do not forget to configure the plugin with credentials, language and order statuses for callbacks.
- To activate the checkout, follow the instructions below - by editing necessary templates.
- The plugin are always shown as "disabled" in the admin panel, regardless of activation. This is cause by the way the checkout is implemented.

Installation

Downloading

The git repo used is [located here](#). You can either clone the repository, checkout a specific tag or simply download a specific version. Downloading specific versions [can be done from this location](#) where the latest tag should be considered the latest stable.

Default method

1. Place all of the files from the upload folder in it's correct folders of your opencart installation.
2. Log into your admin panel and go to Extensions and then enable the Resurs Checkout payment extension. Note: The section in the admin panel may look disabled, even though it is not.
3. Edit Resurs Checkout and fill in your environment (test or production).
4. Add yours credentials. Because Opencart plugin supports different credentials based on the language selected you need to add the language.
5. Click on the button "Update callbacks". This will then also list your registered callbacks for you to double check.
6. Add the different statuses that the order should be placed in depending on events from Resurs Bank.
7. When you hit Save everything should be setup for you and you could go to `yoursite.test/index.php?route=extension/payment/resurs_checkout_checkout`

Upgrading using default method

1. Like the first step, replace the files that you copied from the upload-folder the first time, with the new files.
2. For the system library files, you can replace "resurs-ecomphp" entirely with the new files from the upload directory.

Installing and/or upgrading using developer mindset

1. Place the repository in a separate folder. Make sure that your webserver can reach the files.
2. Run the `symlink.sh`-script, using first parameter as your site root (i.e. path to your siteroot path).
3. When you need to upgrade the plugin, you only have to replace the files in the "repository folder" (or do a git pull).
4. Follow the regular instructions above to continue installing the plugin.

Payment method administration

- Install the plugin.
- Setup what status that should be matched when an action occurs.
- Select the country you have an account for and enter the username and password.
- If you enter a correct username and password, payment methods should be loaded and shown as inactive payment methods.
- Enable the ones you want to be used. If you want to use a custom payment method text in the checkout, fill in "Custom name to be displayed as PaymentMethodName".
- A default image will be shown even if you don't enter any image URL. If you want to use a custom image, fill in the whole URL in the "Image url" and set the image size in "image width" and "image height".
- After you done this, go to order totals to fill in what you want to charge for the payment method.
- Set the sorting order of the Payment Method Resurs Bank, this will only sort them among the other payment methods, all the payment methods from Resurs Bank will come after each other and be sorted by the ID given from Resurs Bank in ascending order.
- Enable the Resurs Bank OpenCart-plugin by setting it to "Enable" and you should now see the payment methods in your shop's checkout phase.

Order Totals Admin - Resurs Bank Fee

- If you have correctly set up the "Payment method administration" there should be payment methods showing.
- Fill in how much you should charge for the payment method without any TAX, and select the corresponding TAX Class.
- Under "Invoice Line", fill in both what should be shown on the Invoice and on the Check Out line.

Sales Order

You can view the Resurs Bank payment flow for an order under history.

Get Address

Instructions about our getAdress-functionality is located in the getAddress folder, both how to enable it manually and how to enable it with VQmod. Only Sweden is supported.

Log

The Resurs Bank OpenCart-plugin logs to "System->Error Logs"

Not supported

- Multi store support for different countries

Changing default checkout to Resurs Checkout

In some installations you have to change the urls to Resurs Checkout manually. Files you want to change is listed below.

catalog/controller/common/cart.php

Change

```
$data['checkout'] = $this->url->link('checkout/checkout', '', true);
```

To

```
$data['checkout'] = $this->url->link('extension/payment/resurs_checkout', '', true);
```

catalog/controller/common/header.php

Change

```
$data['checkout'] = $this->url->link('checkout/checkout', '', true);
```

To

```
$data['checkout'] = $this->url->link('extension/payment/resurs_checkout', '', true);
```

catalog/controller/checkout/checkout.php

Change

```
$data['breadcrumbs'][] = array(
    'text' => $this->language->get('heading_title'),
    'href' => $this->url->link('checkout/checkout', '', true)
);
```

To

```
$data['breadcrumbs'][] = array(
    'text' => $this->language->get('heading_title'),
    'href' => $this->url->link('extension/payment/resurs_checkout', '', true)
);
```

catalog/controller/checkout/cart.php

Change

```
$data['checkout'] = $this->url->link('checkout/checkout', '', true);
```

To

```
$data['checkout'] = $this->url->link('extension/payment/resurs_checkout', '', true);
```

Installation of Partial Payments

The partial payments ("Pay X/month") has it's functionality in a different controller. You need to add the controller resurs-checkout-opencart/catalog/controller/product/resurs_partial_payment.php from this repository and also add the view resurs-checkout-opencart/catalog/view/theme/default/template/product/resurs_partial_payment.tpl to the correct locations on your website.

After you have added these files and activated partial payments in the admin area everything should be installed. What you now need is a snippet to place on the product details view where you want the partial payments to show. The snippet should be added somewhere in catalog/view/theme/default/product/prodcut.tpl on your website (where you want it to be shown).

The snippet is:

```
<?php if ($resurs_partial_payment) { ?>
<li><?php echo $resurs_partial_payment; ?></li>
<?php } ?>
```

And we need to make the \$resurs_partial_payment variable available on the product page so on the products controller (catalog/controller/product/product.php) you need to load in the partial payments controller as a data variable so it is available on the view (approx. line 475 in version 2.3.0.1).

```
$data['resurs_partial_payment'] = $this->load->controller('product/resurs_partial_payment');
```

That should be it.

Usage

Go to the Resurs Checkout by going to www.example.com/extension/payments/resurs_checkout or www.example.com/index.php?route=extension/payment/resurs_checkout depending on what type of Url structure is set up. This is the checkout that you should link all of your links like "Proceed to checkout". Important to notice is that you can't add shipping, gift cards or update your cart on the checkout page. This should be done on the cart page as a two-step checkout process.

Adding shipping, gift cards etc on the checkout page

Natively the plugin do not let you change anything about your order while on the checkout page. This page only shows an order summary and adds the Resurs Checkout iframe to capture the payment. If you want to add support for shipping method or other things that will affect the price you can make use of the function `updateCheckout` that is located in `upload\catalog\controller\extension\payment\resurs_checkout.php`. By calling this method with the reference that you have access to as `$reference` in `upload\catalog\view\theme\default\template\extension\resurs_checkout.tpl|twig` it will update the iframe to use the new price.

Giftvoucher

In your stores settings under the tab "Alternative" there is a section where you can select what status(es) an order has to be before the giftvoucher is valid. This means that the giftcard will not be valid until the order that purchased the giftcard is set to this status. For example the order may need to have status "Complete" before the giftcard will be valid. Once the order (giftcard) is valid you need to send out the email to the recipient of the giftcard. You do that within Gift Vouchers -> click to edit the voucher -> top-right corner is a "Send Email" button. Now the recipient of the gift voucher has received an email containing the giftvoucher code and the order that purchased the voucher is completed. This means that the user can now use the voucher in the checkout.

Changelog

Version 1.1

- Callback URL:s, spelling error.
- Updated getAddress to work for different templates.
- Works when only one payment method is returned from Resurs Bank
- Determines payment method based on specificType from webservice.
- Javascript when setting governmentId on payment method, fixed when government uses -
- Delivery Address corrected.
- Updated Regex for Sweden, to work with century , 19YYMMDD-XXXX and YYMMDD-XXXX
- Better error messages when some error occurs in webservice calls.

PrestaShop Payment Gateways

- [Prestashop Resurs Checkout](#)
- [PrestaShop SimplifiedShopFlow](#)

Internal errorhandling i Cart parts

To make sure we can separate "common errors" from the unknown exceptions we've added a bunch of error codes in the Cart.php-section of the Prestashop-RCO module. They are mostly self explained, but we're listing them as is below.

Code	Exception
800	No shipping address data in session.
801	Failed to resolve shipping address.
802	No payment session has been created.
803	Email missing in customer session data.
804	Failed to find order associated with cart <id>
805	Failed to load order with id <id>

Prestashop Resurs Checkout

- CUSTOMIZABLE PRODUCTS
- Compatibility
- Plugin versions
 - END OF LIFE (EOL)
- Requirements
- Upgrading
- Installation
 - Module configuration example
 - Registered payment methods and callbacks
- Known problems and (eventually) the solutions
 - Handling unfinished orders
 - SSN/Government ID's and direct debiting methods (like SWISH)
 - Going from cart to checkout has weird URL



Problem solving

There are a few known problems in the core functions of Prestashop, some of them are not related to our module at all. Take a look at the bottom at this page for more information.



Before installation

CUSTOMIZABLE PRODUCTS

In PrestaShop 1.7, the store offers customizable products (for example, you can add custom texts on your products; in the sample store, it is possible to add custom text to a mug).

This is not yet (june 2020) supported by the module for version 1.7.6.

This extension helps merchants to integrate Resurs Bank Payment Gateway to their PrestaShop based e-commerce stores. There are two versions of this plugin and they are currently only supporting Resurs Checkout.

Compatibility

This plugin is tested in "vanilla environments", which means that the plugin is installed with a default installed store. If there are plugins installed, they are usually installed from for example a marketplace. We can not fully guarantee functionality if you have other plugins installed, that uses overrides and such.

Plugin versions

Name	Tested with	PHP	Details	Country	Repos	Download	Dev Tracking	Status
Resurs Checkout Web - PrestaShop	1.6	>= 5.4	Resurs Checkout Docs	SE (Confirmed)	Master branch (1.6)	Release-tags for v1.6	JIRA for v1.6	NO LONGER SUPPORTED
Resurs Checkout Web - PrestaShop	1.7.6.x 1.7.7.x	>= 5.4	Resurs Checkout Docs	SE (Confirmed)	Master branch (1.7)	Release-tags for v1.7	JIRA for v1.7	MAINTENANCE
Resurs Simplified Flow (SOAP)	>= 1.7.7	>= 7.4	INSTALL WITH COMPOSER Using Simplified Flow Documentation	SE (Confirmed)	Core Simplified Addon Partpayment Widget Order Management Addon			NEWLY RELEASED

The above RCO-1.6/1.7 was splitted into separate modules when the development of the module for Prestashop v1.7 was initiated. The module for v1.6 went up to 2.6.5 with some updates that were brought into the v1.7-module - and the 1.7 module became 2.7.0

END OF LIFE (EOL)

The status flag above indicates how our support works and when there's eventually an End Of Life for each plugin. As for the maintenance flag, we tend to stop supporting PrestaShop 1.6 in the future as development for both PrestaShop 1.7.5 and 1.7.6 (april 2019) is moving forward. Leaving older deprecated versions is a way to maintain stability with fewer glitches than if we supported long range versions.

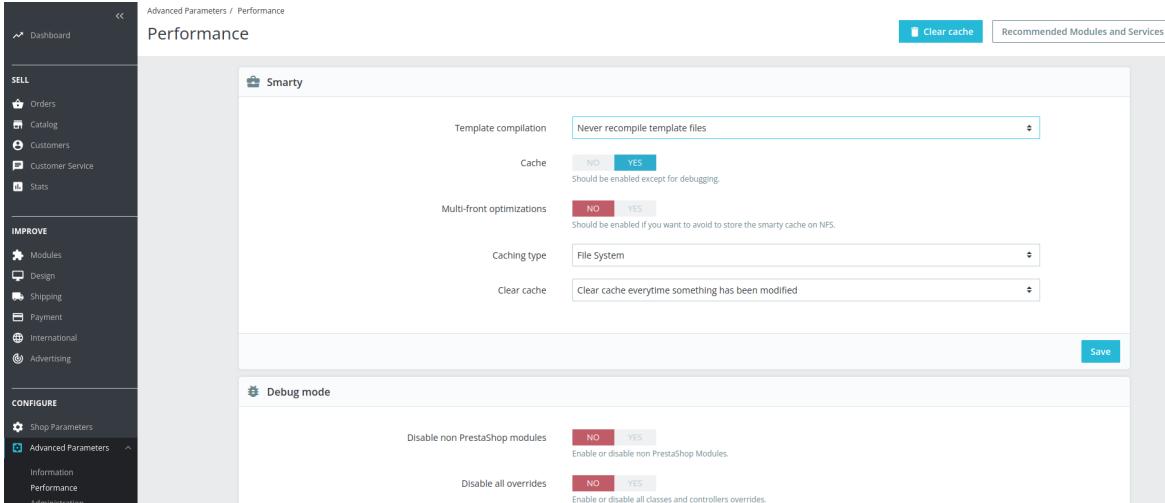
Requirements

- Supported platforms should follow the PrestaShop requirements. However, as of april 2019, none of those plugins supports PHP 5.3 anymore.

- Merchant credentials from Resurs Bank (ID and Password).
- Full SSL support for communicating with Resurs Bank.
- PHP module activated to support curl.
- PHP module activated to support soapclient and XML .

Upgrading

For each upgrade you do, make sure you clear all cache data in PrestaShop admin (Via the Advanced->Performance tab). As there are front end scripts that may require extra cleanups, this is also necessary.



Installation

The below instruction is based on a basic installation, where you have direct access to the server. By means, you probably have ssh access at the hosting provider or something similar. If you're using ftp, it's probably best to extract the zip (described below) at your local computer first - rename the folder, and then upload the file structure to the modules-directory at your shop.

There are other ways, which is in this case unsupported. There is also a "Prestashop Marketplace" where the plan is to get our modules distributed. When this part is fulfilled, it may be possible to install it directly from there. Unfortunately, as of april 2019, this is currently not possible yet.

1. **For both Prestashop 1.6 and 1.7, grab a tagged release** from our repos (**do not use the master-branch** as this usually is a development branch - and even if it's for the moment stable, we can't guarantee full production support at this level).
2. Unzip the downloaded file into <primary folder of your shop>/modules (the downloaded file may look like this: **resursbankplugins-resurs-checkout-prestashop16-ff8652e77e83.zip**)
3. Make sure you rename the primary directory that you unzipped to the modules directory.
If the zip looks like above (resursbankplugins-resurs-checkout-prestashop16-ff8652e77e83.zip), the directory that the module will land under may look like this: **resursbankplugins-resurs-checkout-prestashop16-ff8652e77e83**
4. Make sure your cloned or extracted directory under modules is renamed to **resursbankcheckout**, or the module won't be found by prestashop therefore not work.
5. Activate the module and configure it through the admin module panel. Do not forget to put the username and password in the credentials field.
6. **Also do not forget to register payment methods and callbacks (see screen dump below)!**
7. **Before going production mode, make sure you test the module so you know it covers your needs!**

Module configuration example

SETTINGS

Active	<input checked="" type="button"/> JA	<input type="button"/> NEJ
Debug Enabled	<input checked="" type="button"/> JA	<input type="button"/> NEJ
Store SSN	<input checked="" type="button"/> JA	<input type="button"/> NEJ
Store SSN on billing address.		
Production	<input checked="" type="button"/> JA	<input type="button"/> NEJ
Defines if the payment module is set to production mode		
Username for production	<input type="text"/>	
Password for production	<input type="password"/>	
Username for test/staging	ecomphppipelineTest	
Password for test/staging	<input type="password"/>	
Session lifetime (in seconds)	<input type="text"/>	

Registered payment methods and callbacks

BETALMETODER		
Titel	Aktivera/avaktivera "Betala från" i produktvisningen	Interval som används för att räkna ut delbetalningsalternativ (produktvisning)
Faktura privat		
Företagsfaktura		
Befintligt kort		
Nytt kort	<input checked="" type="checkbox"/>	
Delbetalning	<input checked="" type="checkbox"/>	24 Månader - 0% ränta
VISA/Mastercard		
Swishit		
<input type="button"/> Hänna tillgängliga betalmetoder		
REGISTRERADE CALLBACKS		
Successfully registered callbacks for environment: test		
ANNULMENT	http://presta16.tcte.loc/index.php?fc=module&module=resursbankcheckout&controller=callback&event=annulment&paymentId=(paymentId)&digest=(digest)&t=1554964538	
AUTOMATIC_FRAUD_CONTROL	http://presta16.tcte.loc/index.php?fc=module&module=resursbankcheckout&controller=callback&event=automatic_fraud_control&paymentId=(paymentId)&digest=(digest)&t=1554964538	
BOOKED	http://presta16.tcte.loc/index.php?fc=module&module=resursbankcheckout&controller=callback&event=booked&paymentId=(paymentId)&digest=(digest)&t=1554964538	
FINALIZATION	http://presta16.tcte.loc/index.php?fc=module&module=resursbankcheckout&controller=callback&event=finalization&paymentId=(paymentId)&digest=(digest)&t=1554964539	
TEST	http://presta16.tcte.loc/index.php?fc=module&module=resursbankcheckout&controller=callback&event=test&prm1=param1&prm2=param2&prm3=param3&prm4=param4&prm5=param5	
UNFREEZE	http://presta16.tcte.loc/index.php?fc=module&module=resursbankcheckout&controller=callback&event=unfreeze&paymentId=(paymentId)&digest=(digest)&t=1554964539	
UPDATE	http://presta16.tcte.loc/index.php?fc=module&module=resursbankcheckout&controller=callback&event=update&paymentId=(paymentId)&digest=(digest)&t=1554964539	
<input type="button"/> Registrera callback URL:er		

Known problems and (eventually) the solutions

Handling unfinished orders

Sometimes, customers decide to not fullfill their payments and shuts down their payment session/browsers faster than we can handle the cancellation. This means that some orders may look active, but in reality they are cancelled. In Prestashop, we have added an adapted shellscript written in PHP that is supposed to handle this kind of problems. It is not really supported, and is more to consider an extra addin-tool than something that really belongs to the plugin. It is located under **modules/resursbankcheckout/shell/Cleaner.php**, if you want to try it out. It is usually not recommended to install this to run under the Prestashop internal cron handler, but externally like a shellscript. For shelled-cronjob, the crontab should look like below and it is fine running it once per hour:

```
0 * * * * /bin/php /var/www/prestashop/modules/resursbankcheckout/shell/Cleaner.php
```

From /etc/cron.d it could also be added as a generic cronjob:

```
0 * * * * <owner> /bin/php /var/www/prestashop/modules/resursbankcheckout/shell/Cleaner.php
```

Do not forget to change the paths, binaries and the ownership for both above examples.

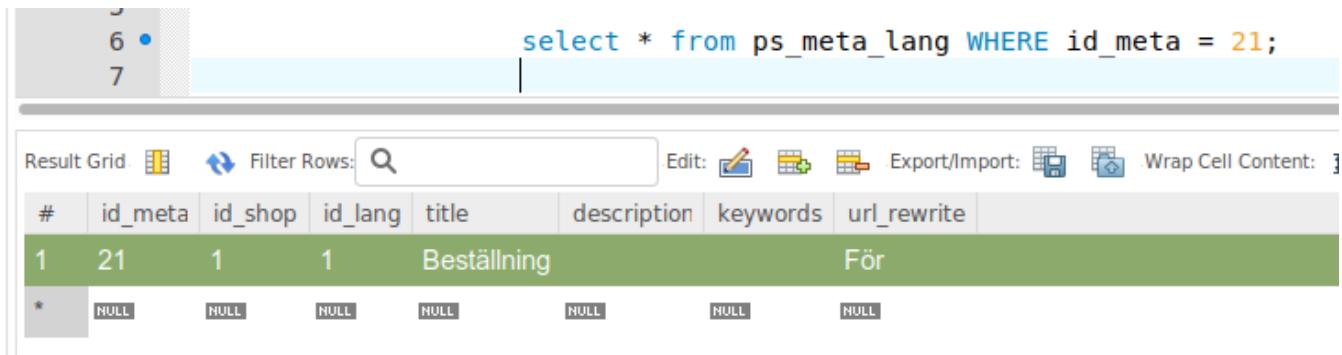
SSN/Government ID's and direct debiting methods (like SWISH)

If a customer chooses to change their billing/delivery address in the checkout without using the getPayment-features (meaning there may be some missing government id's through the checkout), Prestashop will be unable to catch this information on the payment. If you use features in the plugin that collects data for insurances, the module will be unable to store this data for you. By means, it may be "bad for business" especially insurance-features, if the customer somehow are not required to enter this information during the payment.

Going from cart to checkout has weird URL

 **PRESTA17-68** - (Not a bug) Problems with /För in cart-to-checkout-url (Documentation only) DONE

When you use swedish translations the url, depending on your platform, may show up as "<https://your-site/För>". The "För" link is a typo related to a redirect_url that went wrong when it was installed in the database. It looks ugly but is normally not destroying anything.



The screenshot shows a MySQL Workbench interface. A query is being run:

```
select * from ps_meta_lang WHERE id_meta = 21;
```

The results are displayed in a grid:

#	id_meta	id_shop	id_lang	title	description	keywords	url_rewrite
1	21	1	1	Beställning			För
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Solution:

If you want to report this as a bug to Resurs Bank - don't. We can't solve it from our side. Instead, update your database. If you are unsure of what to look for, the above query could probably be changed to something like this:

```
SELECT * FROM ps_meta_lang WHERE url_rewrite = 'För'
```

If you find a row that contains "För" in the url_rewrite-field, change it to "order". If you only have access to prestashop admin and the sql editor there, you could run something that looks like the query below.

Best practice is to look for the correct "wrong row" and change it, just so nothing else is changed by mistake.

```
UPDATE ps_meta_lang SET url_rewrite = 'order' WHERE url_rewrite = 'För'
```

Just make sure your tables are not prefixed as something else.

PrestaShop SimplifiedShopFlow

- System requirements
- Installing and uninstalling
 - Preferred Installation Method
 - Unsupported Installation Method
- Upgrading and other "self helping" support

Simplified ShopFlow for PrestaShop is intended to be built as a stand-alone checkout module for PrestaShop 1.7.7 and above. It is delivered partially with the help from composer, and it is highly preferred to install the module with composer. Please see below for instructions.

System requirements

- **PHP 7.3**
Note: Prior prestashop instances may try to enforce a lower requirement than this, so running the plugin with 7.1 will crash your site.
- SSL (preferably OpenSSL)
- **ext-soap**
XML (SoapClient)
- Curl or php streams for other communications, regarding rest etc.
- **Are you using nginx? Make sure that you configure the buffers/buffer_size to something appropriate (see below) or there may be trouble at some point.** We haven't seen this issue in apache environments (yet) but in case you see similar problems there, make sure that you use the same setup there.
- **Customer phone number is a mandatory field** for payments, so that field must be properly configured, so it cannot be ignored.

Installing and uninstalling

Modules can be installed / uninstalled / disabled / enabled separately. No module can be installed / enabled without Core being installed and enabled. Any action invoked on the Core module (such as installing it) will automatically be invoked on all other available Resurs Bank modules (such as Order Management, Part Payment or Simplified).

Preferred Installation Method

The preferred installation method is based on PrestaShop composer builds.

1. Go to your prestashop root folder.
2. Depending on your packages you want, run the following commands (depends on that you have composer installed!). The package psrbcore is required for all the others to work properly. Just keep in mind, that you can't just only install the core module.

```
composer require resursbank/psrbcore
composer require resursbank/psrbordermanagement
composer require resursbank/psrbssimplified
composer require resursbank/psrbpartpayment
3. Jump into the prestashop module catalog, choose to install the core module. The process is as described in this document very much automatic and will - if they are present - also activate psrbssimplified, psrbpartpayment and psrbordermanagement.
```

Unsupported Installation Method

This is the manual way, which is also stated by PrestaShop as kind-of-deprecated. As PrestaShop is moving forward to 1.7.8 and beyond they are also leaving manually unzipping modules, and instead using the same installation method as our above preferred. This method is unsupported and may not be guaranteed to work properly.

The method though is straight forward.

1. Start with downloading the bundle package from <https://bitbucket.org/resursbankplugins/psrbbundle> (git clone) or <https://bitbucket.org/resursbankplugins/psrbbundle/downloads/?tab=downloads> (download zip-package).
2. Unzip the package and make sure the modules- and vendor package is put into your prestashop root. In the end you should have [prestashoproot]/modules/psrbcore, etc and a [prestashoproot]/vendor/resursbank, etc.
3. Jump into the prestashop module catalog, choose to install the core module. The process is as described in this document very much automatic and will - if they are present - also activate psrbssimplified, psrbpartpayment and psrbordermanagement.

Upgrading and other "self helping" support

- When upgrading our module, make sure you really run the upgrade process (or reset, even if this requires you to reconfigure some parts of the modules). By a proper upgrade/reset procedure you will also get to register eventually new hooks and features that has been implemented since the last release. Without this, your newly upgraded plugin will run without all such features.

Developer Notes

- Forwarding specific errors to front end
- Customer Data Fields

Forwarding specific errors to front end

The below pull request may be a perfect example of how to forward specific errors to front page. This makes the plugin, instead of notifying customers about "something went wrong", render a proper error message on screen. The exceptions being made is API error, which is a bit more verbose than the standard problems.

<https://bitbucket.org/resursbankplugins/psrbssimplified/pull-requests/54>

The errors that we want to push forward to screen is the moment where customers forget or do not enter government id's on the checkout page for Resurs Bank internal payment methods, which is a requirement. This starts with [InvalidGovIdException on row 316](#). At this moment, we throw a specific exception that can later be identified as a "verbose point" in the module. We also throws an error code on each exception so the plugin will be able to handle all errors (if necessary) based on the code. However, for this case we only pick up the exception classname at [the OrderHandler from getProperOrderException](#) where we can define which of the exceptions that should be handled specifically eas "customer messages".

Customer Data Fields

As of [P17-226 - Få ordning på setCustomer](#) [DONE](#) we are changing the way customer data fields are behaving in the checkout. We usually don't have to collect both phone and mobile in the checkout steps, and as phone number is the absolute default in a newly installed PrestaShop store we use the internal phone-field as the primary target. [P17-223 - Do we really need 2 phone numbers on validation?](#) [DONE](#) and [P17-225 - RCO streamline phone numbers](#) [DONE](#) also includes requests for being more streamlined in the checkout, the same way as Resurs Checkout.

Instead of always fetching [all required fields](#), we instead try to centralize one of the two requirements by simply cloning the best value from phone and mobile:

- If phone is empty, use mobile and vice versa.
- If both fields are empty, initiate the checkout form with phone only - and ask the customer for this part.
- When running next step (validate.php), we will now set higher priority on the entered phone number in the final checkout step and use the one entered as both phone and mobile since Resurs still requires a phone number. Since shipping tracking is very common today, a majority of customers is also said to be using mobile numbers.

To properly handle phone numbers as mandatory, it is highly recommended to configure the fields in PrestaShop to be enforcing, not optional.

PrestaShop Simplified: Changes and updates

About the RCO module

The prior module that handled Resurs Checkout will over time be replaced with this new module and is therefore no longer maintained the same way it has been. There's a few things to think of if you plan to upgrade in the future.

Will government ID's be stored in the new module the way it was handled in the old module?

Yes, but there are changes in the prior behaviour.

According to [P17-169](#) - Hantering av personnummer (möjlighet att fånga dessa med en hook för försäkringsbolag) DONE, the same hook that handled the ability to store government id's in the platform is still there. It is mostly - as in the old version - not supported more than by an execution. The hook trigger itself looks like this and can be found in the rco-module too, the same way:

```
Hook::exec(
    'UpdateResursSsn',
    [
        'order' => $order,
        'vat_number' => $currentResursPayment->customer->governmentId,
    ]
);
```

In the simplified module, this trigger can be triggered multiple times, depending on the "way in". The trigger itself is placed in one method that has always a guarantee to run - `getSynchronizedAddress()`. This method is called from the moment when `bookPayment` gets "BOOKED" as response from Resurs and the moment after `bookSignedPayment` has been running. If both events happens to occur, there will also be two calls to the hook.

Note: Auto debited and finalized orders are only handled on the UPDATE-callback since there is high risk of race conditions when several callbacks are sent simultaneously and there's currently no solution for handle them in order (queue by cron for example), since this method is more or less a non-standard solution.

PrestaShop Simplified Described

The simplified shopflow is from an integration perspective much more friendlier to apply to PrestaShop than Resurs Checkout since the payment flow itself does not require much frontend editing. When credentials and payment methods are in place, it is very much straight forward and this page explains how. The described flow is based on a multi-step checkout.

- Initially (at least as guest) you will reach a checkout page where you fill in some of your own basic customer information. If you are logged in, you will instead of a form to fill in get a list of addresses to choose between. At this moment, the plugin is idle.

1 PERSONAL INFORMATION

Order as a guest | Sign in

Social title Mr. Mrs.

First name Only letters and the dot(.) character, followed by a space, are allowed.

Last name Only letters and the dot(.) character, followed by a space, are allowed.

Email

Create an account (optional)
And save time on your next order!

Password ***** Optional

Birthdate (E.g.: 05/31/1970) Optional

Receive offers from our partners

- Next page is a bigger form field where you are entering (and if you become a customer also saving) the customer data to use to confirm your order.

2 ADDRESSES

The selected address will be used both as your personal address (for invoice) and as delivery address.

Alias Optional

First name

Last name

Company Optional

VAT number Optional

Address

Address Complement Optional

Zip/Postal Code

City

Country ▼

Phone Optional

Use this address for invoice too

- As a third step you pick a preferred shipping method for where you want your products (could be different do digital orders).

4. At the final step the customer is now about to choose which payment method to use in the payment.

4 PAYMENT

<input type="radio"/> Bankkort Visa/Mastercard	<input type="radio"/> Part payment	<input checked="" type="radio"/> Invoice
Social security number	8305147715	
Phone number	+46701111111	
Mobile number	+46701111111	
E-mail address	tstre4323@resurs.it	
<input type="radio"/> Existing Resurs card <input type="radio"/> Kreditkort Visa/Mastercard <input type="radio"/> Swish <input type="radio"/> New Resurs card <input type="radio"/> Faktura Företag		
<input type="checkbox"/> I agree to the terms of service and will adhere to them unconditionally.		
<input type="button" value="PLACE ORDER"/>		

This is where the plugin activates at the first time. In psrbsimplified.php, a first hook named paymentOptions is executed (as hookPaymentOptions()). This method renders an array with the available payment methods in the checkout. The renderer is preparing everything necessary to proceed with the payment, like the payment method name, action link (validate.php, which very much translates to "validate cart to order"). It also renders additional information (through setAdditionalInformation() which is used by PaymentOption() in prestashop). This function renders the form data that is required at Resurs to fulfill the order. The final result of this is, as you can see in the screen dump a complete list of payment methods with the required form data for each method. There are different forms depending on which payment method you are using.

Placing the order

When customers are placing the order, it redirects the customer to controllers/front/validate.php, where the purchase process starts. From the above perspective, the validate.php-link is set in the PaymentOption() with set setAction()-method.

The first thing that happens in the validation process, is that we validate all data that is required for an order to be completed in PrestaShop. There are several exceptions in this part of the code, with different exception codes depending on what went wrong. If everything works fine in the postProcess-method, we will pick up the current order created and extract the order reference from it. This is the reference being used to create the order within Resurs simplified API.

Presta-Simplified: Frequently Discussed Issues and Important Notes

- Notes about nginx
- getApiFunctions()
- Resurs Bank plugin needs own statuses for pending/processing orders - why?
- When are the order/payment considered paid?
- Language and translations
- Commits and merges not taking effect in development mode
- Are we using session data at any point in the modules?

Notes about nginx

When the PrestaShop development started, we noted a few moments that nginx could cause errors unrelated to the plugin.

As stated in [P17-234](#) - Köp med "Faktura företag" får 502 Bad gateway. [DONE](#), requests to prestashop can sometimes render errors due to big headers:

upstream sent too big header while reading response header from upstream

This could be solved by adding similar rows to your server configuration:

```
fastcgi_buffers 16 16k;
fastcgi_buffer_size 32k;
```

getApiFunctions()

In (simplifiedshopflow)-Checkout.php, an implementation of the ecom instance is currently present. The method is not fully instantiating the ecom engine, but makes internal functions available that might be good to have. For example features to discover proper form fields, card/PSP type, annuityfactors etc. In its current state it is getApiFunctions is used to validate that a price lies within the min-max-value range for a payment method, but can be use for so much more. Especially if/when it gets centralized for example down to core (if possible).

Resurs Bank plugin needs own statuses for pending/processing orders - why?

Because we are mostly working with orders. This topic returns to us frequently when it comes to PrestaShop, since PrestaShop is said to work with *payment*, not *orders*. This also means that many of the default states used by PrestaShop is very much handling the payment automatically by default through as the "paid" value in the database is set to do this (see the screen dump below). At Resurs we're often working with unpaid orders until it is really paid, which means that the initial callbacks sent from Resurs Bank is **only** confirming the order status, not the payment itself. It is not until we get a true finalization signal from Resurs Bank (commonly via **UPDATE** or a **FINALIZATION**) we can actually confirm that the payment is completed.

This also makes the default states in PrestaShop, initially, entirely unusable for us. If a callback arrives and eventually forces the order to be pushed into a processing state, this also means that the payment will double for each time the plugin successfully does this (by means, it will look like the customer has overpaid the invoice). This behaviour is as of today expected, since the callback services sometimes is sending several callback the same second to the shop platforms, which will cause the race conditions that is doubling the invoice amount.

Best practice to avoid this is to simply **create separate order statuses** where the "paid" value is set to 0.

the

When are the order/payment considered paid?

When you choose an order state that is considering an order/payment as paid and finished, it is important that you choose the state based on what PrestaShop considers as a debited paid state. You should unless you have practical reasons not to do this, stay with the default settings used when the modules (order management) are installed. In our screen dump below we use "Payment accepted", which is a "paid state". The state itself will be triggered as soon as you use the capture feature in PrestaShop or in any other way makes Resurs finalize the order. For example, direct debiting payment methods like SWISH/VIPPS/etc are normally instantly finalized via the UPDATE callback this way.

 Order States

These setting are used to map the payment states in Resurs Bank with the order states in PrestaShop

Pending	Resurs Bank - Payment Review	▼
Processing	Resurs Bank - Confirmed	▼
Annulled	Canceled	▼
Completed	Payment accepted	▼
Credited	Refunded	▼

Save

Language and translations

[P17-222](#) and [P17-258](#) covers some key notes about how translations are automatically handled by our Ecom library instead of using local translations. There are also notes about how we used `currentLocale` in an early development state to decide which language that is considered as "correct" with checkout translations (I.E. "Read more", etc). Those translations are not centralized by the Prestashop language phrases, but instead using a "RCO Template" so the text in use can be centralized.

Commits and merges not taking effect in development mode

Make sure you do not cache anything. Caching and compilation of sources can be disabled from the performance section in the advanced admin configuration and is nearly a requirement for newly committed data to work properly.

Advanced Parameters / Performance

Performance

✓ All caches cleared successfully

 Smarty

Template compilation	Force compilation	▼
Cache	NO YES	
Should be enabled except for debugging.		

Save

Are we using session data at any point in the modules?

As of March 2022, the only place known to use a `session_start()` is during our own leaf-point features in the payments. With early knowledge from our prior RCO module we also know that `session_start()` generally can generate problems depending on how other modules are built. We used to start our own sessions by checking if it was started or not, before starting it. This however led us to conflicts with other modules, which is why we try to avoid using the session at all. The only exception occurs in parts of the payment/order creation phase where we want to catch Resurs Bank API problems in an early state.

If an error occurs before the redirect to our validation script takes over the process, all error messages caught in the `errors[]` variable will be lost, unless we re-initiate the session once more in the validate section. If errors occurred in the prior action, we can now forward them to the next section where either successful payments takes over, or the payment failure redirect. It is in the lastly mentioned functions we need the error messages at most.

Presta-Simplified and translations

The translation segment of PrestaShop is somehow complex. We are currently using ecomphp to fetch necessary translations for where they can be automated. The buildup of the phrases is however very complex, so you should take a look at the summary located here: [EComPHP and localization](#).

The summary is based on [P17-307](#) - Synka betalmetodernas texter del 3 [DONE](#) .

Status updates and callbacks

Orders are very much handled through [callbacks](#) from Resurs. The way callbacks works has less meaning as of today, even if PrestaShop has some exceptions for them. When Resurs is sending out a callback, the order management module will check for the current order status (via `getPayment`) instead of the callback type sent from Resurs and set a order state based on the response. The states are customizable from the administration panel (as long as you include order management as a module).

The screenshot shows a configuration page titled "Order States". A note at the top states: "These setting are used to map the payment states in Resurs Bank with the order states in PrestaShop". Below this, there is a table mapping five payment states from Resurs to PrestaShop order states:

Resurs State	PrestaShop Order State
Pending	Resurs Bank - Payment Review
Processing	Resurs Bank - Confirmed
Annulled	Canceled
Completed	Payment accepted
Credited	Refunded

A blue "Save" button is located at the bottom right of the form.

Each of the above states can freely be customized, but is always installed with preferred defaults. When a callback are received from Resurs, the library EComPHP will set an internal state integer, that is mapped to a state configured from PrestaShop.

WooCommerce

Table of contents

- [Where to find Merchant API Plugin \(Resurs Bank Payments for WooCommerce\)](#)
- [Version 2.2-series \(Revoked: Resurs Bank Payment Gateway for WooCommerce\)](#)

This extension helps merchants to integrate Resurs Bank Payment Gateway to their WordPress based e-commerce stores WooCommerce.

Where to find Merchant API Plugin (*Resurs Bank Payments for WooCommerce*)

Click here: [Resurs Merchant API 2.0 for WooCommerce](#) (Documentation and download).

Version 2.2-series (*Revoked: Resurs Bank Payment Gateway for WooCommerce*)

[Version 2.2.x has been moved to this page.](#)



This plugin has been replaced

Resurs Bank Payment Gateway for WooCommerce has been revoked from WordPress plugin registry. Do not use the old release - Payment flows in this plugin is deprecated at Resurs Bank.

Be aware that the older Resurs Checkout-API's (RCO Legacy/Facelift) are no longer supported in the merchant API - to keep using RCO Legacy, you need to stay on the 2.2 release.

0 decimals in WooCommerce

- Decimals in WooCommerce
- Why doesn't the payment gateways calculate as WooCommerce?
- Time for some math
- Order row quantity and coupons
- But I don't want to display decimals in my store – how do I do?
- I want to do this as a plugin - how would it look?

Under *General settings* in WooCommerce there is a section called *Currency options*. Here you can set the store currency and how the currency symbol should be positioned. There is also a setting that lets you choose how many decimal points you want prices to be displayed with.

At a first glance this setting only appear to be a display option. A setting that makes your prices look a little bit fancier. But to set the decimals to 0 can cause some real problems for you and your store.

Decimals in WooCommerce

When you set the number of decimals to 0 WooCommerce starts rounding product prices. Or to be a bit more specific – the ratio between the net price and the VAT. If you are using a payment gateway that sends each order row individually to the payment provider, or have an extension that connects to your accounting/erp system, then the calculation of order total might differ between the systems.

Why doesn't the payment gateways calculate as WooCommerce?

You might think that this is something we as extension developers should solve so that the figures added to the WooCommerce order is the same as the ones sent to the payment provider.

The problem is that when the price for a product is rounded and stored in the order it can be difficult to recreate the same kind of rounding in an external system. Let me give you an example.

Time for some math

A product has the price of 49 EUR including 25% VAT. This means a net price of 39.2 with a tax of 9.8 EUR. If you set the decimal points to 0, WooCommerce will round this to a net price of 39 and a tax of 10.

If we now imagine a payment provider who wants us to send each order row with the product price excluding tax plus the tax rate, this would mean 39 and 25%. But if you add 25% to 39 you will get a price including tax that equals 48,75.

In many cases when each order line is sent to the payment provider, order total is also sent as a parameter. When the sum of the order rows is compared to order total, these two figures don't match. Usually the payment provider answers with an error message and the purchase can't be finalized.

Order row quantity and coupons

In some cases it will work with this type of rounding. Instead problems might occur if the customer buys a number of items of a specific product. The payment provider usually wants the unit price plus the number of items purchased but WooCommerce does the rounding after the entire order row has been calculated. Another scenario that could cause similar problems is when using coupons.

But I don't want to display decimals in my store – how do I do?

Luckily there are ways to avoid displaying decimals as long as it's about prices with whole numbers (integers). Set the decimal points to 2 in the currency settings and then add this code snippet to your themes functions.php or in a separate plugin.

```
/**  
 * Trim zeros in price decimals  
 */  
add_filter( 'woocommerce_price_trim_zeros', '__return_true' );
```

With this code 49,00 will be displayed as 49. If you only have prices with whole numbers in your store and you're displaying prices including VAT, then it's only the VAT that actually will be displayed with 2 decimals. This results in a more correct calculation of prices and VAT in your store while you don't have to display unnecessary decimal points.

I want to do this as a plugin - how would it look?

This is how a plugin could look like. A file like this can be placed directly in wp-content/plugins, and then activated from the plugin manager.

Example <WP-root>/wp-content/plugins/woocommerce-price-trim-zeros.php

```
<?php
/*
Plugin Name: WooCommerce price trim zeros
*/

/**
 * Trim zeros in price decimals
 */
add_filter( 'woocommerce_price_trim_zeros', '__return_true' );
```

Download:



Figuring out remote ip for whitelisting

In test we sometimes need to whitelist your server ip address, for example when your server resides in a country that is not based in the nordic regions.

Normally it is not very hard to figure out which ip address that has to be whitelisted. There are many services you can use on the internet to fetch the proper address, or you could simply run a console base command like this from your server:

```
curl https://ipv4.netcurl.org/ip.php  
192.168.1.198
```

And your server's ip address will reveal instantly.



Feature is for v2.2-series only

The MAPI plugin don't have the described features below included.

Version 2.2-series

From plugin version 2.2.80 we've added a feature that handles this resolving for you. Go to your Resurs Bank main admin panel (the one where you enter your credentials) and scroll to the bottom of the page. You will see something like this:

Information om tillägg	
Plugin/Gateway	v2.2.80
PHP	v7.3.33-1+ubuntu18.04.1+deb.sury.org+1
EComPHP	EComPHP v1.3.69-20220126
curl driver	Tillgänglig v7.58.0
SoapClient	Tillgänglig
SSL/https (wrapper)	Tillgänglig OpenSSL 1.1.1 11 Sep 2018
Communication	NETCURL-v6.1.5
External ip and information	<button>Request Information</button>

As you can see, there's a button sayin "Request Information". Doing this will enable (unless there is javascript problems on your page, or your server has blocked access to the internet) a spinner:

External ip and information

Request Information

Shortly after this something like this will reveal in that section:

External ip and information

Request Information

ip: 123.123.221.221,
host: my.nifty.store.com,
SSL_PROTOCOL: TLSv1.2

The data that shows up on the screen is the data we need, to whitelist your ip address in our end.

Resurs Merchant API 2.0 for WooCommerce

- [Plugin basics and information](#)
- [Trouble shooting and error handling](#)
- [Installation from WordPress plugin repository](#)
- [Manually installing plugin](#)
- [Store configuration requirements](#)
- [Plugin configuration](#)
- [Order management](#)
- [Part payment widget](#)
- [MAPI Checkout Flow](#)
- [MAPI-WC - Customizations](#)

- [Requirements](#)
- [Upgrade information](#)
- [Download/install the plugin](#)
- [FAQ & Generic questions](#)
 - [Can I change the order number sequence?](#)

This section is reserved for the new WooCommerce plugin from Resurs bank.

Requirements

- **At least PHP 8.1**
- **At least** WooCommerce 7.6.0
- SSL-connectivity (preferably OpenSSL)
- CURL (ext-curl with necessary libraries) 7.61.0 or higher
- **Curl with CURLAUTH_BEARER-support**

If you run Ubuntu (bionic) the lowest available curl version will probably be 7.58.0 (focal is currently - aug22 - on 7.68) and in many systems bearers was introduced in 7.61.0 - however, you need to make sure it is really present in newer releases too.

Upgrade information

Are you installing this module over the prior release (2.2-series)? Make sure you uninstall the old plugin before installing this. We've seen conflicts between the both releases, for where the old plugin overrides parts of the new by stating that the CURLAUTH_BEARER is missing, even though the old wp-admin layout is still seen. This is very common when the both plugins are active side by side.

Download/install the plugin

Install the plugin via WordPress plugin repository (the plugin manager in wp-admin). It is NOT recommended to install the plugin manually since you will miss all automatic upgrades.

Url to the plugin itself is <https://wordpress.org/plugins/resurs-bank-payments-for-woocommerce/>

FAQ & Generic questions

Can I change the order number sequence?

Yes!

To update the order number sequence, update the database auto increment number like this:

UPDATE database

```
ALTER TABLE `wp_database` .`wp_posts` AUTO_INCREMENT = 200000000;
```

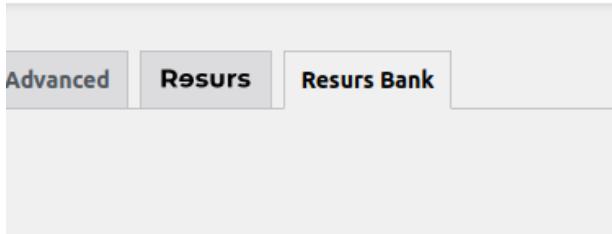
Change **wp-database** to your database name and set the **AUTO_INCREMENT** number to something that suits you.

Plugin basics and information

This is planned space for Resurs for WooCommerce MAPI Release v1.0. The MAPI-Commerce package is a rebuild of Resurs for WooCommerce that is adapted into the latest Resurs API (MAPIv2, as of 2022). All old flows has been disconnected from this plugin and since it will be a huge breaking change due to this among other features, it will be released separately from the old plugin.

- Old plugin vs new plugin (side by side/migration)
- Zero decimals in WooCommerce
- Changing the payment method configuration

Old plugin vs new plugin (side by side/migration)



As of the latest master branch it is possible to run both the new version and old version side by side. It is however **not** recommended to do so as both modules are fetching and handling payment methods differently. Best practice is to never run the plugins simultaneous, to avoid problems such as duplicate getAddress-fields and/or payment methods in the checkout. You can not only set the old plugin disabled by the "Enabled"-checkbox.

Resurs Bank payment gateway configuration

v2.2.104/PHP v8.1.2-1ubuntu2.11

Plugin and checkout

Enable checkout functions Enabled

To make the checkout features function properly, this setting must be enabled. Disabling this setting will shut down the entire plugin.

<input type="checkbox"/> Resurs Bank Payment Gateway for WooCommerce Deactivate Settings	Connects Resurs as a payment gateway for WooCommerce Version 2.2.104 By Resurs Bank AB View details
<input type="checkbox"/> Resurs Bank Payments for WooCommerce Deactivate Settings	Connect Resurs Bank as WooCommerce payment gateway. Version 1.0.0 Documentation

Zero decimals in WooCommerce

In newer installs of WooCommerce the setting for number of decimals to use in the checkout may be set to 0 as the default value. This is usually what you *do not want*, due to problems with roundings. If you are new to WooCommerce, make sure to look this up and change it if necessary. The recommended settings here is 2 decimals (**Resurs do not fully support more than 2**).

If you want to run with 0 decimals regardless of the warnings, you can [check out this page](#) for a proper solution.

General

Default customer location

Enable taxes Enable tax rates and calculations
Rates will be configurable and taxes will be calculated during checkout.

Enable coupons Enable the use of coupon codes
Coupons can be applied from the cart and checkout pages.
 Calculate coupon discounts sequentially
When applying multiple coupons, apply the first coupon to the full price and the second coupon to the discounted price and so on.

Currency options

The following options affect how prices are displayed on the frontend.

Currency

Currency position

Thousand separator

Decimal separator

Number of decimals

Changing the payment method configuration

When making any changes to payment methods at Resurs Bank it is strongly recommended that you clear the cache in the [Plugin configuration](#) to avoid potential problems.

Trouble shooting and error handling

From time to time, you'll need to handle problems with the plugin. Before you contact us with error reporting, make sure the below settings tab are correctly filled in. To avoid us to break your platform, the field for **log path** is by default empty. If your system can't write logs to the log path defined there, it may not work at all (which is why logging is disabled by default). Best practice here is to use you **path-to-wordpress/wp-content/uploads/wc-logs** which is where WooCommerce is normally writing their ownlogs. You can look at the example screen dump below.

By using the same path as your wc-logs directory, you will be able to look at the log files the same way that you browse WooCommerce logs.

The screenshot shows the 'Advanced' tab of the API Settings page. It includes fields for 'Log Enabled' (checked), 'Log Path' (set to '/var/www/wp3/wp-content/uploads/wc-logs'), 'Log Level' (set to 'INFO'), 'Enable Cache' (checked), and 'Enable Get Address Widget' (checked). A 'Save changes' button is at the bottom.

When logging is enabled with the above "practices", you will be able to see the logs inside the wordpress platform under WooCommerce logs section like this:

The screenshot shows the 'Logs' tab in the WordPress Admin. It displays a log entry titled 'ecom.log' from May 8, 2023, at 8:36 am GMT+0000. The log entry details a generic error for documentation, showing the stack trace from Resursbank\Woocommerce\Modules\Gateway\Resursbank->createPayment() to the main script.

```
2023-05-08T08:36:28+00:00 ERROR: Generic error for documentation., #0 /var/www/woocom3/wp-content/plugins/resursbank-woocommerce/src/Modules/Gateway/Resursbank.php(128): Resursbank\Woocommerce\Modules\Gateway\Resursbank->createPayment()
#1 /var/www/woocom3/wp-content/plugins/woocommerce/includes/class-wc-checkout.php(1050): Resursbank\Woocommerce\Modules\Gateway\Resursbank->process_payment()
#2 /var/www/woocom3/wp-content/plugins/woocommerce/includes/class-wc-checkout.php(1279): WC_Checkout->process_order_payment()
#3 /var/www/woocom3/wp-content/plugins/woocommerce/includes/class-wc-ajax.php(484): WC_Checkout->process_checkout()
#4 /var/www/woocom3/wp-includes/class-wp-hook.php(308): WP_AJAX->checkout()
#5 /var/www/woocom3/wp-includes/class-wp-hook.php(322): WP_Hook->apply_filters()
#6 /var/www/woocom3/wp-includes/plugin.php(517): WP_Hook->do_action()
#7 /var/www/woocom3/wp-includes/class-wc-ajax.php(96): do_action()
#8 /var/www/woocom3/wp-includes/class-wp-hook.php(308): WC_AJAX->do_wc_ajax()
#9 /var/www/woocom3/wp-includes/class-wp-hook.php(322): WP_Hook->apply_filters()
#10 /var/www/woocom3/wp-includes/plugin.php(517): WP_Hook->do_action()
#11 /var/www/woocom3/wp-includes/template-loader.php(13): do_action()
#12 /var/www/woocom3/wp-blog-header.php(19): require_once('...')
#13 /var/www/woocom3/index.php(17): require('...')
#14 {main}
```

Installation from WordPress plugin repository

This page contains information about how to install the plugin properly.

The official release is located at <https://www.wordpress.org/plugins/resurs-bank-payments-for-woocommerce/> and can be installed directly from the plugin installation system in WordPress.

The screenshot shows the WordPress Admin Dashboard. On the left, the sidebar menu is visible with the 'Plugins' item selected, indicated by a blue background and a red badge with the number '4'. The main content area is titled 'Add Plugins' with a 'Upload Plugin' button. A notice bar at the top recommends several plugins: 'Classic Widgets', 'Demo Installer', and 'Elementor Web'. It also mentions an update for 'WooCommerce'. Below this, there are tabs for 'Search Results', 'Featured', 'Popular', 'Recommended', and 'Favorites', with 'Search Results' currently selected. A plugin card for 'Resurs Bank Payments for WooCommerce' is displayed. The card features a circular logo divided into three segments (teal, yellow, orange), the plugin's name, a 'Update Now' button, and a 'More Details' link. Below the name, it says 'Resurs Bank Payment Gateway for WooCommerce.' and 'By RB-Tornevall'. Underneath the card, there is a rating section with five yellow stars and '(0)' reviews, a note that it has 'Less Than 10 Active Installations', and a compatibility status message: '✓ Compatible with your version of WordPress' followed by 'Last Updated: 6 mins ago'.

Manually installing plugin

If you plan to install a development version or not directly from WordPress plugin manager, this is the place to look at. If you plan to run this plugin in a "production state", please install it properly within the WordPress plugin manager. This page currently only has information for git installs.

- [Installations with zip file \(most similar to the WordPress Plugin Repository\)](#)
- [Installations with git](#)
 - [Dual plugins \(how to act on it\)](#)
 - [Creating your own zip-release from git](#)

Installations with zip file (most similar to the WordPress Plugin Repository)



About the current-zip file

Currently, there are no stable tag present in the repository. When there is a stable tag, the "current" file will always contain the most recent release. While waiting, the code structure in this zip is based on the master branch.

1. Upload the plugin ([resurs-bank-payment-gateway-for-woocommerce-current.zip](#) - 1.0.0, **last update 2023-04-27**) archive to the "/wp-content/plugins/" directory.
2. Activate the plugin through the "Plugins" menu in WordPress.
3. Configure the plugin via Resurs Bank control panel in admin.

Installations with git

Go to your WordPress plugin structure (normally located in **[WP-ROOT]/wp-content/plugins** and run this command:

```
git clone --recurse-submodules -j8 https://bitbucket.org/resursbankplugins/resursbank-woocommerce.git
```

The slug (path) used by this repository is not the proper name standard for the module. While writing this, the official slug is not entirely set yet, but there should be no problems installing it with another slug name than the default. However, if you use another path than the default, [be aware of the security issues that may come with this](#).

The above command will ensure you get all requirements installed in your structure, but primarily with the master branch installed. You should consider checking out a stable tag if you really need to use this install method. In this case you'll also need to check for updates manually and do manual checkouts.

This kind of installation neither guarantee that submodules are updated properly. In some cases, you'll need to update lib/ecom manually, with an extra git pull.

Dual plugins (how to act on it)

This is how it may look if you run the old version and just installed the new one. It is always recommended to deactivate the old one before activating the new one (and vice versa) before you start running them. The namespaces in the plugins are different to each other, so they should not crash the platform if you enable both of them in the same time.

<input type="checkbox"/> Resurs Bank Payment Gateway for WooCommerce Deactivate Settings	Connects Resurs as a payment gateway for WooCommerce Version 2.2.104 By Resurs Bank AB View details
<input type="checkbox"/> Resurs Bank Payments for WooCommerce Activate Delete	Connect Resurs Bank as WooCommerce payment gateway. Version 1.0.0

Creating your own zip-release from git

Clone the repository as shown above. Run a script similar like this. You need rsync and git installed. Observe that this is only a simplified example. The important thing is to make sure that the codebase has both the module and lib/ecom merged. The script below will do this, and also make sure that the latest ecom2-release is present. **Currently, ecom2 don't have any stable tag!**

Example script

```
#!/bin/bash

branch="master"

if [ "" = "$1" ] ; then
    branch="$1"
fi

src="resurs-bank-payments-for-woocommerce-bitbucket"
dest="zip/resurs-bank-payment-gateway-for-woocommerce"

if [ ! -d $dest ] ; then
    mkdir $dest
fi

echo "Synchronize $src with $dest"

rsync -a --info=progress1,progress2 --delete $src/ $dest
cd $dest
git checkout $branch
git pull

cd lib/ecom
git pull origin master

cd ../../

echo "Bundle by cleanup ..."
find . -name .gitignore -exec rm -v {} \; >/dev/null 2>&1
find . -type d -name .git -exec rm -rvf {} \; >/dev/null 2>&1

cd ..

echo "Archiving ..."

zip -r resurs-bank-payment-gateway-for-woocommerce.zip resurs-bank-payment-gateway-for-woocommerce
```

Store configuration requirements

Stock Keeping Unit (SKU)

In order for the order management functionality built into the plugin to work as intended all products sold in your shop need to have a SKU configured.

The setting for this can be found in the `Inventory` tab in the `Product data` box on each product.

Product data — Simple product

SKU: woo-hoodie-with-zipper

Manage stock? Manage stock level (quantity)

Stock status: In stock

Sold individually Limit purchases to 1 item per order

Number of decimals

By default WooCommerce is configured to show prices with zero decimals, for certain technical reasons this can occasionally cause rounding errors so we strongly recommend that you change this setting so to two decimals.

This setting can be changed by going to `WooCommerce Settings General` and scrolling to the `Currency options` section. It's called `Number of decimals` and should be at the very bottom of the section.

Currency options

The following options affect how prices are displayed on the frontend.

Currency: Swedish krona (kr)

Currency position: Right with space

Thousand separator: (empty)

Decimal separator: ,

Number of decimals: 2

Save changes

Plugin configuration

This page contains information on the configuration of the plugin.

Basic configuration

1. Go to WooCommerce Settings and click on the Resurs Bank tab
2. Enter your credentials in the API Settings tab
3. Choose the correct Store ID for your store from the dropdown
4. Save the settings

Detailed configuration information

API Settings

This tab is for basic connection settings and is where you enter your credential from Resurs Bank, whether to use the Production or Test API.

Allmänt Produkter Moms Frakt Betalningar Konton och integritet E-post Integration Avancerat **Resurs Bank**

[API inställningar](#) | [Betalmetoder](#) | [Delbetalning](#) | [Hantering av beställningar](#) | [Callbacks](#) | [Avancerat](#) | [Om plugin](#)

Aktiverad

Miljö

Klient-ID

Klienthemlighet

Butiks-ID

Spara ändringar

Payment Methods

This tab has no settings on it but allows you to see which payment methods have been configured for your account. Example:

Allmänt Produkter Moms Frakt Betalningar Konton och integritet E-post Integration Avancerat **Resurs Bank**

[API inställningar](#) | [Betalmetoder](#) | [Delbetalning](#) | [Hantering av beställningar](#) | [Callbacks](#) | [Avancerat](#) | [Om plugin](#)

Namn	Minsta totalbelopp	Högsta totalbelopp
Swish -	1.00	1,000,000.00
Revolverande kredit -	1.00	50,000.00
Delbetalning -	1.00	50,000.00
Faktura -	10.00	50,000.00
Faktura Företag	10.00	50,000.00
Bankkort Visa/Mastercard	0.01	1,000,000.00

Part Payment

Part payment widget

Order Management

Here you can enable/disable the different order management features of the plugin.

General Products Tax Shipping Payments Accounts & Privacy Emails Integration Advanced Resurs Bank

API Settings | Payment Methods | Part Payment | Order management | Advanced

Enable Capture Capture the Resurs Bank payment automatically when an order is completed.

Enable Cancel Cancel the Resurs Bank payment automatically when an order is cancelled.

Enable Refund Refund the Resurs Bank payment automatically when an order is refunded.

Enable Modify Reflect changes applied on an order to the corresponding payment at Resurs Bank.

Save changes

Callbacks

Callbacks are notifications sent to a specified URL when a payment reaches a certain status, such as authorization or rejection.

Allmänt Produkter Moms Frakt Betalningar Konton och integritet E-post Integration Avancerat Resurs Bank

API inställningar | Betalmetoder | Delbetalning | Hantering av beställningar | Callbacks | Avancerat | Om plugin

Testa callbacks

Test av callback inkom

Authorization Callback URL https://[REDACTED]/?wc-api=resursbank&callback=AUTHORIZATION

Management Callback URL https://[REDACTED]/?wc-api=resursbank&callback=MANAGEMENT

Spara ändringar

Advanced

This tab contains more advanced settings such as logging and cache settings. As a general rule it is recommended to keep the cache enabled as it significantly reduces the number of requests made to the API (and thus improves performance under most circumstances).

[API inställningar](#) | [Betalmetoder](#) | [Delbetalning](#) | [Hantering av beställningar](#) | [Callbacks](#) | [Avancerat](#) | [Om plugin](#)

Loggning aktiverad

Loggsökväg Lämna tomt för att inaktivera loggning.

Loggningsnivå Lägstanivå för loggning, meddelanden under denna nivå kommer ej att loggas.

Aktivera cache

Rensa cache

Aktivera hämta adress widget

Sort payment methods according to admin

API Timeout

Spara ändringar

Order management

The plugin has functionality built into it for automatically updating the payment at Resurs Bank when the order is manually edit/updated in WooCommerce order view.

By default all of these features are enabled when the plugin is enabled but can be disabled in the settings under WooCommerce → Settings → Resurs Bank → Order Management.

The screenshot shows the 'Resurs Bank' settings page in the WooCommerce admin. The 'Order management' tab is active. There are four sections with checkboxes:

- Enable Capture**: Capture the Resurs Bank payment automatically when an order is completed.
- Enable Cancel**: Cancel the Resurs Bank payment automatically when an order is cancelled.
- Enable Refund**: Refund the Resurs Bank payment automatically when an order is refunded.
- Enable Modify**: Reflect changes applied on an order to the corresponding payment at Resurs Bank.

A blue 'Save changes' button is at the bottom left.

Enable Capture

When this setting is enabled the payment will be automatically captured when the order status is set to Completed in WooCommerce.

Once an order has been set to Completed it can no longer be reverted to another status.

The module does not support partial captures.

An order which has been partially captured using the Merchant Portal can still have the remaining order amount captured from WooCommerce.

Enable Cancel

This feature works much like the `Enable Capture` setting in that it automatically cancels the payment when an order is cancelled in WooCommerce. Similarly, once an order has been cancelled it can't be reverted to another status but a payment that's been partially cancelled through the Merchant Portal can still have its remaining order lines cancelled through the order view.

Enable Refund

Like the `Enable Capture` and `Enable Cancel` features this feature will both automatically handle refunds of the payment when a refund is applied to the order in WooCommerce and once an order has been refunded it can't be reverted to another status and allow complete refunds of the payment when a partial refund has already been applied through the Merchant Portal.

Enable Modify

With this feature enabled changes you make to an order will be reflected on the payment.

However, there are some caveats. First of all, this only works if the payment can be captured, cancelled or if it has been fully cancelled. Furthermore, if the sum total for the changed order are equal to what they were before changing the order no call will be made to Resurs Bank. E.g. if you substitute one product for another with the exact same price then the change will not be transmitted to the payment.

One workaround for this issue is to cancel the payment through the Merchant Portal before editing the order in WooCommerce as this changes the authorized amount on the order to 0.

It should also be noted that it's not possible to exceed the credit limit which has been authorized for the payment. Instead the customer should place a new order.

Should the modify action fail it can sometimes result in the payment being cancelled despite the order not being cancelled. If this happens you can click the `Recalculate` button to update the order state as this also pushes the order update to the payment at Resurs.



Please also note that changes made from Resurs Merchant Portal will not reflect back to the WooCommerce system and have to be made there as well.

Part payment widget

This feature of the plugin allows your store to display a small widget on individual product pages with information about available part payment options including a modal iframe popup with more detailed information.

[Home / Accessories / Cap](#)

A modal window titled "Resurs Bank" showing representative examples of repayment plans. It includes a logo and a heading "TEST DELBETALNING". The modal is divided into six sections: "3 MÅN", "6 MÅN", "12 MÅN", "24 MÅN", "36 MÅN", and "48 MÅN". Each section displays the following details:

Aterbetalningstid	Ordinarie månadsbelopp	Kreditbelopp	Kreditränta	Uppläggningsavgift	Administrationsavgift / månad	Effektiv ränta	Kreditkostnad	Totalt belopp att betala
3 månader	161,00 kr *	395,00 kr	0,00 %	0,00 kr **	29,00 kr	1,00 %	88,00 kr	483,00 kr
6 månader	95,00 kr *	395,00 kr	0,00 %	0,00 kr **	29,00 kr	1,00 %	175,00 kr	570,00 kr
12 månader	62,00 kr *	395,00 kr	0,00 %	0,00 kr **	29,00 kr	1,00 %	349,00 kr	744,00 kr
24 månader	45,00 kr *	395,00 kr	0,00 %	0,00 kr **	29,00 kr	2,00 %	688,00 kr	1 080,00 kr
36 månader	40,00 kr *	395,00 kr	9,70 %	0,00 kr **	29,00 kr	3,00 %	1 045,00 kr	1 440,00 kr
48 månader	40,00 kr *	395,00 kr	16,72 %	0,00 kr **	29,00 kr	4,00 %	1 525,00 kr	1 920,00 kr

The modal also features a "Related products" section at the bottom.

Configuration

The widget configuration options can be found on the **Part payment tab** under **WooCommerce Settings Resurs Bank**.

Before configuring the widget you need to set the global plugin configuration on the [API Settings](#) tab.

The screenshot shows a configuration interface for a part payment widget. It includes fields for enabling the widget, selecting a payment method, choosing an annuity period, setting a limit, and a save changes button.

Part payment widget enabled	<input checked="" type="checkbox"/> Enabled
Payment method	Resurs konto (merchant_api)
Annuity period	3 Månader - 0% ränta
Limit	700

Save changes

Part payment widget enabled

Toggle this setting to enable or disable the widget.

Payment method

This option allows you to choose which of your payment method to use for the widget.

Only payment methods that support part payment will be available here.

Annuity period

Controls how long of a payment period to calculate the monthly cost for.

Like with the payment method setting only supported periods will be shown.

Limit

Sets a lower monthly installment limit under which the widget will not be displayed.

If you to set the limit higher than the payment method's maximum configured purchase price you will see a warning message after saving your settings.

You should set this value high enough that the monthly cost is at least SEK 150 (Sweden) or EUR 15 (Finland).

MAPI Checkout Flow

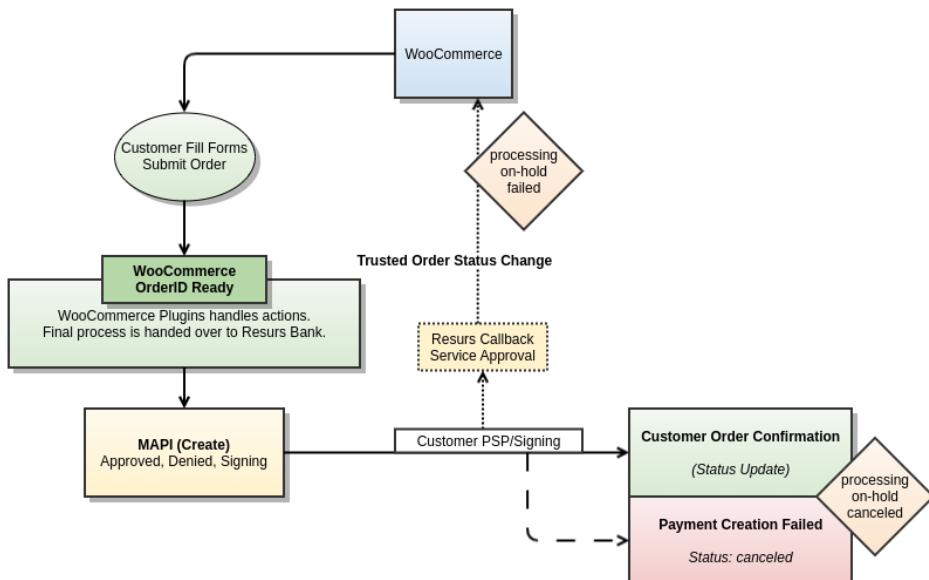
Reserved space for the described MAPI-flow. It very much looks like the prior flow, with a few enhancements.

Purchasing with the new Merchant API.

The Merchant API is, like the former simplified shopflow, made to go with a platform default rules but with some improvements. The WooCommerce order will (as before) be created first (in *Pending* status), then Resurs order is placed and finally - depending on the result - updated to *Processing*. The procedure looks like this:

1. Purchase button is clicked. WooCommerce starts handling the order, setting the first *Pending* status.
2. When WooCommerce checks (stock count, etc) are finished, Resurs plugin takes over to create the payment. During this process, signing the payment also occurs.
3. When Resurs is done, the order can be either accepted (*processing*), rejected (*failed*) or paused (*on-hold*). Depending on the outcome, Resurs will redirect customers back, either to the final order received-page ("thank you") or a failure-page (if Resurs rejects the order, customers redirected to a fail-page will render the order to be *canceled*).
4. Side by side with the above steps, callbacks are also sent from Resurs, in case that the customer fails to return to the thank you page. This makes sure that the statuses are properly set.

Visualized Flow



How callbacks works

1. When a purchase is completed and the payment is ready to be handled at Resurs side a few callbacks are executed synchronously from Resurs. You can see the flow and more information [about the callbacks here](#).
2. To make sure that the callback received from Resurs is correct, the plugin will make a secure request with Resurs to confirm the order status, before handling the order locally at WooCommerce side.
3. When an order is synchronized with Resurs, the order status will update to either *processing*, *on-hold* or *failed*. If the order is set to *on-hold*, this means that more actions could follow. For example, an order at Resurs can be temporarily frozen (which is the cause of *on-hold*) and will be pushed further (to *processing*) when ready to handle.

MAPI-WC - Customizations

This section contains a set of examples of how you can code with the module on your own. Much of the codebase is modular which means that you practically can use the same hooks as we do. Please be aware of the fact that our module follows a lot of standard practices, so some things may not be supported or compatible with your platform if you have own customizations.

The methods and examples on this page is not actually supported by Resurs and reside here only to be initial examples of other ways of coding.

- [Custom customer type handling](#)

Custom customer type handling

Normally, in at least Sweden, we use a widget called getAddress that we pick up customer information based on the customer type (private person - NATURAL, or company - LEGAL). If you decide to not use it but still need to update payment methods based on other sections that picks up customer types, you can use the hooks that we use with the standard getAddress requests. The trigger will prepare internal session values to change the customer type so that the payment methods can update with correct data.

The urls generated in this example very much follows the standards of WordPress' permalinks, so if you can you would want to look for the frontend variable set called **rbCustomerTypeData**. In our basic front end we usually look up the customer type by what's entered in the company name (in the billing forms). Like this:

```
rbCustomerTypeData['apiUrl'] + '&customerType=' + (rbIsCompany() ? 'LEGAL' : 'NATURAL'),
```

If you know how to set up frontend-hooks by binding html-elements you can easily bind your own request like this:

```
// Code to bind a function, for example radio button triggers goes here.

// 
// In your header with scripts or similar put up function like the one below.
// Just make sure you can access rbCustomerTypeData.
// The URL usually looks like this:
// /?resursbank=set-customer-type&customerType=NATURAL
//

function customizedRbUpdateCustomerType(setCustomerTypeValue) {
    jQuery.ajax(
        {
            url: rbCustomerTypeData['apiUrl'] + '&customerType=' + (setCustomerTypeValue === 'LEGAL' ? 'LEGAL'
: 'NATURAL'),
        }
    ).done(
        function (result) {
            // When the request is completed, retrigger woocommerce checkout features with the internal trigger.
            if (typeof result === 'object' && result['update']) {
                jQuery('body').trigger('update_checkout');
            } else {
                alert("Unable to update customer type.");
            }
        }
    )
}
```

Resurs Bank Payment Gateway for WooCommerce (v2.2)

Resurs Checkout & Simplified Flow

This space is reserved for the prior WooCommerce plugin named "Resurs Bank Payment Gateway for WooCommerce". If you are looking for the new Merchant API-plugin named "Resurs Bank Payments for WooCommerce", [you should look here!](#)

Download

This plugin is not available at the official WordPress registry as it is not allowed to have two similar plugins available at the same time. This plugin has instead been replaced with a plugin that supports Resurs Merchant API.

Each individual version tag is also available from [bitbucket](#) if you have needs to reinstall it, or upgrade it. All installs have to be done manually as of 12 april 2023.

Download the latest plugin version below:

- [2.2.105.zip](#)

Older releases:

- [2.2.104.zip](#)
- [2.2.103.zip](#)
- [2.2.102.zip](#)
- [2.2.101.zip](#)
- [2.2.100.zip](#)

Requirements

- At least 1 GB memory or above. Preferably give php (via php.ini) a free amount of memory to play with by setting memory_limit to -1.
- At least PHP 7.1 since WooCommerce itself requires it in newer version.
- [PHP-SoapClient + XML](#) (see requirements).
- Curl or access to php streams.
- An SSL client.

You can also see [PHP and development libraries](#)

Multisite/WordPress Networks

The plugin **do** support WordPress networks (aka multisite), however it does not support running one webservice account over many sites at once. The main rule that Resurs Bank works with is that one webservice account only works for one site. Running multiple sites **do require** multiple webservice accounts! Read more about this at [The problems with WPMU \(Wordpress Network\) and Resurs webservices](#).

The flow behind the plugin

This WooCommerce-plugin support multiple flows from Resurs Bank depending on your needs (and country)

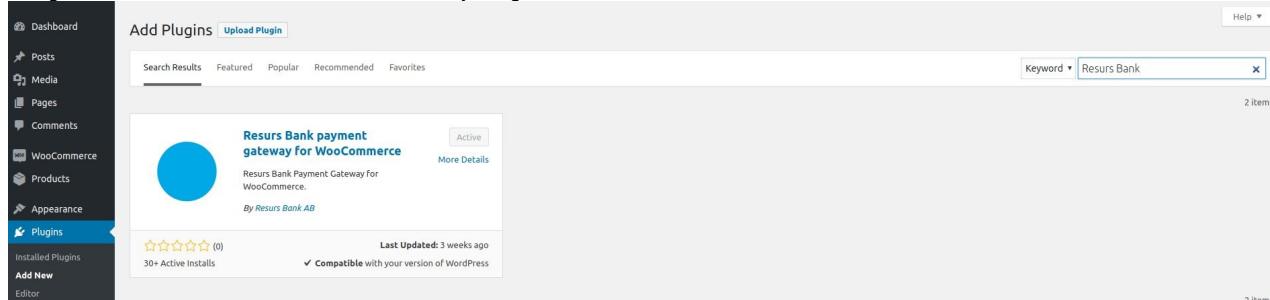
- Simplified flow
- Hosted flow (Paypal-like checkout)
- Resurs Checkout

It is also recommended that you read about the [concepts and domain](#), order status FROZEN and its handling is important in the fraud and order concept.

Plugin installation - versions, requirements and issue tracking

It is *recommended* that you install the plugin via the plugin manager in Wordpress (Example: <site-url>/wp-admin/plugin-install.php? s=Resurs+Bank&tab=search&type=term).

Using the search terms "Resurs Bank" should lead you right:



Internal behaviours, troubleshooting and known issues

Compatibility

The plugin are compatible and works together with most of WooCommerce and WordPress platforms. Discoveries has been made, that a few modules might be conflicting, but as of february 2022 we've removed the old list of compatibility issues as they are outdated. If you find any new conflicts, feel free to report this.

Disabling fields for getAddress, but some fields are still showing?

This is an expected behaviour and normally occurs when you have mixed payment methods (by means, you have payment methods that covers both NATURAL and LEGAL customers). To be able to switch over to "Legal mode", the radio buttons used at the get address forms is still required or you will be stuck with NATURAL-customers.

Handling decimals

Setting decimals to 0 in WooCommerce will result in an incorrect rounding of product prices. It is therefore advised to set decimal points to 2. [Read more about it here.](#)

Handling discounts and coupons with hosted flow

If you plan to handle discounts and coupons within the hosted flow, make sure that the setting for handling VAT is disabled. Hosted flow does not allow payments with negative tax values.

Webhosting companies and blocked callbacks

Some web hosting companies tend to block access from web-browsers like "Java". As our callbacks are sent from a Java-client from Resurs Bank, Callbacks might not work properly if your hosting provider blocks access based on the browser identification.

Aftershop, handling completed orders and discount

In a normal state, without discounts and customized orders in the order administration, each action that renders debiting, refunds, annulments and so on is based on the order content on Resurs Bank side. In short, such actions will make the plugin do a `getPayment` first. Just to make sure that correct articles will be handled as they can be customized by title, price and description. When done, the order will get annulled, credited or finalized.

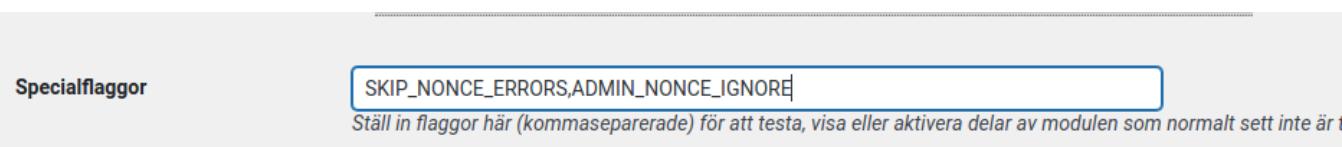
When it comes to discount, orders will be handled differently. As the original price, compared to an added discount, may have changed the properties of each line, the plugin instead honors the content given by WooCommerce. As the price, due to a discount, has changed the plugin now has to trust the data received from woocommerce. Therefore, the payments in Resurs merchant portal could look different compared to the content of woocommerce order admins. For example, if an article is completely discounted (meaning the coupon used has set the article price to 0), that article may not show up in the portal as debited - since the customer does not have to pay for it. If you see such cases, that happens because the plugin consider the order customized by woocommerce.

Site caching

Sites that uses caching, to get better performance, may experience problems at for example upgrades or code updates - in cases where javascripts are cached by mistake in renderers and browsers.

Can't update user credentials in wp-admin

From time to time, especially when you do a lot of actions in admin for the Resurs plugin, updates seem to suddenly stop working. Normally, many functions in the admin control is reached but protected by `wordpress nonces`, which sometimes stops validating properly. If you are in an emergency need, when this happens you can simply use two flags in the settings in the Resurs advanced section (`SKIP_NONCE_ERRORS` and `ADMIN_NONCE_IGNORE`). When those flags are enabled, you can save your data safely again.



Do the plugin handle any e-mail sending

tldr;

No.

Why?

WooCommerce is handling all the logics behind the e-mail sending. What's really handling the e-mail flow is the status changes. When something happens in the plugin that triggers the order status to change, WooCommerce is handling the rest. If WooCommerce is configured to send e-mail (which can be seen in the Email-section of the admin panel), it will also send an e-mail. Order statuses can however be partially mapped differently (see below).

Email notifications

Email notifications sent from WooCommerce are listed below. Click on an email to configure it.

To ensure your store's notifications arrive in your and your customers' inboxes, we recommend connecting your email address to your domain and setting up a dedicated SMTP server. If something doesn't seem right, check our [Email troubleshooting page](#).

Email	Content type	Recipient(s)
<input checked="" type="checkbox"/> New order	text/html	
<input checked="" type="checkbox"/> Cancelled order	text/html	
<input checked="" type="checkbox"/> Failed order	text/html	
<input checked="" type="checkbox"/> Order on-hold	text/html	Customer
<input checked="" type="checkbox"/> Processing order	text/html	Customer
<input checked="" type="checkbox"/> Completed order	text/html	Customer
<input checked="" type="checkbox"/> Refunded order	text/html	Customer
<input type="radio"/> Customer invoice / Order details	text/html	Customer
<input checked="" type="checkbox"/> Customer note	text/html	Customer

Order statuses on callbacks

Some payment methods causes the plugin to instantly finalize the order. By default, this will put the order in "completed" which for some merchants also means that it is delivered. This is more common among travel companies and merchants that sells digital products, besides payment methods like **Vipps** and **Swish**. To prevent payment methods like Vipps and Swish to put an order into completed, you should take a look in the advanced sections under "Callbacks". This can change the way statuses, and thereby e-mail, are handled.

Callbacks	
Order status on instant finalizations	<input type="text" value="Processing"/> <small>Payment methods like SWISH, Vips, direct bank transfers, and so on tend to be followed by direct debiting which finalizes orders before they are shipped. To prevent this, you can set up a specific status for such payment methods when Resurs callback event FINALIZATION occurs. Default status is "completed".</small>
Instant debitible payment methods	<input type="text" value="Chosen by plugin"/> <small>Payment methods that are considered instantly debitible during a payment process. If you choose the top alternative the plugin will choose for you if the correct default method (swish) are available.</small>

Customized part payment information on product pages (WooCommerce)

When you set up your woocommerce platform there may be payment methods available for part payments. They could be easily activated by clicking on the method you prefer display on separate product pages.

ID	Namn	Kassatitel	Delbetalningsfaktor	Aktivera/inaktivera
			Aktivera/inaktivera genom att klicka på X-en	
SWISH	Swish	Swish		<input checked="" type="checkbox"/>
COMPINVOICE	Företagsfaktura	Företagsfaktura		<input checked="" type="checkbox"/>
INVOICE	Faktura privat	Faktura privat		<input checked="" type="checkbox"/>
PSPCARD	Bankkortsbetalning	Bankkortsbetalning		<input checked="" type="checkbox"/>
PSPCARD	Kreditkortsbetalning	Kreditkortsbetalning		<input checked="" type="checkbox"/>
PARTPAYMENT	Delbetalning	Delbetalning	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
CARD	Befintligt kort	Befintligt kort	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
NEWCARD	Nytt kort	Nytt kort	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

When activated, simply pick a part payment period like this:



Activation of this feature normally looks like this (the notification text just above the text "Delbetalा fråν" will always display in staging mode):

HAPPY NINJA

★★★★★ (2 kundrecensioner)

kr1,180.00

Test aktiverat: I produktion visas denna information när minsta belopp överstiger 0.

Delbetalा fråν kr49.00 per månad | [Läs mer](#)

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam feugiat vitae ultricies

The customization

If this text isn't enough for you (note that this is not a multilingual feature today), you can customize this text by creating your own page (WordPress page), like this:

The screenshot shows the WordPress editor interface. At the top, there are standard editing tools like a W icon, a plus sign, a pen, and arrows. To the right are buttons for 'Byt till utkast', 'Förhandsgranska', 'Uppdatera', and settings. Below the toolbar is a rich text editor with a toolbar containing icons for bold, italic, underline, etc. A modal window is open on the right, titled 'Block'. It shows two tabs: 'Dokument' (selected) and 'Block'. Under 'Block', there is a 'Stycke' block with the sub-instruction 'Starta med grunden för varje berättelse.' Below this is a 'Typografi' section with 'Förinställd storlek' set to 'Standard' and 'Anpassad' with a dropdown menu. At the bottom of the modal is a 'Ändra till' button.

Vår text för delbetalningar

Om du väljer att delbetal din vara med i [annuityDuration] månader, så betalar du endast [payFrom] kronor per månad!

When this page is saved and ready to push up to the store, update the configuration to point to this page:

The screenshot shows a configuration screen for a 'delbetalningswidget'. The title is 'Inställningar för delbetalningswidget'. On the left, there's a section for 'Widget för anpassningsbar delbetalningvy'. A dropdown menu is open, showing 'Vår text för delbetalningar' as the selected option. Below the dropdown is a note: 'Om du väljer en sida här, så kommer den bli den primära vyn för din anpassade delbetalningswidget.' At the bottom of this section is a link 'Tillgängliga kortkoder för oavstående vy'.

And your store will now instead display this text:

The screenshot shows a product listing on a store. At the top, it says '★★★★★ (2 kurarecensioner)'. Below that is the price 'kr1,180.00'. Underneath the price is a text block: 'Om du väljer att delbetal din vara med i 24 månader, så betalar du endast 49 kronor per månad!'. At the very bottom of the listing, there is some very small, illegible text.

Hooks/filters v2.2 (and core tweaks)

- Filter: Sending storeId into the payload
- Filter: Handling of payment methods prefix paths
- Filter: Sort order of payment methods in wp-admin
- Filter: Part payment widget string
- Action: Order info
- Action: Handling sessions
- Partial refunds (on price level) - resurs_refund_price_override
- Front end hooks for country changes

Core Tweaks

- The old row editor for partial annul/crediting is no longer available
- Performance issues and plugin breaks within wp-admin (2.2.24+)

Settable advanced flags

- GIT_BIN
- ECOM_CACHE_TIME
- CURL_TIMEOUT
- DISABLE_SSL_VALIDATION
- Miscellaneous
 - FEE_EDITOR
 - DEBUG
 - NONCE_ERRORS
 - SKIP_DIGEST_VALIDATION
 - XDEBUG_SESSION_START
 - PTEUSERS

The filters described on this page always requires that you keep our plugin updated.

Filter: Sending storeId into the payload

As of the release of v2.1.x we not only deliver the plugin with EComPHP 1.1.x, we also support storeids in the shop, via the filter **resursbank_set_storeid**-filter.

```
function setResursStoreId() {
    $return = 107999; // Enter your code here, to fetch or generate the storeId to return.

    return $return;
}
add_filter( 'resursbank_set_storeid', 'setResursStoreId', 10, 1 );
```

A sample plugin for testings, including (almost) the above example can be found at <https://bitbucket.org/resursbankplugins/resurs-storeid-test/overview>

Filter: Handling of payment methods prefix paths

As of 2.2.18 we have changed the path structure for where payment method models are stored. From now on, since they are files, they are stored in subpaths within the includes structure and the directories are created based on WordPress database table_prefix names. However, if this method is not good enough, you have a small chance to fix such "errors", by using the filter **resurs_bank_model_prefix**. For example, if you wish to keep those directories numeric only, you can add something like this to an external plugin for your site:

```
function altPrefix($return) {
    global $table_prefix;
    $return = preg_replace('/[^0-9]/', '', $table_prefix);
    return $return;
}
add_filter('resurs_bank_model_prefix', 'altPrefix');
```

The function example could actually contain whatever you need to adapt a multisite solution. In newer versions, we do not use models like this so this should not be a problem for future versions.

Filter: Sort order of payment methods in wp-admin

The filer **resurs_admin_sort_methods_by_value** makes it possible to change the content of the default payment method "name" in the WP-admin. Since version 2.2.17.

```

/**
 * @param string $description
 * @param stdClass $paymentMethodContent
 * @return mixed
 */
function resursMethodValue($description, $paymentMethodContent)
{
    return $paymentMethodContent->id;
}

add_filter('resurs_admin_sort_methods_by_value', 'resursMethodValue', 10, 2);

```

The above example will change the output from the default description order:

Method	Title	AnnuityFactor <small>Activate/Disable by clicking the X-boxes</small>	ID	Enable/Disable
Betala med kort	Betala med kort	<input checked="" type="checkbox"/>	NEWCARD	<input checked="" type="checkbox"/>
Betala med kort	Betala med kort	<input type="checkbox"/>	CARD	<input checked="" type="checkbox"/>
Delbetaling	Delbetaling	<input checked="" type="checkbox"/>	PARTPAYMENT	<input checked="" type="checkbox"/>
		24 Månader - 0% ränta		
Faktura privat	Faktura privat	<input type="checkbox"/>	INVOICE	<input checked="" type="checkbox"/>
Fakturakampanj	Fakturakampanj	<input type="checkbox"/>	INVOICE2	<input checked="" type="checkbox"/>
FtgFkt	FtgFkt	<input type="checkbox"/>	COMPINVOICE	<input checked="" type="checkbox"/>
Swisha	Swisha	<input type="checkbox"/>	SWISH	<input checked="" type="checkbox"/>

to sort methods by their id names:

Method	Title	AnnuityFactor <small>Activate/Disable by clicking the X-boxes</small>	ID	Enable/Disable
Betala med kort	Betala med kort	<input type="checkbox"/>	CARD	<input checked="" type="checkbox"/>
FtgFkt	FtgFkt	<input type="checkbox"/>	COMPINVOICE	<input checked="" type="checkbox"/>
Faktura privat	Faktura privat	<input type="checkbox"/>	INVOICE	<input checked="" type="checkbox"/>
Fakturakampanj	Fakturakampanj	<input type="checkbox"/>	INVOICE2	<input checked="" type="checkbox"/>
Betala med kort	Betala med kort	<input checked="" type="checkbox"/>	NEWCARD	<input checked="" type="checkbox"/>
Delbetaling	Delbetaling	<input checked="" type="checkbox"/>	PARTPAYMENT	<input checked="" type="checkbox"/>
		24 Månader - 0% ränta		
Swisha	Swisha	<input type="checkbox"/>	SWISH	<input checked="" type="checkbox"/>

The way this is handled should not affect anything but the list itself.

Filter: Part payment widget string

We have seen moments where the part payment widget text data is not translated properly. In earlier versions (< 2.2.18) this data had to be hard coded if you had to replace the string in the widget (in cases where translations don't work). The below plugin-filter might help you with this, without the necessary hard coding. This filter works from 2.2.18+

Part payment text replacer

```

/**
 * @param $defaultString The html content string created by the plugin.
 * @param $annuityData The data values from annuity factors.
 * @return string Your new string.
 */
function resursPartPaymentStringReplace($defaultString, $annuityData){
    return sprintf("Pröjsa från %s spänn", $annuityData);
}
add_filter('resursbank_custom_annuity_string', 'resursPartPaymentStringReplace', 10, 2);

```

Action: Order info

In version 2.0 of our WooCommerce plugin you are able to integrate with a few hooks in our plugin. The hooks are recently introduced and therefore considered experimental, and there might be more of them in the future. For the moment, we support some hooks in the callbacks, which means for each received callback from Resurs Ecommerce there will be a forwarding hook, that is able to registered what happened in the callback.

Coding example

```
function resurs_hook_orderinfo($paymentInfo) {  
  
    /*  
     * What we get from here:  
     *  
     * Array  
     *  
     * [id] => resursOrderReference  
     * [fraud] => 1  
     * [frozen] => 1  
     * [status] => IS_ANNULLLED  
     * [booked] => time()  
     * [finalized] => time()  
     * [callback] => ANNULMENT  
     */  
  
    // Do what you need to do here  
}  
add_action('resurs_hook_callback', 'resurs_hook_orderinfo', 1, 1);
```

The example above is very simple - the callback hook takes one argument and contains data about the received callback in an array (See the example).

Field	Type	Explained
<i>id</i>	string	The order reference id, booked at Resurs Bank
<i>fraud</i>	boolean	If the booked order is flagged possible fraud (will change on "not suspected" to false)
<i>frozen</i>	boolean	If the order is frozen
<i>status</i>	string	paymentStatus from Resurs Bank
<i>booked</i>	integer	Unix timestamp for when the order was booked
<i>finalized</i>	integer	Unix timestamp for when the order was finalized (if it was)
<i>callback</i>	string	If the hook was triggered by a callback , this is the name of the callback

Action: Handling sessions

In very rare cases, there might be a need of controlling how the session is handled. There are two new filters in the latest version that handles whether `session_start()` should be invoked by the plugin, or not.

Filter name	Description
<code>resursbank_start_session_before</code>	Using this filter and setting return value to true will be the same as disabling the session during this phase. This filter can disable the session creation completely. Use at your own risk, if you believe that you need to handle this by yourself.
<code>resursbank_start_session_outside_admin_only</code>	Disable session creation when in admin if true (will make a <code>!is_admin()</code> -check). Activation of this filter will turn <code>session_start()</code> off, when in admin.

Partial refunds (on price level) - `resurs_refund_price_override`

Normally, partial refunds in woocommerce are supported on quantity level (as of 2.2.21). This means that you now only can remove entire rows from an order, you can also for example change the quantity (i.e. if you by mistake ordered a product with 2 pieces in quantity, you can annul/credit one of them). However, when it comes to manipulation of pricing per row, we usually do not allow such things. But there is a override filter for this, if you want to take further risks with the partial annullment/refunding functions: **resurs_refund_price_override**. Doing something like below, with your own plugin you can make our plugin override the price manipulation protection.

```
function refundFunction() {  
    return true;  
}  
add_filter('resurs_refund_price_override', 'refundFunction');
```

Front end hooks for country changes

In prior versions of the plugin there had been no relations between the getAddress-form and the country changes, if the store has several delivery countries, mostly due to the codebase. As the codebase is not included in WooCommerce own triggers (which they should, see example [here](#) and [here](#) for a proper implementation with payment review fragments - in that example, instead of exclude the getAddress-forms from a natural flow it is instead included in it - and renders fields when WooCommerce itself requests to do so). In this specific case, nothing will happen unless the scripts gets notified on the country changes.

This commit adds a new trigger, handled from the plugin as long as billing_country can be bound to an onChange-event via jQuery:

```
$('#billing_country').on('change', function () {
    $('body').trigger('resursCountryChange', {newCountry: this.value});});
```

How it is caught:

```
$RB(document).on('resursCountryChange', function(e, data) {});
```

This specific event is used internally by our own plugin on the condition that the government id field is forced to be shown in the last parts of the checkout, mostly to demonstrate how it works. This setting can be found in the "advanced"-section:

The event triggering is likely to depend on how templates are set up to show the billing data fields, and the importance lies within that #billing_country must be present at the document-ready event.

Core Tweaks

The old row editor for partial annul/crediting is no longer available

It actually is. But for a while this function is moved into the status type of the order. By means, editing order to partially credit/annul rows on an order is only available as long as WooCommerce has a pending order. In older releases of WooCommerce, this editor has always been active. But don't feel sorry. You can still re-activate this function, even if it's probably not recommended. What you need is a plugin that loads after ours - or if you edit the core (**resursbank main.php**) and add the code below. Note: In future 2.2-releases, this function will be added to the plugin.

Editable order code

```
/**
 * @param bool $isEditable
 * @param $that WC_Admin_Order
 *
 * @return bool
 */
function resurs_order_is_editable($isEditable, $that)
{
    $resursOrderId = wc_get_payment_id_by_order_id($that->get_id());

    if (!empty($resursOrderId)) {
        return true;
    }

    return $isEditable;
}

add_filter('wc_order_is_editable', 'resurs_order_is_editable', 10, 2);
```

Performance issues and plugin breaks within wp-admin (2.2.24+)

There are rare moments when our plugin may interfere with parts of wp-admin which causes strange untraceable errors. As long as we can't find error logs that indicates what's wrong, we've added an extremely odd filter that controls the major activity for the plugin, and where it is allowed to act. In the beginning of this plugin's era it was built in a way too fragile way, so it is normally active in each "browser session" (where one page reload here is counted as a single session). This means that the plugin are running on pages (like the template editor) even if it has no role there. Normally, this works fine, as the plugin should be idle. We've been working on the same behaviour in the frontend parts, where filters and actions restricts usage where it does not belong. But the road is quite long to make this perfect.

For wp-admin, this is where the filter comes in. If works like this - and you can build your own plugin that lifts or restricts access in the same way we did. The filter itself is called **allow_resurs_run** and is activated by a setting in the control panel. As we have a lot more store managers than just you, we don't want to ruin life for them by always having this feature active.

The screenshot shows a settings page with a header 'Blandat'. Below it is a section titled 'Prevent performance interferences in wp-admin' with a checkbox labeled 'Aktiverad' (Enabled). A note below the checkbox reads: 'If you experience very high pressure from the plugin when it is enabled, this setting tries to prevent its own presence on pages where it normally should not interfere with the platform. This setting is experimental.'

If this setting is enabled, the filter is triggered after the plugin has checked if it is located on the admin-pages. As this feature is experimental, it should be **considered activated as a last resort only**. The setting makes sure that the plugin is active in a few scenarios only, at wp-admin-stage. For the moment, this happens in the shop-order-editor and woocommerce settings only. However, it CAN be extended by something like below. In the example, the filter always returns true, so it will become permanently active in wp-admin. However, if you know there's a section in wp-admin you need to reach and you believe that the plugin is the showstopper, you can adapt this part to just prevent activity on that page. But don't forget to activate the feature first!

```
add_filter('allow_resurs_run', 'wp_admin_resurs_limitations', 10, 2);

/**
 * @param $allow Current inbound allow state.
 * @param $info Very basic requests from _REQUEST and _POST parameters that could easily be analyzed.
 * @return bool If true, the plugin is allowed to proceed.
 */
function wp_admin_resurs_limitations($allow,$info) {
    return true;
}
```

Settable advanced flags

In some cases we think it is not necessary to have a setting for each detailed setting that can be done. In v2 there are also, unfortunayely, a very limited count hooks and filters so when it is really necessary to change the regular behaviour of the plugin some flags can be set. We however, do not take any responsibility over such actions. Below is a list of some of them. They can be set in to ways: **FLAG_KEY** and **FLAG_KEY=VALUE**, where only the flag without a value will be considered a boolean while the other way is considered a key with a value.

Flag	Value	Behaviour
GIT_BIN	string	Activates a function that checks if your plugin is on the latest commit (when cloned from a master repository).
ECOM_CACHE_TIME	int	getPayments can in newer EComPHP-releases be cached for a limited time (3 sec). Each time a getPayment is done, EComPHP will return a stored copy of the getPayment-API instead of making a new request. Since the default time is 3 seconds, this setting can be used to change this value.
CURL_TIMEOUT	int	Double-pass value. The timeout set in this flag will be passed through over to EComPHP and further into the communications driver (6.0.x) to replace the default connectivity timeout .
DISABLE_SSL_VALIDATION		Experiencing much trouble with SSL certificates (very common with self signed certificates as the communications driver by default requires a valid SSL certificate to be able to communicate). This flag is passed over from EComPHP to the driver to lower the security requirements. Normally this should only be used in test environments.

Miscellaneous

FEE_EDIT_OR		Activates a fee editor directly in the MAIN configuration for payment methods.
DEBUG		Passed debug mode over to EComPHP to activate debugging features (deprecated, as EComPHP normally in newer versions allows debugging anyway).
NONCE_ERRORS		Developer feature to test nonce validations.
SKIP_DIGEST_VALIDATION		Developer feature.
XDEBUG_SESSION_START		Developer feature for xdebuggers.

PTEUSERS	string	Internal usage only.
-----------------	--------	----------------------

The problems with WPMU (Wordpress Network) and Resurs webservices

This page has been added to describe why multisite setups - where **several sites handles one webservice account** - will never work properly. Please be aware that this is not the same solution as if you were running WPML (the multilingual plugin).

First of all, the problem does not actually lies within the plugin but in Resurs Bank - it is actually Resurs Bank that does not support multisite accounts. There are some issues that is simply not supported here:

Order ID numbering

Normally, the plugin are connecting an order to its order ID in the platform by a datestamp. You can also run the plugin by setting the order id as the incremental post ID for the multisite. If you tend to follow an incremental id and initialize your first site with order number sequence 1000, your second with 2000, and so on, those order numbers will at some point in time collide.

When orders are created like this on Resurs Bank where the same webservice account are used over several sites, your sites may be able to sort this issue out. On Resurs Bank this won't happen as all orders are created on the same account. When your first site reaches 1999 and tries to switch over to sequence 2000, the webservice will stop working as an exception will be thrown at bookPayment level - saying that order id 2000 already exists. You could of course use a higher sequence as this is only an example. However, at some point - mentioned above - there might be a risk of conflicts between your sites.

Callbacks: Only one site will be able to handle callbacks as the database tables are split up to be running each site separately

When the plugin are registering callbacks it is very site specific. In a WP Network only one URL will be registered without the ability to identify which site the callback is intended to land on. In theory you may make this work by handling each callback by its order id. However, this is not recommended (see the section about Order ID numbering).

The callback URLs is also a problem

When callbacks are registered, salt keys are also registered at the webservice endpoint. Those salt keys are stored at the site where they were registered. By means, if you have four sites and register callback with site 2, the other sites will be unable to handle the callbacks due to the different saltkeying. Besides, your site 2 may register different URL's compared to your other sites. This is a limitation that resides at Resurs Bank.

Using sku as article numbers in old plugin

The old module once had support for sku-values as articlenumbers on order rows, but has since the evolving of our plugins been removed from the configuration panel due to decisions. The old plugin is currently explicitly using id/permalinks for the order rows instead. At the time, this was more safe than the opposite option as sku's very often was missing from old products.

However, the ability to use woocommerce sku-id's instead of post-id (when they exist) is still available in the plugin, but not as an active configurable setting. Since it is not included in the configuration array anymore, you are therefore also unable to override it normally. The setting itself will (when enabled) is actively using the sku instead of the regular id, and fail over to the regular id if sku is not set - if you still manage to enable the setting.

Yes. There is a solution.

Temporary override sku feature plugin

Create a file in your plugin directory (**/wp-content/plugins**) in your platform, for example named resurs-force-sku.php - the file should contain the code below.

Save it and activate it as a standalone plugin in your store.

To remove and disable this feature (as it also saves the value **useSku** in the database), disable this plugin from your plugin manager. Then go to the Resurs Bank-tabs and save the configuration again.

resurs-force-sku.php

```
<?php

/**
 * Plugin Name: Force sku usage for resurs-bank-payment-gateway-for-woocommerce (v2.2.105 and above)
 * Description: Restores the usage of sku names in orders.
 * Version: 1.0.0
 * Author:
 */

add_action('plugins_loaded', function () {
    add_filter('resurs_bank_form_fields', function ($formFields, $sectionName) {
        if (
            (preg_match('/resurs_bank/i', $sectionName) || $sectionName === 'defaults') &&
            !isset($formFields['useSku']))
        ) {
            $ns = 'woocommerce_resurs-bank_settings';
            // Validate prior options in case useSku has been saved as disabled.
            $optionCheck = get_option($ns);
            if (isset($optionCheck['useSku']) && $optionCheck['useSku'] === 'no') {
                $optionCheck['useSku'] = 'yes';
                // Update the options if that's the case.
                update_option($ns, $optionCheck);
            }

            // Return useSku to the plugin as if it was still configurable, to convince
            // woocommerce to bring it to storefront.
            $formFields['useSku'] = [
                'id' => 'useSku',
                'title' => __(
                    'Use sku instead of id, when available',
                    'resurs-bank-payment-gateway-for-woocommerce'
                ),
                'type' => 'checkbox',
                'default' => 'yes',
                'description' => __(
                    'Enforcing sku as artno instead of id, when it is available.',
                    'resurs-bank-payment-gateway-for-woocommerce'
                ),
            ];
        }
        return $formFields;
    }, 11, 2);
});
```

After Shop Service API

Content of this page

- Annulling (Cancel order) & Crediting (Refund) - [annulPayment](#) & [creditPayment](#)
- Debit order - [finalizePayment](#)
- Make a new, additional debit on an existing order - [additionalDebitOfPayment](#)
- Add meta data to order - [addMetaData](#)
- Search for payments - [findPayments](#)
- Get detailed information about order - [getPayment](#)
- Get a specified document from order (PDF) - [getPaymentDocument](#)
- Get names of all documents for an order - [getPaymentDocumentNames](#)
- Issue customer identification token - [issueCustomerIdentificationToken](#)
- Invalidate Customer Identification Token - [invalidateCustomerIdentificationToken](#)
- Available actions, depending on status and amount.
- General setup for orderstatus at Resurs Bank.

When the customer completes a purchase and a order is created at Resurs Bank, the order is authorized and debitable. In order for the money to be transferred to the merchant's bank account the order has to be finalized (debited).

This page describes how a order can be handled after its creation and how you finalize the order.

You can do all this from Resurs Bank [paymentAdmin](#), a web-based interface.

Or you can use the after shop webservices described below.

WSDL: <https://test.resurs.com/ecommerce-test/ws/V4/AfterShopFlowService?wsdl>
Service Endpoint: <https://test.resurs.com/ecommerce-test/ws/V4/AfterShopFlowService>

Annulling (Cancel order) & Crediting (Refund) - [annulPayment](#) & [creditPayment](#)

Should you need to undo a payment, then you should use the [annulPayment](#) method.

If the order has been [finalized](#) then you must use the [creditPayment](#) method instead.

When to use what? Learn more about [Annulment and Crediting](#)

Debit order - [finalizePayment](#)

When the order is complete, you call this method, the money is then transferred from the customer account to yours.

Make a new, additional debit on an existing order - [additionalDebitOfPayment](#)

For creating an additional debit of the payment.

Add meta data to order - [addMetaData](#)

If you wish to add more meta data to the payment. This can be used to register additional information about the payment, and they may also be used for searching.

Search for payments - [findPayments](#)

If you want to search for specific payments.

To get the size of the searchresult, use [calculateResultSize](#) which for example can be used for paging.

Get detailed information about order - [getPayment](#)

Returns the details of a payment.

Get a specified document from order (PDF) - [getPaymentDocument](#)

Retrieves a specified document from the payment as a pdf.

Get names of all documents for an order - [getPaymentDocumentNames](#)

Retrieves the document names available for the payment.

Issue customer identification token - [issueCustomerIdentificationToken](#)

Issues a customer identification token that can identify this customer in further operations.
 These functions do require the customer to be identified, and they require either a token, or information to identify the customer.
 Tokens are intended to be saved with the user profile in the webshop.
 In this way we delegate identification of the customer to the webshop after the initial identification is done.

Invalidate Customer Identification Token - [invalidateCustomerIdentificationToken](#)

Invalidates customer identification token(s).

Available actions, depending on status and amount.

getPayment	getPaymentResponse <status>DEBITABLE<status>	getPaymentResponse <status>CREDITABLE<status>	getPaymentResponse <status>DEBITABLE<status> <status>CREDITABLE<status>
additionalDebitOfPayment	YES	YES	YES
annulPayment	YES	NO	YES
finalizePayment	YES	NO	YES
creditPayment	NO	YES	YES

Maximum amount for:	is
additionalDebitOfPayment	limit - totalAmount
annulPayment	sum(AUTHORIZE.totalAmount) - sum(DEBIT.totalAmount) - sum(ANNUL.totalAmount)
finalizePayment	sum(AUTHORIZE.totalAmount) - sum(DEBIT.totalAmount) - sum(ANNUL.totalAmount)
creditPayment	sum(DEBIT.totalAmount) - sum(CREDIT.totalAmount)



In order to be able to charge more than the authenticated amount (maximum up to the requested limit) you must first make an additionalDebitOfPayment, this will increase the authenticated amount. Note that to update existing order line, "artNo", "description" and "unitAmountWithoutVat" have to match.

Example:

The order is authenticated for 2 products at a price of 100 sek / pcs, ie a total of 200 sek.
 To increase the number of products to 3, you send additionalDebitOfPayment with the increase, in this case 1 pcs (This provided that the current limit allows it).
 Then you can debit (finalizePayment) 3 items with a sum of 300 sek.

NOTE! In order to use this feature, this need to be arranged with the partner's KAM. This "shadow limit" can be applied either for a fixed amount, e.g. 2000SEK, or a percentage of the checkout value, e.g. 10%.

After shop event	Resurs Invoice	Resurs partpayment	Resurs Card /Account	Visa/MasterCard	Bank payments directly from account to account: <i>Swish, Trustly</i>
Debiting whole order	YES	YES	YES	YES	NO
Debiting part order	YES	YES	YES	YES	NO
Crediting whole order	YES	YES	YES	YES	YES
Crediting part order	YES	YES	YES	YES	YES

Annulment whole order	YES	YES	YES	YES	NO
Annulment part order	YES	YES	YES	NO	NO
Additional Debit of Payment	YES	YES	YES	NO	NO

General setup for *orderstatus* at Resurs Bank.

You can use this when mapping your own system's orderstatus to Resurs *orderstatus*. Make a [SOAP](#) or [REST](#) for current order, to determine the current order status.

Orderstatus at Resurs	Case
On-Hold (Pending)	getPaymentResponse.frozen=true
Processing (Ready/Confirmed Resurs)	getPaymentResponse.status: "DEBITABLE"=true
Completed	getPaymentResponse.status: "IS_DEBITED"=true AND "DEBITABLE"=false AND getPaymentResponse.totalAmount > 0
Cancelled (Annulled)	getPaymentResponse.status: "IS_ANNULLED"=true AND "IS_CREDITED"=false AND getPaymentResponse.totalAmount = 0
Refunded (Credited)	getPaymentResponse.status: "IS_CREDITED"=true AND getPaymentResponse.totalAmount = 0



Payment type Swish (SE) is automatically finalized as default.

Additional Debit of Payment

additionalDebitOfPayment

Makes a new, additional debit on an existing payment. This reserves the amount on the customer's account. NB: If it is a credit payment, there must be room for the additional debit within the limit.

Input (Literal)

Name	Type	Occurs	Nullable?	Description
paymentId	id	1..1	No	The identity of the payment to which to make an additional debit.
paymentSpec	paymentSpec	0..1	No	The specification of the additional payment.
createdBy	nonEmptyString	0..1	No	The username of the person performing the operation.

Faults

Name	Content	Description
ECommerceErrorException	ECommerceError	Failed to make an additional debit on the payment. See error for details.

Introduction

If the customer wants to add a product on an existing order that has not yet been shipped, you can add it with help of additionalDebitOfPayment.

AdditionalDebitOfPayment makes a new, additional debit on an existing payment/order. This reserves the amount on the customer's account. If it is a credit payment, there must be room for the additional debit within the limit. If the payment method on the other hand is not a card there is an extra room for later debits/purchases when the limit application form was filled by the customer.

It's also possible to make part debits of a payment. For example, if the customer has ordered 5 products, it's possible to debit and ship 3 of these products and wait for the rest of them for unknown circumstances. This can also be made from [Payment administration @ Resurs Bank](#).

Restrictions

additionalDebitOfPayment cannot be done on external payment methods such as Swish and card transactions (VISA/Mastercard) via Nets/d2i

What is paymentSpec?

The payment details. In it's simplest form it's just sum, i.e. totalAmount and totalVatAmount are set, but there are no specLines. If nothing else is said you shall send specLines .

Contains elements as defined in the following table.

Component	Type	Occurs	Nullable?	Description
specLines	specLine	0..*	No	The list of payment lines. In the case you're sending a simple payment, without lines, this parameter should be left empty. Sending payment lines may, or may not, be mandatory, depending on the contract with Resurs Bank.
totalAmount	positive Decimal	1..1	No	The total payment amount. The sum of all line amounts (if there are lines supplied) including VAT. If this payment is without lines this is the only value to be set on the payment spec.
totalVatAmount	decimal	0..1	Yes	The total VAT amount of the payment when there are specification lines supplied. If there are no lines this field must be empty (null).

Paymentspec - speclines



Observe that in order to have an invoice with specified order data, make sure to include the specLines in the web service call.

specLines are not mandatory for processing payments.
specLines can vary between start, finalize, credit and annul. It doesn't matter. Only the sum matter.
specLines make better invoices and help the merchant

The code below shows an example of one paymentSpec row when calling startPaymentSession method

paymentSpec example in bookPayment

Paymentspec - rounding

[see Rounding](#)

additionalDebitOfPayment - code example

additionalDebitOfPayment

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Body>
    <additionalDebitOfPayment xmlns="http://ecommerce.resurs.com/v4/msg/aftershopflow">
      <paymentId>300007457</paymentId>
      <paymentSpec>
        <specLines>
          <id>106</id>
          <artNo>106</artNo>
          <description>Frakt Norge</description>
          <quantity>1</quantity>
          <unitMeasure/>
          <unitAmountWithoutVat>76</unitAmountWithoutVat>
          <vatPct>25</vatPct>
          <totalVatAmount>19</totalVatAmount>
          <totalAmount>95</totalAmount>
        </specLines>
        <totalAmount>95</totalAmount>
        <totalVatAmount>19</totalVatAmount>
      </paymentSpec>
      <createdBy>User</createdBy>
    </additionalDebitOfPayment>
  </soap:Body>
</soap:Envelope>
```

Annulling

annulPayment

Annuls the payment. This removes the reservation on the customer's account. NB: For a payment to be annulled, it must be booked. If it has been finalized, it can no longer be annulled. (Finalized payments have to be credited.)

Input (Literal)

Name	Type	Occurs	Nillable?	Description
paymentId	id	1..1	No	The identity of the payment.
partPaymentSpec	paymentSpec	0..1	No	The specification of the payment annulment.
createdBy	nonEmptyString	0..1	No	The username of the person performing the operation.

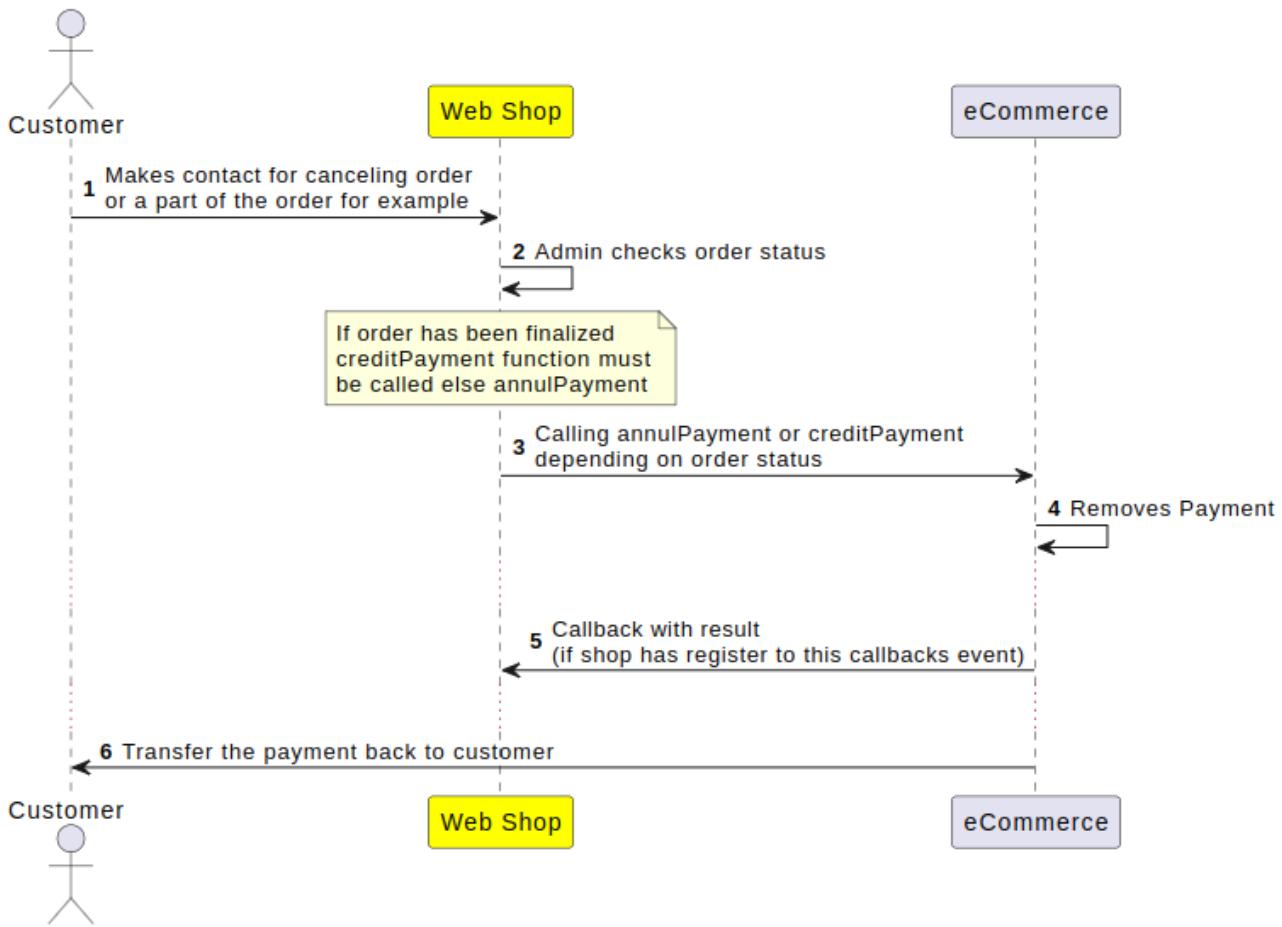
Faults

Name	Content	Description
ECommerceErrorException	ECommerceError	Failed to annul the payment. See error for details.

Introduction

Annuls the payment. This removes the reservation on the customer's account. A payment can only be annulled if it's booked. If it has been booked and then finalized, it can no longer be annulled. (Finalized payments have to be credited).

If you are unsure when to use this method, [read more about Annulment and Crediting](#).



What is a PaymentSpec

The payment details. In it's simplest form it's just sum, i.e. totalAmount and totalVatAmount are set, but there are no specLines. If nothing else is said you shall send specLines .

Contains elements as defined in the following table.

Component	Type	Occurs	Nillable?	Description
specLines	specLine	0..*	No	The list of payment lines. In the case you're sending a simple payment, without lines, this parameter should be left empty. Sending payment lines may, or may not, be mandatory, depending on the contract with Resurs Bank.
totalAmount	positive Decimal	1..1	No	The total payment amount. The sum of all line amounts (if there are lines supplied) including VAT. If this payment is without lines this is the only value to be set on the payment spec.
totalVatAmount	decimal	0..1	Yes	The total VAT amount of the payment when there are specification lines supplied. If there are no lines this field must be empty (null).

Paymentspec - speclines



Observe that in order to have an invoice with specified order data, make sure to include the specLines in the web service call.

specLines are not mandatory for processing payments.
 specLines can vary between start, finalize, credit and annul. It doesn't matter. Only the sum matter.
 specLines make better invoices and help the merchant

The code below shows an example of one paymentSpec row when calling startPaymentSession method

paymentSpec example in bookPayment

Paymentspec - rounding

see [Rounding](#)

annulPayment - code example

annulPayment

```
<!-- Fully Specified -->
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Body>
    <annulPayment xmlns="http://ecommerce.resurs.com/v4/msg/aftershopflow">
      <paymentId>300007457</paymentId>
      <partPaymentSpec xmlns="">
        <specLines>
          <id>shippingfee</id>
          <artNo>shippingfee</artNo>
          <description>Fraktgebyr</description>
          <quantity>1</quantity>
          <unitMeasure>st</unitMeasure>
          <unitAmountWithoutVat>95</unitAmountWithoutVat>
          <vatPct>0</vatPct>
          <totalVatAmount>0</totalVatAmount>
          <totalAmount>95</totalAmount>
        </specLines>
        <totalAmount>95</totalAmount>
        <totalVatAmount>0</totalVatAmount>
      </partPaymentSpec>
      <createdBy>IntegrationService</createdBy>
    </annulPayment>
  </soap:Body>
</soap:Envelope>

<!-- Unspecified -->
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:aft="http://ecommerce.resurs.com/v4/msg/aftershopflow">
  <soapenv:Header/>
  <soapenv:Body>
    <aft:annulPayment>
      <paymentId>12345</paymentId>
      <createdBy>Aftershop_annulPayment</createdBy>
    </aft:annulPayment>
  </soapenv:Body>
</soapenv:Envelope>
```

Specified vs unspecified annulment

If you use the unspecified call without specline or amount, the annulment row in the Payment admin will have a VAT of 20% (0,199999) instead of 25%. The total amount is however correct. If you want also the annulment row to be correct, you will have to use the fully specified call.

Partial annulment on VISA/Mastercard

Partial annulment cannot be done on VISA/Mastercard payment method. If the full amount is not to be debited the customer - you need to finalize this amount and then use [creditPayment](#)

Calculate Searchresult Size

calculateResultSize

Returns the number of payments that match the specified requirements. Can be used for paging of the results.

Input (Literal)

Name	Type	Occurs	Nillable?	Description
searchCriteria	searchCriteria	1..1	No	The search criteria.

Output (Literal)

Name	Type	Occurs	Nillable?	Description
return	int	1..1	No	The number of payments matching the specified search criteria.

Faults

Name	Content	Description
ECommerceErrorException	ECommerceError	Failed to calculate the search result size. See error for details

Introduction

Returns the number of payments that match the specified search requirements. Can be used for paging of the results.

Example

Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:aft="http://ecommerce.resurs.com/v4/msg/aftershopflow">
    <soapenv:Header/>
    <soapenv:Body>
        <aft:calculateResultSize>
            <searchCriteria>
                <paymentMethodId>8</paymentMethodId>
                <statusSet>DEBITABLE</statusSet>
                <statusNotSet>IS_DEBITED</statusNotSet>
            </searchCriteria>
        </aft:calculateResultSize>
    </soapenv:Body>
</soapenv:Envelope>
```

Response

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body>
        <ns2:calculateResultSizeResponse xmlns:ns3="http://ecommerce.resurs.com/v4/msg/exception" xmlns:ns2="http://ecommerce.resurs.com/v4/msg/aftershopflow">
            <return>56</return>
        </ns2:calculateResultSizeResponse>
    </soap:Body>
</soap:Envelope>
```


Crediting / Refunding

creditPayment

Credits the payment. This returns the payment amount from the representative to the customer's account. NB: For a payment to be credited, it must be finalized. (Non-finalized payments have to be annulled.)

Input (Literal)

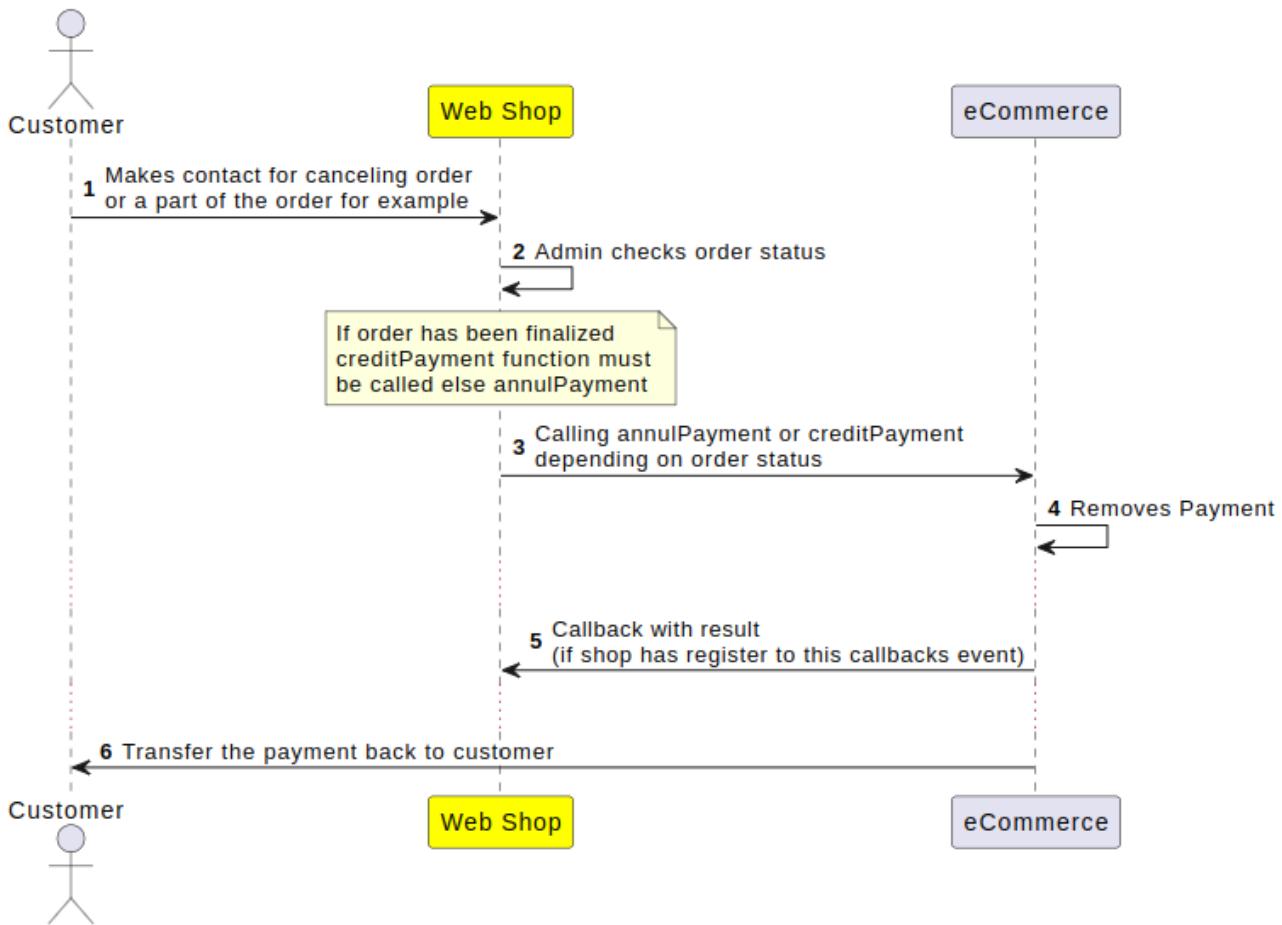
Name	Type	Occurs	Nillable?	Description
paymentId	id	1..1	No	The identity of the payment.
preferredTransactionId	id	0..1	No	Will be printed on the accounting summary. Can be used to track the transaction. If not set it will fallback on paymentId for this value.
partPaymentSpec	paymentSpec	0..1	No	The specification of the payment crediting. PS! Mandatory if payment method is INVOICE
createdBy	nonEmptyString	0..1	No	The username of the person performing the operation.
creditNoteId	id	0..1	No	The credit note number. This will be printed on the credit note. For payment methods other than INVOICE, setting this will generate an error.
creditNoteDate	date	0..1	No	The credit note date. This will be printed on the credit note. For payment methods other than INVOICE, setting this will generate an error. Note: use the format "yyyy-MM-dd" for date.
invoiceDeliveryType	invoiceDeliveryType	0..1	Yes	How the credit invoice should be delivered to the customer. Default: EMAIL

Faults

Name	Content	Description
ECommerceErrorException	ECommerceError	Failed to credit the payment. See error for details.

Introduction

Credits the payment. This returns the payment amount from the representative to the customer's account. For a payment to be credited, it must be finalized. (Non-finalized payments have to be annulled.) If you are unsure when to use this method, read more about [Annulment and Crediting](#).



What is a paymentSpec

The payment details. In its simplest form it's just sum, i.e. totalAmount and totalVatAmount are set, but there are no specLines. If nothing else is said you shall send specLines .

Contains elements as defined in the following table.

Component	Type	Occurs	Nillable?	Description
specLines	specLine	0..*	No	The list of payment lines. In the case you're sending a simple payment, without lines, this parameter should be left empty. Sending payment lines may, or may not, be mandatory, depending on the contract with Resurs Bank.
totalAmount	positive Decimal	1..1	No	The total payment amount. The sum of all line amounts (if there are lines supplied) including VAT. If this payment is without lines this is the only value to be set on the payment spec.
totalVatAmount	decimal	0..1	Yes	The total VAT amount of the payment when there are specification lines supplied. If there are no lines this field must be empty (null).

Paymentspec - speclines

⚠️ Observe that in order to have an invoice with specified order data, make sure to include the specLines in the web service call.

specLines are not mandatory for processing payments.
 specLines can vary between start, finalize, credit and annul. It doesn't matter. Only the sum matter.
 specLines make better invoices and help the merchant

The code below shows an example of one paymentSpec row when calling startPaymentSession method

paymentSpec example in bookPayment

Paymentspec - rounding

[see Rounding](#)

Credit Payment - code example

creditPayment

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:ns1="http://ecommerce.resurs.com/v4/msg/aftershopflow">
  <SOAP-ENV:Body>
    <ns1:creditPayment>
      <paymentId>100100446</paymentId>
      <preferredTransactionId xsi:nil="true"/>
      <partPaymentSpec>
        <specLines>
          <id>1</id>
          <artNo>other</artNo>
          <description>Other</description>
          <quantity>1</quantity>
          <unitMeasure/>
          <unitAmountWithoutVat>199</unitAmountWithoutVat>
          <vatPct>0</vatPct>
          <totalVatAmount>0</totalVatAmount>
          <totalAmount>199</totalAmount>
        </specLines>
        <totalAmount>199</totalAmount>
        <totalVatAmount>0</totalVatAmount>
      </partPaymentSpec>
      <createdBy>User</createdBy>
      <creditNoteId>100100446-CR1</creditNoteId>
      <creditNoteDate>2015-11-21</creditNoteDate>
    </ns1:creditPayment>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

creditPayment with discount

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:ns1="http://ecommerce.resurs.com/v4/msg/aftershopflow">
  <SOAP-ENV:Body>
    <ns1:creditPayment>
      <paymentId>creditexample1</paymentId>
      <preferredTransactionId xsi:nil="true"/>
      <partPaymentSpec>
        <specLines>
          <id>1</id>
          <artNo>1111</artNo>
          <description>Shoes</description>
          <quantity>1.00</quantity>
          <unitMeasure>st</unitMeasure>
          <unitAmountWithoutVat>800.00000</unitAmountWithoutVat>
          <vatPct>25</vatPct>
          <totalVatAmount>200.00</totalVatAmount>
          <totalAmount>1000.00</totalAmount>
        </specLines>
        <specLines>
```

```
<id>2</id>
<artNo>1112</artNo>
<description>Socks</description>
<quantity>1.00000</quantity>
<unitMeasure>st</unitMeasure>
<unitAmountWithoutVat>200.00000</unitAmountWithoutVat>
<vatPct>25.00000</vatPct>
<totalVatAmount>50.000000000000000</totalVatAmount>
<totalAmount>250.000000000000000</totalAmount>
</specLines>
<specLines>
    <id>3</id>
    <artNo>1113</artNo>
    <description>Rabatt</description>
    <quantity>1.00000</quantity>
    <unitMeasure>st</unitMeasure>
    <unitAmountWithoutVat>-120.00000</unitAmountWithoutVat>
    <vatPct>25.00000</vatPct>
    <totalVatAmount>-30.000000000000000</totalVatAmount>
    <totalAmount>-150.000000000000000</totalAmount>
</specLines>
    <totalAmount>1100</totalAmount>
    <totalVatAmount>220</totalVatAmount>
</partPaymentSpec>
    <createdBy>Partner Integration</createdBy>
    <creditNoteId>1001022226-PI2</creditNoteId>
    <creditNoteDate>2021-02-16</creditNoteDate>
</ns1:creditPayment>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Finalize Payment

finalizePayment

Finalizes a payment. When a payment is finalized, the amount will be transferred from the customer's account to that of the representative. NB: For a payment to be finalized, it must be booked and it cannot be frozen.

Input (Literal)

Name	Type	Occurs	Nillable?	Description
paymentId	id	1..1	No	The identity of the payment.
preferredTransactionId	id	0..1	No	Will be printed on the accounting summary. Can be used to track the transaction. If not set it will fallback on paymentId for this value. NOT SUPPORTED for external payment methods (VISA/Mastercard/SWISH/TRUSTLY etc)
partPaymentSpec	partPaymentSpec	1..1	No	If you are to finalize an invoice, you will need to supply the partPaymentSpec with specLines . Without the specLines , the customer won't see the orderrows that has been supplied in your bookPayment/POST. For any other payment method, using partPaymentSpec with specLines is optional.
createdBy	nonEmptyString	0..1	No	The username of the person performing the operation.
orderId	id	0..1	No	The order number.
orderDate	date	0..1	Yes	The order date. For payment methods other than INVOICE, setting this will generate an error. Note: use the format "yyyy-MM-dd" for date.
invoiceId	id	0..1	Yes	The invoice number. This will be printed on the invoice. For payment methods other than INVOICE, setting this will generate an error. An alternative is to let Resurs Bank generate the invoice ID, in this case this field is omitted.
invoiceDate	date	0..1	Yes	The invoice date. This will be printed on the invoice. For payment methods other than INVOICE, setting this will generate an error. Note: use the format "yyyy-MM-dd" for date.
invoiceDeliveryType	invoiceDeliveryType	0..1	Yes	How the invoice should be delivered. If used, put in: "EMAIL"



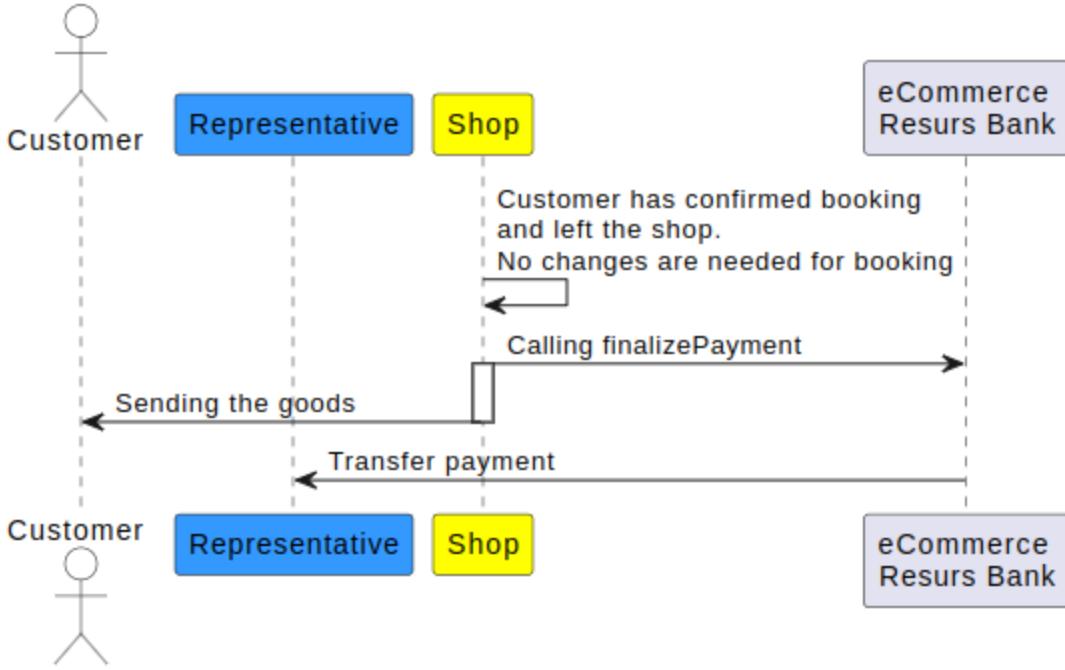
Find out if the payment method is an invoice by making a `getPaymentMethods`. If `<specificType>INVOICE</specificType>` in `getPaymentMethodsResponse`, payment method is an invoice.

Faults

Name	Content	Description
ECommerceErrorException	ECommerceError	Failed to finalize the payment. See error for details.

Introduction

This method should be called just before the goods are delivered. When a payment is finalized, the amount will be transferred from the customer's account to the representative. For a payment to be finalized, it must be booked and it cannot be frozen. You can see more about each parameter above in the method box and what these parameters stands for.



What if something need to change?

Does something on the order need to be changed due to the customer contacting the web shop? This is achieved either via web services or the [payment admin](#).

Does the whole order or only part of it need to be canceled? Or should any more items be added? For more information and how this is done, please see [after shop flow](#) and / or [payment admin](#)

What is paymentSpec?

The payment details. In its simplest form it's just sum, i.e. `totalAmount` and `totalVatAmount` are set, but there are no `specLines`. If nothing else is said you shall send `specLines`.

Contains elements as defined in the following table.

Component	Type	Occurs	Nillable?	Description
<code>specLines</code>	<code>specLine</code>	<code>0..*</code>	No	The list of payment lines. In the case you're sending a simple payment, without lines, this parameter should be left empty. Sending payment lines may, or may not, be mandatory, depending on the contract with Resurs Bank.
<code>totalAmount</code>	<code>positive Decimal</code>	<code>1..1</code>	No	The total payment amount. The sum of all line amounts (if there are lines supplied) including VAT. If this payment is without lines this is the only value to be set on the payment spec.
<code>totalVatAmount</code>	<code>decimal</code>	<code>0..1</code>	Yes	The total VAT amount of the payment when there are specification lines supplied. If there are no lines this field must be empty (null).

Paymentspec - speclines

Observe that in order to have an invoice with specified order data, make sure to include the specLines in the web service call.

`specLines` are not mandatory for processing payments.

`specLines` can vary between start, finalize, credit and annul. It doesn't matter. Only the sum matter.

`specLines` make better invoices and help the merchant

The code below shows an example of one `paymentSpec` row when calling `startPaymentSession` method

paymentSpec example in bookPayment

Paymentspec - rounding

see Rounding

What is the difference between a payment and a payment-diff and how are they related?

Finalize Payment for different payment methods

The finalize payment differs somewhat from payment methods, for example if the payment method is an invoice, an invoice document will be created in the finalization step. See the examples below to see the difference between a card payment and an invoice.

Finalize Payment - code example

Please be aware that the example below shows the simplest form of finalization. It's also possible to finalize just a portion of the payment, as would be required upon delivery of half of an order.

finalizePayment

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:aft="http://ecommerce.resurs.com/v4/msg/aftershopflow">
  <soapenv:Header/>
  <soapenv:Body>
    <aft:finalizePayment>
      <paymentId>Pay-1349186086946-3861</paymentId>
      <preferredTransactionId>TrD-12345</preferredTransactionId>
      <partPaymentSpec>
        <totalAmount>30.00</totalAmount>
      </partPaymentSpec>
      <createdBy>User</createdBy>
      <orderId>Ord-23456</orderId>
    </aft:finalizePayment>
  </soapenv:Body>
</soapenv:Envelope>
```

If you are finalizing an invoice you might want to specify the payment more for a more detailed invoice. Note that you can activate an invoice sequence that will automatically generate an invoiceId for you, read more about invoice sequence [here](#). The example below shows a more detailed form of finalization.

Detailed finalizePayment invoice

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:aft="http://ecommerce.resurs.com/v4/msg/aftershopflow">
  <soapenv:Header/>
  <soapenv:Body>
    <aft:finalizePayment>
      <paymentId>Pay-1349186086946-3861</paymentId>
      <preferredTransactionId>TrD-12345</preferredTransactionId>
      <partPaymentSpec>
        <specLines>
          <id>1</id>
          <artNo>NUT-001</artNo>
          <description>Nut (M8)</description>
          <quantity>21.00</quantity>
          <unitMeasure>st</unitMeasure>
          <unitAmountWithoutVat>0.80</unitAmountWithoutVat>
          <vatPct>25</vatPct>
          <totalVatAmount>4.2</totalVatAmount>
          <totalAmount>21.00</totalAmount>
        </specLines>
        <specLines>
          <id>2</id>
          <artNo>BOLT-002</artNo>
          <description>Bolt (M8x125mm)</description>
          <quantity>17.00</quantity>
        </specLines>
      </partPaymentSpec>
    </aft:finalizePayment>
  </soapenv:Body>
</soapenv:Envelope>
```

```

<unitMeasure>st</unitMeasure>
<unitAmountWithoutVat>1.60</unitAmountWithoutVat>
<vatPct>25</vatPct>
<totalVatAmount>6.8</totalVatAmount>
<totalAmount>34.00</totalAmount>
</specLines>
<totalAmount>55.00</totalAmount>
<totalVatAmount>11.00</totalVatAmount>
</partPaymentSpec>
<orderId>Ord-23456</orderId>
<orderDate>2012-10-02</orderDate>
<invoiceId>DebInv-34567</invoiceId>
<invoiceDate>2012-10-02</invoiceDate>
</aft:finalizePayment>
</soapenv:Body>
</soapenv:Envelope>

```

Detailed finalizePayment invoice with discount

```

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:aft="http://ecommerce.resurs.com/v4/msg/aftershopflow">
<soapenv:Header/>
<soapenv:Body>
<aft:finalizePayment>
<paymentId>finalizeexample1</paymentId>
<partPaymentSpec>
<specLines>
<id>1</id>
<artNo>1111</artNo>
<description>Shoes</description>
<quantity>1.00</quantity>
<unitMeasure>st</unitMeasure>
<unitAmountWithoutVat>800.00000</unitAmountWithoutVat>
<vatPct>25</vatPct>
<totalVatAmount>200.00</totalVatAmount>
<totalAmount>1000.00</totalAmount>
</specLines>
<specLines>
<id>2</id>
<artNo>1112</artNo>
<description>Socks</description>
<quantity>1.00000</quantity>
<unitMeasure>st</unitMeasure>
<unitAmountWithoutVat>200.00000</unitAmountWithoutVat>
<vatPct>25.00000</vatPct>
<totalVatAmount>50.000000000000000</totalVatAmount>
<totalAmount>250.000000000000000</totalAmount>
</specLines>
<specLines>
<id>3</id>
<artNo>1113</artNo>
<description>Rabatt</description>
<quantity>1.00000</quantity>
<unitMeasure>st</unitMeasure>
<unitAmountWithoutVat>-120.00000</unitAmountWithoutVat>
<vatPct>25.00000</vatPct>
<totalVatAmount>-30.000000000000000</totalVatAmount>
<totalAmount>-150.000000000000000</totalAmount>
</specLines>
<totalAmount>1100</totalAmount>
<totalVatAmount>220</totalVatAmount>
</partPaymentSpec>
</aft:finalizePayment>
</soapenv:Body>
</soapenv:Envelope>
</soapenv:Envelope>

```

**Note!**

When handling Visa/Mastercard-transcations, you must finalize 100% of the authorized amount

Find Payments

findPayments

Searches for payments that match the specified requirements.

Input (Literal)

Name	Type	Occurs	Nillable?	Description
searchCriteria	searchCriteria	1..1	No	The search criteria.
pageNumber	positiveInteger	0..1	No	The desired page number.
itemsPerPage	positiveInteger	0..1	No	The number of items to return per page.
sortBy	sortOrder	0..1	No	The sort order of the results.

Output (Literal)

Name	Type	Occurs	Nillable?	Description
return	basicPayment	0..*	No	The of payments matching the specified search criteria.

Faults

Name	Content	Description
ECommerceErrorException	ECommerceError	Failed to search for payments. See error for details.

Searches for payments that matches the specified requirements and returns the payments matching the specified search criteria. If you know the identity of the payment you are looking for, it is better to use the [getPayment method](#).

The search may consist of information like governmentId, customer name, paymentmethodId etc. You can specify the search criteria as much or as little you want to help you find what payment(s) you are looking for.

Sort order

Can be sorted by these options:

Value	Description
PAYMENT_ID	Sort the result on payment identity.
CUSTOMER_GOVERNMENT_ID	Sort the result on customer government identity.
CUSTOMER_NAME	Sort the result on customer name.
BOOKED_TIME	Sort the result on payment booking time.
MODIFIED_TIME	Sort the result on payment modification time.
FINALIZED_TIME	Sort the result on payment finalization time.
AMOUNT	Sort the result on total payment amount, taking into consideration the payment part status.

Example

The example below shows a very simple search for payments that has been paid with paymentMethod 'Nytt Kort'

```
findPaymentRequest
```

```

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:aft="http://ecommerce.resurs.com/v4/msg/aftershopflow">
    <soapenv:Header/>
    <soapenv:Body>
        <aft:findPayments>
            <searchCriteria>
                <paymentMethodId>NYTT_KORT</paymentMethodId>
                <statusSet>DEBITABLE</statusSet>
                <statusNotSet>IS_DEBITED</statusNotSet>
            </searchCriteria>
        </aft:findPayments>
    </soapenv:Body>
</soapenv:Envelope>

```

findPaymentResponse

```

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body>
        <ns2:findPaymentsResponse xmlns:ns3="http://ecommerce.resurs.com/v4/msg/exception" xmlns:ns2="http://ecommerce.resurs.com/v4/msg/aftershopflow">
            <return>
                <paymentId>Sess-1346079637802-2201</paymentId>
                <paymentMethodId>NYTT_KORT</paymentMethodId>
                <paymentMethodName>Nytt kort</paymentMethodName>
                <governmentId>7409208423</governmentId>
                <fullName>Palma Manuela</fullName>
                <booked>2012-08-27T17:00:26+02:00</booked>
                <modified>2012-08-27T17:00:26+02:00</modified>
                <totalAmount>30.00000</totalAmount>
                <frozen>false</frozen>
                <status>DEBITABLE</status>
            </return>
            <return>
                <paymentId>Sess-1346079643364-6334</paymentId>
                <paymentMethodId>NYTT_KORT</paymentMethodId>
                <paymentMethodName>Nytt kort</paymentMethodName>
                <governmentId>7301021528</governmentId>
                <fullName>Mohammed Ayham</fullName>
                <booked>2012-08-27T17:00:31+02:00</booked>
                <modified>2012-08-27T17:00:31+02:00</modified>
                <totalAmount>30.00000</totalAmount>
                <frozen>false</frozen>
                <status>DEBITABLE</status>
            </return>
        </ns2:findPaymentsResponse>
    </soap:Body>
</soap:Envelope>

```

Get Payment

getPayment

Retrieves detailed information about the payment.

Input (Literal)

Name	Type	Occurs	Nillable?	Description
paymentId	id	1..1	No	The identity of the payment.

Output (Literal)

Name	Type	Occurs	Nillable?	Description
return	payment	1..1	No	The payment details.

Faults

Name	Content	Description
ECommerceErrorException	ECommerceError	Failed to retrieve the payment details. See error for details.

Introduction

Retrieves detailed information about a specific payment. You need the paymentId for the payment you want to have information about. To get all available payments you should use the [findPayments](#) method.

Example

An example showing request/response for a get payment

Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:aft="http://ecommerce.resurs.com/v4/msg/aftershopflow">
  <soapenv:Header/>
  <soapenv:Body>
    <aft:getPayment>
      <paymentId>Pay-1372762301644-3166</paymentId> <!-- Seraching for payment with id in the exshop-->
    </aft:getPayment>
  </soapenv:Body>
</soapenv:Envelope>
```

Response

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns2:getPaymentResponse xmlns:ns3="http://ecommerce.resurs.com/v4/msg/exception" xmlns:ns2="http://ecommerce.resurs.com/v4/msg/aftershopflow">
      <return>
        <id>Pay-1372762301644-3166</id>
        <totalAmount>0.0000000000000000000000</totalAmount>
        <metaData>
          <key>Expressfrakt</key>
          <value>true</value>
        </metaData>
        <limit>3030.00000</limit>
      </return>
    </ns2:getPaymentResponse>
  </soap:Body>
</soap:Envelope>
```

```

<paymentDiffs>
  <type>AUTHORIZE</type>
  <created>2013-07-02T12:51:51+02:00</created>
  <createdBy>SHOP_FLOW</createdBy>
  <paymentSpec>
    <specLines>
      <id>BOLT-002</id>
      <artNo>BOLT-002</artNo>
      <description>Bolt (M8x125mm)</description>
      <quantity>10.00000</quantity>
      <unitMeasure>st</unitMeasure>
      <unitAmountWithoutVat>1.60000</unitAmountWithoutVat>
      <vatPct>25.00000</vatPct>
      <totalVatAmount>4.000000000000000000</totalVatAmount>
      <totalAmount>20.000000000000000000</totalAmount>
    </specLines>
    <specLines>
      <id>NUT-001</id>
      <artNo>NUT-001</artNo>
      <description>Nut (M8)</description>
      <quantity>10.00000</quantity>
      <unitMeasure>st</unitMeasure>
      <unitAmountWithoutVat>0.80000</unitAmountWithoutVat>
      <vatPct>25.00000</vatPct>
      <totalVatAmount>2.000000000000000000</totalVatAmount>
      <totalAmount>10.000000000000000000</totalAmount>
    </specLines>
    <totalAmount>30.000000000000000000</totalAmount>
    <totalVatAmount>6.000000000000000000</totalVatAmount>
  </paymentSpec>
</paymentDiffs>
<paymentDiffs>
  <type>DEBIT</type>
  <transactionId>TrD-1372762301644-3166</transactionId>
  <created>2013-07-02T12:51:51+02:00</created>
  <paymentSpec>
    <totalAmount>30.000000000</totalAmount>
    <totalVatAmount>6.0000000000</totalVatAmount>
  </paymentSpec>
  <orderId>Ord-1372762301644-3166</orderId>
  <invoiceId>DebInv-1372762301644-3166</invoiceId>
</paymentDiffs>
<paymentDiffs>
  <type>CREDIT</type>
  <transactionId>TrC-1372762301644-3166</transactionId>
  <created>2013-07-02T12:51:52+02:00</created>
  <paymentSpec>
    <totalAmount>30.000000000</totalAmount>
    <totalVatAmount>6.0000000000</totalVatAmount>
  </paymentSpec>
  <invoiceId>CrN-1372762301644-3166</invoiceId>
</paymentDiffs>
<customer>
  <governmentId>8305147715</governmentId>
  <address>
    <fullName>Vincent Williamsson Andersson</fullName>
    <firstName>Vincent</firstName>
    <lastName>Williamsson Andersson</lastName>
    <addressRow1>Glassgatan 15</addressRow1>
    <postalArea>Göteborg</postalArea>
    <postalCode>41655</postalCode>
    <country>SE</country>
  </address>
  <phone>0707123456</phone>
  <email>testdata@resurs.se</email>
  <type>NATURAL</type>
</customer>
<booked>2013-07-02T12:51:49+02:00</booked>
<finalized>2013-07-02T12:51:51+02:00</finalized>
<paymentMethodId>Faktura</paymentMethodId>
<fraud>false</fraud>

```

```
<frozen>false</frozen>
<status>IS_DEBITED</status>
<status>IS_CREDITED</status>
<paymentMethodType>INVOICE</paymentMethodType>
</return>
</ns2:getPaymentResponse>
</soap:Body>
</soap:Envelope>
```

Error example

When trying to get a payment that doesn't exist you get this error with the fixableByYou flag set to true.

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <soap:Fault>
      <faultcode>soap:Server</faultcode>
      <faultstring>This order group do not exists</faultstring>
      <detail>
        <ns3:ECommerceError xmlns:ns2="http://ecommerce.resurs.com/v4/msg/aftershopflow" xmlns:ns3="http://ecommerce.resurs.com/v4/msg/exception">
          <errorTypeDescription>REFERENCED_DATA_DONT_EXISTS</errorTypeDescription>
          <errorTypeId>8</errorTypeId>
          <fixableByYou>true</fixableByYou>
          <userErrorMessage>Efterfrågad order/betalning (Pay-1372762301644-31656) kan inte hittas i
databasen.</userErrorMessage>
        </ns3:ECommerceError>
      </detail>
    </soap:Fault>
  </soap:Body>
</soap:Envelope>
```

Get Payment Document (PDF)

getPaymentDocument

Retrieves a specified document from the payment.

Input (Literal)

Name	Type	Occurs	Nillable?	Description
paymentId	id	1..1	No	The identity of the payment.
documentName	nonEmptyString	1..1	No	The name of the document.

Output (Literal)

Name	Type	Occurs	Nillable?	Description
return	pdf	0..*	No	The document.

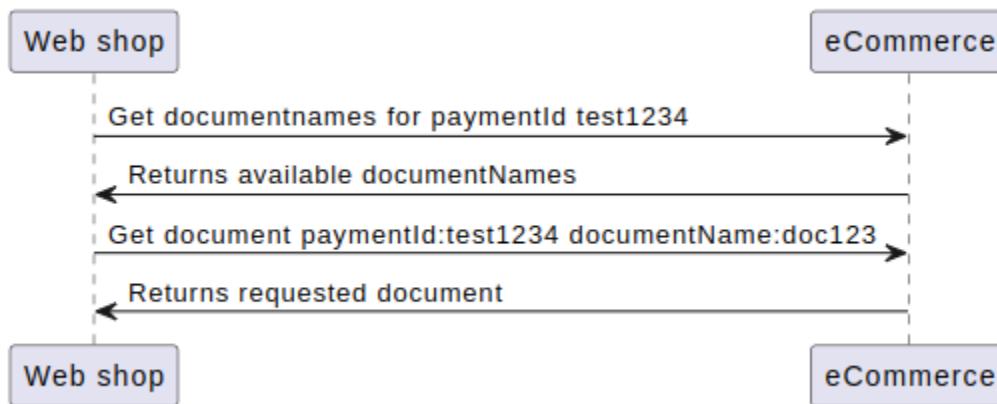
Faults

Name	Content	Description
ECommerceErrorException	ECommerceError	Failed to retrieve the specified payment document. See error for details

Introduction

Retrieves a specified document from the payment as a pdf, for example the invoice. You can get the available document names for a payment by calling the [getPaymentDocumentNames](#) method.

When trying to get the requested document, the invoice for example, you send in the paymentId and the documentName as parameters.



Example

This example shows request/response from the exshop account.

Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:aft="http://ecommerce.resurs.com/v4/msg/aftershopflow">
  <soapenv:Header/>
  <soapenv:Body>
    <aft:getPaymentDocument>
      <paymentId>Pay-1372762113890-5677</paymentId>
      <documentName>INVOICE_20130702-124247_1514</documentName> <!-- a documentname which I got from getDocumentNames in exshop with the paymentId -->
    </aft:getPaymentDocument>
  </soapenv:Body>
</soapenv:Envelope>
```

Response

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns2:getPaymentDocumentResponse xmlns:ns3="http://ecommerce.resurs.com/v4/msg/exception" xmlns:ns2="http://ecommerce.resurs.com/v4/msg/aftershopflow">
      <return>
        <name>INVOICE_20130702-124247_1514</name>
        <pdfData>JVBERi0xLjQKJeljz9MKNiAwIG9iago8PC9UeXBll1hPYmpLY3QvQ29sb3JTcGFjZS9...etc</pdfData> <!-- Have shortened the response because of the length -->
      </return>
    </ns2:getPaymentDocumentResponse>
  </soap:Body>
</soap:Envelope>
```

Get Payment Document Names

getPaymentDocumentNames

Retrieves the names of all documents associated with the payments. These include, but are not necessarily limited to, previously generated invoices and credit notes sent to the customer.

Input (Literal)

Name	Type	Occurs	Nillable?	Description
paymentId	id	1..1	No	The identity of the payment.

Output (Literal)

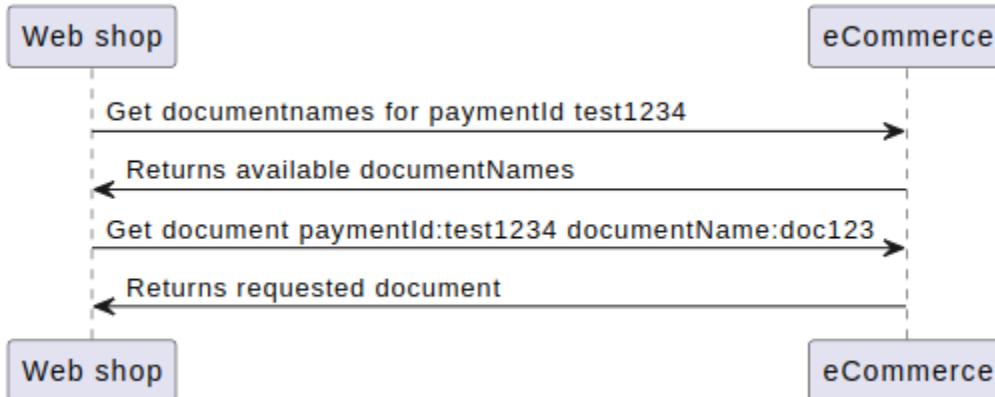
Name	Type	Occurs	Nillable?	Description
return	string	0..*	No	The names of all documents associated with the payment.

Faults

Name	Content	Description
ECommerceErrorException	ECommerceError	Failed to retrieve the list of payment document names. See error for details

Introduction

This method retrieves all available documents that are associated with the payment, the invoice document for example. This method retrieves the names of the available document which you then use in the [getPaymentDocument](#) method along with the paymentId to retrieve the requested document.



Example

This example shows the request/response when trying to get available document names for a specific payment.

Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:aft="http://ecommerce.resurs.com/v4/msg/aftershopflow">
<soapenv:Header/>
<soapenv:Body>
<aft:getPaymentDocumentNames>
```

```
<paymentId>Pay-1372762113890-5677</paymentId> <!-- get available document names with paymentid in  
exshop -->  
</aft:getPaymentDocumentNames>  
</soapenv:Body>  
</soapenv:Envelope>
```

Response

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">  
  <soap:Body>  
    <ns2:getPaymentDocumentNamesResponse xmlns:ns3="http://ecommerce.resurs.com/v4/msg/exception" xmlns:ns2="  
http://ecommerce.resurs.com/v4/msg/aftershopflow">  
      <return>INVOICE_20130702-124247_1514</return>  
      <return>INVOICE_CREDITNOTE_20130702-124248_1515</return>  
    </ns2:getPaymentDocumentNamesResponse>  
  </soap:Body>  
</soap:Envelope>
```

MetaData AfterShop

addMetaData

Adds meta data to the payment. The meta data can be used to register additional information about the payment, and they may also be used for searching. Currently, meta data cannot be removed from a payment. However, existing values can be over-written.

Input (Literal)

Name	Type	Occurs	Nillable?	Description
paymentId	id	1..1	No	The identity of the payment.
key	string	1..1	No	The meta data key.
value	string	0..1	No	The meta data value.

Faults

Name	Content	Description
ECommerceErrorException	ECommerceError	Failed to add meta data to the payment. See error for details.

What & why

Key/value data and its additional information in an order, determined by the e-retailer. This can be added to the order and can later be useful when [searching](#) for a payment. It can be anything, like information about the shipment. You can manage metadata through [PaymentAdmin](#) or using the [aftershop webservice](#)

What is metadata?

In short, it is key/value data piggybacked on the payment.

Read here: [Associated metadata](#)

Recognized keys and meaning

Generally we don't look at the metadata. Listed below are the exceptions to that rule.

Key name	Expected format	Description
invoiceExtRef	String. 46 chars.	In the case that the payment generates invoices and credit notes this value will be printed as 'Your reference', for example the sales person responsible. Mostly for company invoices.
CustomerId	String. 20 chars.	In the case that the payment generates invoices and credit notes this value will be printed as 'Customer Id'



Faktura

Fakturaadress
 Vincent Williamsson Alexandersson
 Glassgatan 15
 41655 Göteborg

Leveransadress
 Vincent Williamsson Alexandersson
 Glassgatan 15
 41655 Göteborg

Fakturanummer 100171	Kundnummer Max 20 tkn	Fakturadatum 2016-10-19	Er ref / projektnummer Max 46 tkn
Betalningsreferens/OCR 8538792891	Betalningsvillkor 41 dagar	Förfallodatum 2016-11-30	
Betalnummer 1001083	Dröjsmålsränta 20,00 %		

Art. Nr.	Beskrivning	Antal	A-pris inkl. moms	Moms	Belopp
0	Order line:1	1	625,00 kr	25 %	625,00 kr
2	Order line:2	1	12,50 kr	25 %	12,50 kr
80	Discount	1	-6,25 kr	25 %	-6,25 kr
90	Shipping & Handling	1	5,00 kr	0 %	5,00 kr
				Totalt SEK exkl. moms	510,00 kr
				Momsbelopp	126,25 kr
				Totalt SEK inkl. moms	636,25 kr
från 100,00 kr/mån*					

* Genom samarbete med Resurs Bank erbjuder vi möjligheten till förlängd kredit och räntefri delbetalning. [Läs mer här](#) eller på nästa sida.

Introduction

Adds meta data to the payment. The meta data can be used to register additional information about the payment, and they may also be used for searching. Currently, meta data cannot be removed from a payment. However, existing values can be over-written.

Example

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:aft="http://ecommerce.resurs.com/v4/msg/aftershopflow">
  <soapenv:Header/>
  <soapenv:Body>
    <aft:addMetaData>
      <paymentId>Sess-1346079637802-2201</paymentId>
      <key>Transport</key>
      <!--Optional:-->
      <value>Use airline transport</value>
    </aft:addMetaData>
  </soapenv:Body>
</soapenv:Envelope>
```

Payment administration GUI

Merchant Portal is Resurs Banks new web-GUI for administrating orders. Merchant Portal has been built with an omni-state of mind where you can have information regarding both e-commerce orders and your in store-sales.

A few new fatures in Merchant Portal are that the Super User can add his/her own users, one can switch the order which the payment methods are seen by the customer in the Resurs Checkout and an FAQ with the most common questions.

Merchant Portal

 Resurs Bank	 Resurs Bank	 Resurs Bank	 Resurs Bank
Välkommen till Merchant Portal	Velkommen til Merchant Portal	Velkommen til Merchant Portal	Tervetuloa Merchant Portaliin
VÄLJ LAND Sverige	VELG LAND Norge	VÆLG LAND Danmark	VALITSE MAA Ruotsi
BUTIKS-ID <input type="text"/>	BUTIKK-ID <input type="text"/>	BUTIKS-ID <input type="text"/>	MYYMÄLÄN TUNNUS <input type="text"/>
ANVÄNDARNAMN ELLER E-POST * <input type="text"/> Fyll i ditt användarnamn eller e-post	BRUKERNAVN ELLER E-POST * <input type="text"/> Fyll ut brukernavn eller e-post	BRUGERNAVN ELLER E-MAIL * <input type="text"/> Udfyld med dit brugernavn eller e-mail	KÄYTTÄJÄTUNNUS TAI SÄHKÖPOSTI * <input type="text"/> Anna käyttäjätunnus tai sähköposti
LÖSENORD * <input type="password"/> <input type="checkbox"/> Visa lösenord	PASSORDO * <input type="password"/> <input type="checkbox"/> Vis passord	ADGANGSKODE * <input type="password"/> <input type="checkbox"/> Vis adgangskode	SALASANA * <input type="password"/> <input type="checkbox"/> Näytä salasana
LOGGA IN	LOGG INN	LOG IND	KIRJAUDU SISÄÄN
Glemt ditt lösenord? Återställ det här.	Glemt passord? Tilbakestil her.	Glemt din adgangskode? Nulstil det her.	Unohditko salasanasi? Palauta se tästä.



Merchant Portal is offered in Swedish, Finnish, Norwegian and Danish.



If you prefer to use your own business system, you can choose to integrate our **after shop webservices**.

Merchant Portal



Välkommen till Merchant Portal

VÄLJ LAND

Sverige

BUTIKS-ID

ANVÄNDARNAMN ELLER E-POST *

|

LÖSENORD *

Visa lösenord

LOGGA IN

Glömt ditt lösenord? [Återställ det här.](#)

Configuration Merchant Portal

Invoice number (Fakturanummer)

This is an invoice sequence that can help the store to generate unique invoice numbers. If you choose to use this, you set a desired start number. Once the sequence is activated you no longer have the option to fill in desired invoice number when debiting an order.

The setting is found at Settings Payment methods Invoice number (Inställningar Betalsätt Fakturanummer)

Invoice data (Fakturadata)

This is the data that will be used when your invoices are created. When you click save a note saying "saved" will appear at the top of the page and indicate that the data is saved. To add a logotype to your invoice click the choose file button (Välj filer...) and choose the image you want to use (.png and .jpg is supported). When you click the upload button the logotype should appear next to the form if the upload was successful.

The setting is found at Settings Business
Business information (Inställningar Företaget Företagsuppgifter)



If the buttons dont work as expected try to clear your browser's cache using "Ctrl + F5" and try again.

Invoice number

Betalsätt

Här kan du konfigurera betalsätt beroende på kanal, se aktuella avgifter och giltighetstid samt göra inställningar för Swish-integrationen.

Konfigurera betalsätt				REDIGERA
VÄLJ KANAL				
<input type="button" value="Merchant Portal"/>				
BETALSÄTT I MERCHANT PORTAL				
Betalsätt	Giltighetstid på betalsätt	Avgift	Visningsordning	
Swish				
REDIGERA				
BETALNINGSMOTTAGARE				
<input type="text"/>				
CERT.-LÖSENORD				
<input type="text"/>				
CERTIFIKAT				
<input type="button" value="VÄLJ FIL"/>				
Fakturanummer				
REDIGERA				
NÄSTA FAKTURA KOMMER ATT GENERERAS MED NUMMER				
<input type="text" value="0"/>				

Invoice data

Företagsuppgifter

Fyll i företagets uppgifter. Tänk på att informationen som anges kommer att exponeras i material som berör betalningar, såsom fakturor, avier, e-post m.m.

Företagsuppgifter		REDIGERA
JURIDISKT NAMN	<input type="text"/>	
BUTIKSNAMN FÖR DIGITALA UTSKICK	<input type="text" value="mio3"/>	
ADRESS	<input type="text"/>	
POSTNUMMER	<input type="text"/>	
STAD	<input type="text"/>	
LAND	<input type="text" value="Sverige"/>	
TELEFON	<input type="text"/>	
FAX	<input type="text"/>	
E-POST	<input type="text"/>	
WEBB	<input type="text"/>	
MOMSREGNR.	<input type="text"/>	
ORGANISATIONSNUMMER	<input type="text"/>	
F-SKATT	<input checked="" type="checkbox"/>	

Manipulate Payments in Merchant Portal

 The Merchant Portal web-GUI is offered in Swedish, Norwegian, Finnish and Danish.

When the customer completes a purchase, a payment is created. In order for the money to be transferred to the merchants bank account, the order has to be finalized. This page describes how a payment can be changed after its creation and how you finalize the order. You can do all this from your platform, Resurs Banks web-based interface, or by using the [Aftershop Flow](#)

With Merchant Portal, you can adjust the Resurs payment after an order has been made. The adjustments that are supported are described below.

Annull

Annulment can only be done on non-finalized order rows. If a payment is partially finalized, you can still annul the row/s that is/are authorized.

You can annul in the following ways:

1. Select "Annull", (choose which order lines to annul) and press "Submit"
2. Change from "Quantity" to "Amount" and manually type in your wanted description, VAT-percentage and amount. Press "Submit".

Debit

Debiting can be made on a non-annulled payment and can be done either on selected order rows or the whole order.

You can debit in the following ways:

1. Select "Debit", (choose which order lines to finalize) and press "Submit"
2. Change from "Quantity" to "Amount" and manually type in your wanted description, VAT-percentage and amount. Press "Submit".

Credit

Crediting can be made on the part of the order rows which already has been debited, either partially or the whole order.

You can credit in the following ways:

1. Select "Credit", (choose which order lines to credit) and press "Submit"
2. Change from "Quantity" to "Amount" and manually type in your wanted description, VAT-percentage and amount. Press "Submit".

Additional authorization of payment

Search bar



Order example

OrderSpecification									
Order id: 200003									
Artikl	Beskrivning	Å pris inkl. moms	Moms	Summa	Värde moms	Antal	ÅRSLISTAD	KRISTEN	JÄNNILERA
							0.0	0.0	0.0
14612_M	Crosshatch Väst - M 57-58cm	790.00 kr	25 %	790.00 kr	173.00 kr	1.0	0.0	0.0	0.0
SHIPPING	DHL Service Point	80.00 kr	25 %	80.00 kr	16.00 kr	1.0	0.0	0.0	0.0
Total		870.00 kr		174.00 kr			0.00 kr	0.00 kr	0.00 kr

Additional authorization of payment

Lägg till artikel

ART.NR *	<input type="text"/>
BESKRIVNING *	<input type="text"/>
Å-PRIS EXKL. MOMS *	<input type="text"/>
MOMS *	<input type="text"/> 25
ANTAL *	<input type="text"/>
ENHET *	<input type="text"/>

AVBRYT **LÄGG TILL**

Additional authorization of payment

Normally, you cannot add an order row since this would mean that the total amount is above the initial approved amount. However, in some cases the merchant and Resurs Account manager have agreed upon setting an extralimit that is based on a percentage (or a fixed amount) of the initial amount. If so, you can add an order row to the existing payment instead of the customer making a completely new order for that one/few product-/s.

In addition, a new authorization can always be made if a product has been annulled by accident.

This feature **only** applies to Resurs Bank payment methods.

You can add an order row in the following ways:

1. Select "Add an article", put in the required fields and press "Add"
2. The order row is now authorized. If you want to debit the added order row, follow the instructions above.



Try it out here: [https://merchantportal.integration.resurs.com
/login](https://merchantportal.integration.resurs.com/login)

Search Payments in Merchant Portal

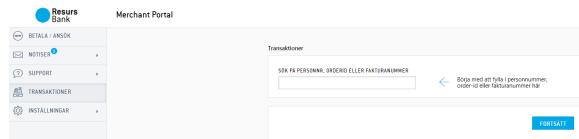
Search payments

To search for a specific order, simply enter the customers civic number (social security number), order-Id or invoice number.

The search bar is found under "Transactions" in the main bar to the left.

Sort by column

Information can easily be sorted by column. When you have the search bar visible, press "Continue" without entering any information in the bar. Then, simply press the desired column name once and the transactions are now sorted the way you requested. All columns are sortable.



Transaktioner							
Visa alla							
Transaktion	Datum	Persone nr	Namn	Köpkanal	Beträffatt	Belopp	Status
891941	2022-10-01 13:07	8350****	Vincent Williamson-Alex...	Faktura	5 000,00 kr	Debiterbar	
37025	2022-10-01 13:39	8350****	Vincent Williamson-Alex...	Faktura	5 000,00 kr	Debiterbar	
254078	2022-11-30 17:52	8350****	Vincent Williamson-Alex...	Revhuvande konto/kort	5 000,00 kr	Debiterbar	
57164	2022-11-29 09:17	8350****	Vincent Williamson-Alex...	Revhuvande konto/kort	5 000,00 kr	Debiterad	
20221129012	2022-11-29 08:27	5012****	Oliver Lennson Williams...	E-handel	Revhuvande konto	1 000,00 kr	Debiterbar
20221128011	2022-11-28 15:18	5012****	Oliver Lennson Williams...	E-handel	Revhuvande konto	10 000,00 kr	Debiterbar
202211281012	2022-11-28 15:14	8350****	Vincent Williamson-Alex...	E-handel	Befintlig kort	10 000,00 kr	Debiterbar
779762	2022-11-25 09:09	8350****	Vincent Williamson-Alex...	E-handel	Befintlig kort	5 000,00 kr	Debiterbar
146001	2022-11-25 09:06	8350****	Vincent Williamson-Alex...	E-handel	Befintlig kort	5 000,00 kr	Debiterbar
522009	2022-11-23 12:17	8350****	Vincent Williamson-Alex...	E-handel	Befintlig kort	5 000,00 kr	Debiterbar
990006	2022-11-23 10:11	8350****	Vincent Williamson-Alex...	E-handel	Befintlig kort	5 000,00 kr	Debiterbar

Configuration Service

Here you will find information about the settings of your web services such as authentication, etc.

See next invoice number - [peekInvoiceSequence](#)

To see the next invoice number to be used for automatic generation of invoice number, you can easily use the [peekInvoiceSequence](#) method which returns the next number as an integer.

Set next invoice number - [setInvoiceSequence](#)

To manually set the next invoice number that will be used for automatic generation of invoice number you can call the [setInvoiceSequence](#) method with desired invoice number.

Register event callback - [registerEventCallback](#)

To register callbacks.

Get event callback information - [Get registered callback](#)

To fetch information about a registered callback.

Unregister event Callback - [unregisterEventCallback](#)

To unregister callbacks.

Get registered callback

getRegisteredEventCallback

Retrieves a new event callback.

Input(Literal)

Name	Type	Occurs	Nillable?	Description
eventType	id	1..1	No	The type of event call-back being registered. Typical example is UNFREEZE for notification of frozen payments being thawed after manual fraud control. For full details on the call-back events available, please contact Resurs Bank.

Faults

Name	Content	Description
ECommerceErrorException	ECommerceError	Failed to register event callback. See error for details.

Callbacks

Read more about callbacks [here](#).

Example: getRegisteredEventCallback

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ns1="http://ecommerce.resurs.com/v4/msg/configuration">
    <SOAP-ENV:Body>
        <ns1:getRegisteredEventCallback>
            <eventType>BOOKED</eventType>
        </ns1:getRegisteredEventCallback>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Example HTML

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body>
        <ns3:getRegisteredEventCallbackResponse xmlns:ns3="http://ecommerce.resurs.com/v4/msg/configuration"
            xmlns:ns2="http://ecommerce.resurs.com/v4/msg/exception">
            <uriTemplate>https://www.netcurl.org/?callback=BOOKED&ts=1586864389</uriTemplate>
        </ns3:getRegisteredEventCallbackResponse>
    </soap:Body>
</soap:Envelope>
```

Peek Invoice Sequence

peekInvoiceSequence

Returns the next invoice number to be used for automatic generation of invoice numbers.

Output(Literal)

Name	Type	Occurs	Nillable?	Description
nextInvoiceNumber	integer	0..1	No	The next invoice number to be used for automatic generation of invoice numbers.

Faults

Name	Content	Description
ECommerceErrorException	ECommerceError	Failed to return invoice number sequence. See error for details

Introduction

This function returns the next invoice number that will be used for automatic generation of invoice number. You can change the invoice number manually by calling [setInvoiceSequence](#) with the wanted invoice number. You can do this from the [paymentAdmin](#), a web-based interface to handle payments and orders or by using the webservice.

Example

Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:con="http://ecommerce.resurs.com/v4/msg/configuration">
  <soapenv:Header/>
  <soapenv:Body>
    <con:peekInvoiceSequence/>
  </soapenv:Body>
</soapenv:Envelope>
```

Response

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns3:peekInvoiceSequenceResponse xmlns:ns3="http://ecommerce.resurs.com/v4/msg/configuration" xmlns:ns2="http://ecommerce.resurs.com/v4/msg/exception">
      <nextInvoiceNumber>444</nextInvoiceNumber>
    </ns3:peekInvoiceSequenceResponse>
  </soap:Body>
</soap:Envelope>
```

Register Event Callback

registerEventCallback

Registers a new event callback.

Input(Literal)

Name	Type	Occurs	Nillable?	Description
eventType	id	1..1	No	The type of event call-back being registered. Typical example is UNFREEZE for notification of frozen payments being thawed after manual fraud control. For full details on the call-back events available, please contact Resurs Bank.
uriTemplate	nonEmptyString	1..1	No	The call-back event URI template, with placeholders for the parameters to be returned. All placeholders are supplied in curly brackets, i.e. {} The actual placeholders depend on the type of call-back event. However, there is one common: digest. For further information on URLs and placeholders, please contact Resurs Bank. Example: http://www.resurs.se/?id={identifier}&rep=4&digest={digest}
basicAuthUserName	nonEmptyString	0..1	No	If Basic Access Authentication is to be used, the user name.
basicAuthPassword	nonEmptyString	0..1	No	If Basic Access Authentication is to be used, the password.
digestConfiguration	digest Configuration	0..1	No	If a digest is to be used to confirm that the call-back is actually issued by Resurs Bank, the configuration of that digest.

Faults

Name	Content	Description
ECommerceErrorException	ECommerceError	Failed to register event callback. See error for details.

Callbacks

Read more about callbacks [here](#).

Example

Request

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns3:registerEventCallback xmlns:ns2="http://ecommerce.resurs.com/v4/msg/exception" xmlns:ns3="http://ecommerce.resurs.com/v4/msg/configuration">
      <eventType>BOOKED</eventType>
      <uriTemplate>http://host.com/?wc-api=WC_Resurs_Bank&event-type=BOOKED&paymentId={paymentId}&digest={digest}</uriTemplate>
      <basicAuthUserName xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
      <basicAuthPassword xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
      <digestConfiguration>
        <digestAlgorithm>SHA1</digestAlgorithm>
        <digestParameters>paymentId</digestParameters>
        <digestSalt>11624530085b7bddc2646326</digestSalt>
      </digestConfiguration>
    </ns3:registerEventCallback>
  </soap:Body>
</soap:Envelope>
```

Set Invoice Data

setInvoiceData

Sets the representative data that will be used on the invoices. The data will be set once and will be used on all future invoices until the representative calls setInvoice again.

The invoice is generated after you finalize the payment and will contain the invoiceData and the speclines submitted in the finalization.

Input(Literal)

Name	Type	Occurs	Nillable?	Description
name	nonEmptyString	1..1	No	The representative's name.
street	nonEmptyString	1..1	No	The representative's street name.
zipcode	nonEmptyString	1..1	No	The representative's zip-code.
city	nonEmptyString	1..1	No	The representative's city.
country	nonEmptyString	1..1	No	The representative's country.
phone	nonEmptyString	1..1	No	The representative's phone number.
fax	nonEmptyString	0..1	No	The representative's fax.
email	nonEmptyString	1..1	No	The representative's e-mail.
homepage	nonEmptyString	1..1	No	The representative's homepage.
vatreg	nonEmptyString	1..1	No	The representative's vat registration number.
orgno	nonEmptyString	1..1	No	The representative's organization number.
companystaxnote	boolean	1..1	No	Whether or not the company has a company tax note (Not available in Norway).
logotype	base64Binary	1..1	No	The representative's logotype.
modifiedby	string	1..1	No	The name of the user that last modified the data.

Set Invoice Sequence

setInvoiceSequence

Sets the next invoice number to be used for automatic generation of invoice numbers.

Input(Literal)

Name	Type	Occurs	Nillable?	Description
nextInvoiceNumber	integer	0..1	No	The next invoice number to be used for automatic generation of invoice numbers.

Faults

Name	Content	Description
ECommerceErrorException	ECommerceError	Failed to set the invoice number sequence. See error for details.

Introduction

You can set the next number to be used for invoices. The new number can not be lower than the existing, it must be a higher unused number. For example if the current invoice number is 223 you can't set the next invoice number to be 220. You can do this operation from the [paymentAdmin](#), a web-based interface to handle payments and orders or by using the webservice

Example

Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:con="http://ecommerce.resurs.com/v4/msg/configuration">
    <soapenv:Header/>
    <soapenv:Body>
        <con:setInvoiceSequence>
            <nextInvoiceNumber>445</nextInvoiceNumber>
        </con:setInvoiceSequence>
    </soapenv:Body>
</soapenv:Envelope>
```

Response - void

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body>
        <ns3:setInvoiceSequenceResponse xmlns:ns3="http://ecommerce.resurs.com/v4/msg/configuration" xmlns:ns2="http://ecommerce.resurs.com/v4/msg/exception"/>
    </soap:Body>
</soap:Envelope>
```

Example Error

When trying to set the invoice number lower or equal to the previous number

Error

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body>
        <soap:Fault>
            <faultcode>soap:Server</faultcode>
            <faultstring>Setting a invoice number lower than last number is not allowed</faultstring>
            <detail>
```

```
<ns2:ECommerceError xmlns:ns2="http://ecommerce.resurs.com/v4/msg/exception" xmlns:ns3="http://ecommerce.resurs.com/v4/msg/configuration">
    <errorTypeDescription>ILLEGAL_ARGUMENT</errorTypeDescription>
    <errorTypeId>1</errorTypeId>
    <fixableByYou>true</fixableByYou>
    <userErrorMessage>Startnummer för faktura får inte vara lägre än tidigare nummer</userErrorMessage>
</ns2:ECommerceError>
</detail>
</soap:Fault>
</soap:Body>
</soap:Envelope>
```

Unregister Event Callback

unregisterEventCallback

Registers an existing event callback.

Input(Literal)

Name	Type	Occurs	Nillable?	Description
eventType	id	1..1	No	

Faults

Name	Content	Description
ECommerceErrorException	ECommerceError	Failed to unregister event callback. See error for details.

Introduction

This method is to unregister a callback event. For more on how this is done please read about Callbacks [here](#).

Developer Service

The Developer WebService provides miscellaneous services that can be useful during the representative integration. The methods provided in here are not for use during normal service operation and are only available for simulation.

Get limit application template - [getLimitApplicationTemplate](#)

Retrieves the limit application response template for local use.

Is agreement signed - [isSigned](#)

Determines whether or not a specific agreement has been successfully signed.

Trigger event - [triggerEvent](#)

Triggers a test event. This is for testing the callback functionality.

getLimitApplicationTemplate

getLimitApplicationTemplate

Retrieves the limit application response template. Normally this is not necessary, as the form creates this itself. However, if the representative decides to implement the form functionality locally, without using the features provided by Resurs Bank, this method will show the format of a valid response.

Input (Literal)

Name	Type	Occurs	Nillable?	Description
paymentMethodId	id	1..1	No	The identity of the payment method
sum	positiveDecimal	1..1	No	The limit amount.

Output (Literal)

Name	Type	Occurs	Nillable?	Description
return	string	1..1	No	The limit application form response template.

Faults

Name	Content	Description
ECommerceErrorException	ECommerceError	Failed to retrieve the limit application response template. See error for details.

isSigned

isSigned

Determines whether or not a specific agreement has been successfully signed.

Input (Literal)

Name	Type	Occurs	Nillable?	Description
signingId	id	1..1	No	The identity of the signing session. Please note that this has to be from the signing URL directly.

Output (Literal)

Name	Type	Occurs	Nillable?	Description
return	boolean	1..1	No	Whether or not the agreement was signed.

Faults

Name	Content	Description
ECommerceErrorException	ECommerceError	Failed to determine whether or not the agreement was successfully signed. See error for details.

Introduction

A function to see if the [signing](#) has been made. The [signing](#) is for completing a purchase when Resurs Bank says [signing](#) is required.

triggerEvent

triggerEvent

Triggers a test event. This is for testing the callback functionality.

Input (Literal)

Name	Type	Occurs	Nillable?	Description
eventType	id	1..1	No	The type of event to be triggered.
param	string	0..*	No	The data to be used when triggering the event.

Output (Literal)

None

Faults

Name	Content	Description
ECommerceErrorException	ECommerceError	Failed to trigger the specified event. See error for details.

For available callback-types please see the [callback page](#) and scroll down to Available callback-type

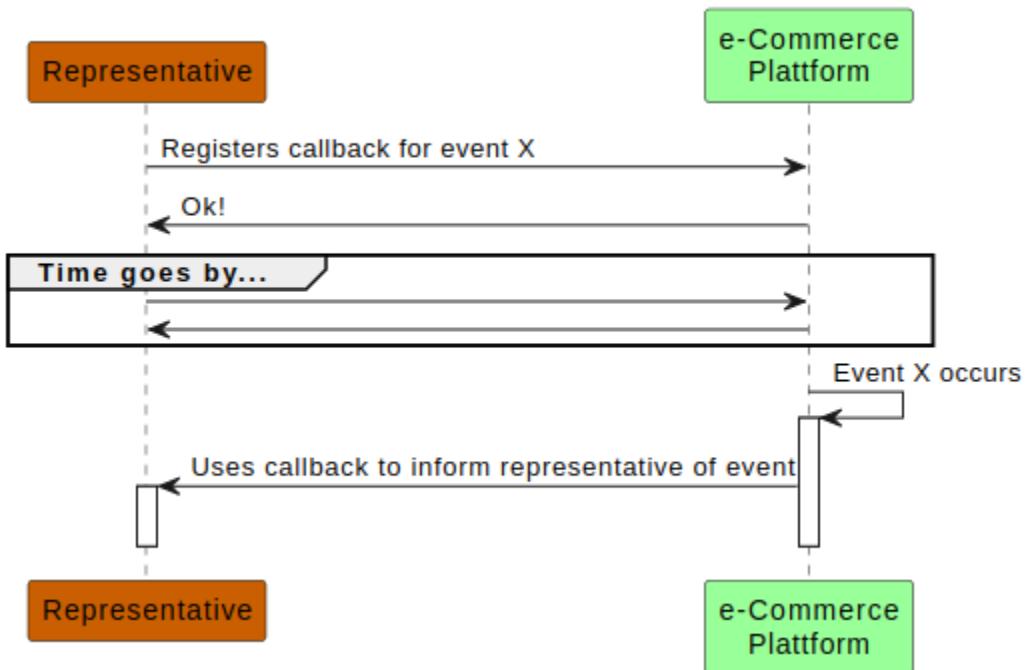
Callbacks

- Status and callback chart
- Available Callbacks
- What's a callback?
- Digest and checksum
- Register Callback
- Authentication
- Fallback - resending failed callbacks

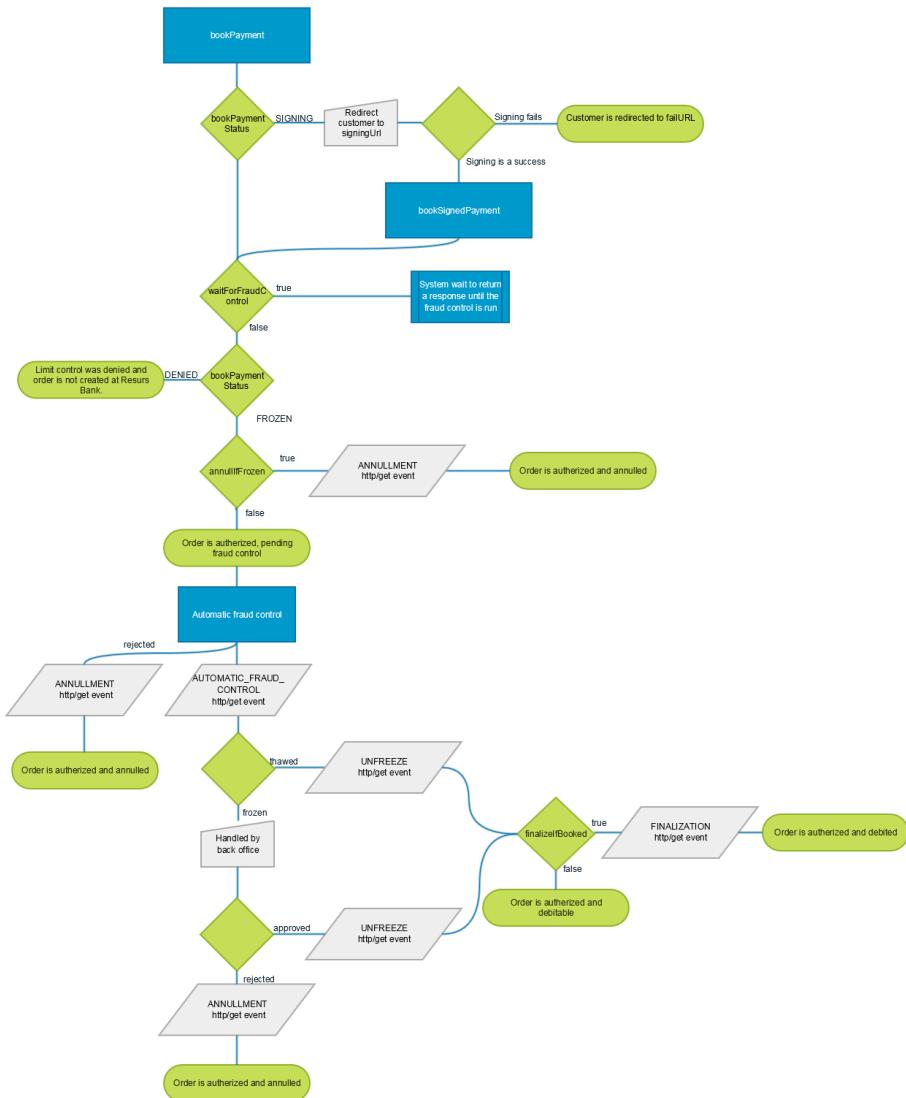
User-Agent notice

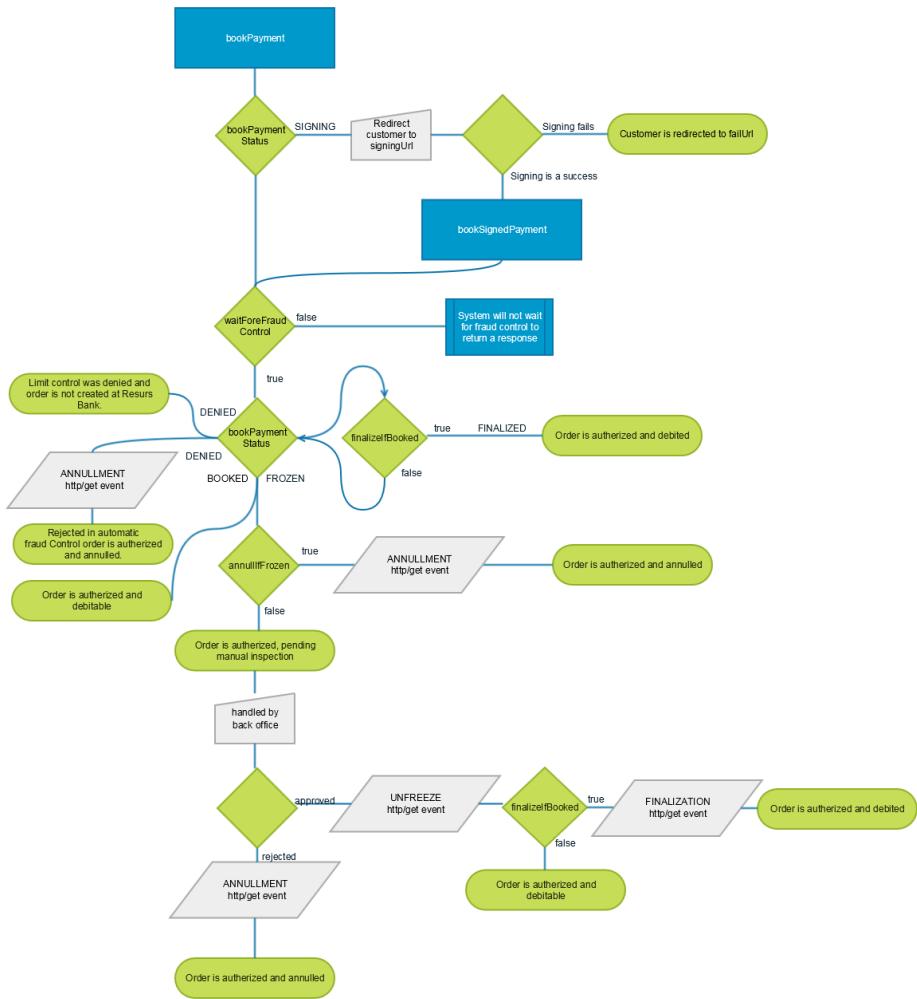
Generally callbacks are also sent with two different user-agents, to secure that callbacks are really delivered. Some hosting providers usually blocks traffic from clients that are identified as a Java client.

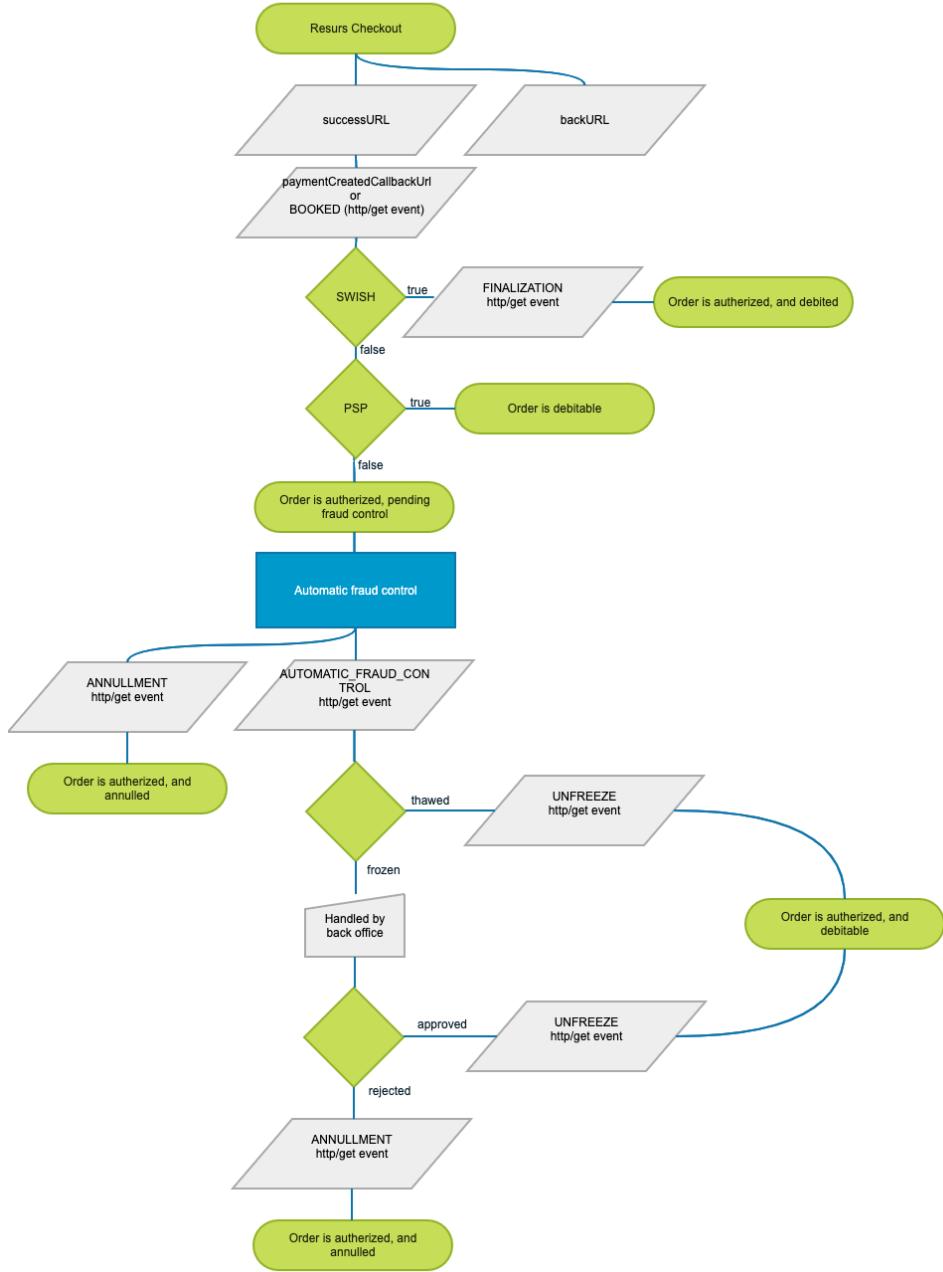
Polling the e-commerce service for status changes would prevent scaling of the service, and it would prevent efficient handling. The answer is callbacks, simple HTTP calls initiated by Resurs Bank.

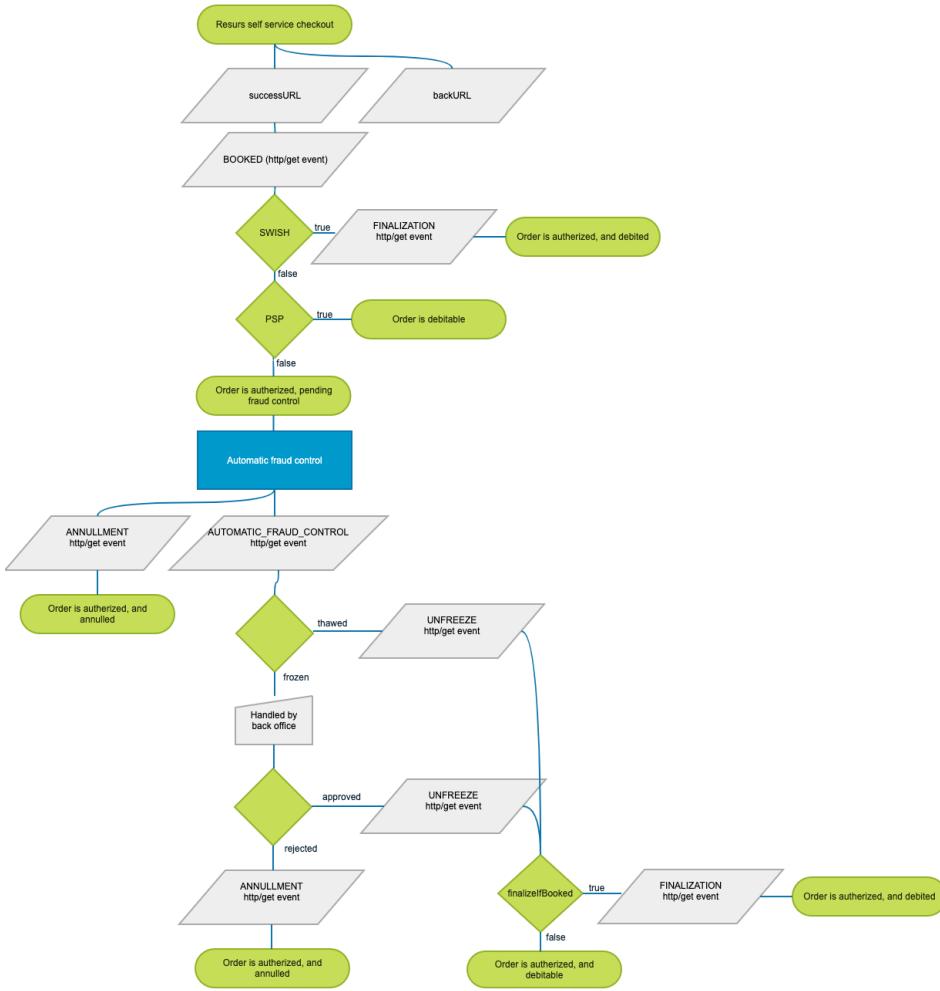


Status and callback chart









Available Callbacks

- **BOOKED** - The order is created in Resurs Bank system. (Note! Order can be in a FROZEN state.)
- **AUTOMATIC_FRAUD_CONTROL** - A payment has automatically been fraud screened
- **UNFREEZE** - A payment is "thawed" from a FROZEN status.
- **ANNULMENT** - A payment is fully annulled.
- **FINALIZATION** - Resurs Bank finalizes a payment. (Note! Only applicable if finalizedIfBooked parameter is set to true.)
- **UPDATE** - A payment is updated.
- **TEST** - Is used to test the callback machinery

What's a callback?

A callback is a simple HTTP call **from** Resurs Bank **to** the representative (not necessary the shop). This is in opposite direction of the "normal" communication, where the representative makes calls to Resurs Bank. The main reason to resort to a callback solution is that it's not possible in practicality for the representative to poll for different status changes in the e-commerce service in most cases. (It is reminiscent of [Paypals IPN](#))

The representative *registers* a callback URL with the web service. When the event occurs, Resurs Bank will make a simple HTTP/GET call with query parameters (defined by the callback type) to that URL. If a status below 300 is returned to Resurs Bank that event will be considered delivered, otherwise several more tries will be carried out.



- When receiving an event, please return an OK status (HTTP 2xx). Otherwise the e-commerce service will continue to try to inform until the timeout is reached.
- We do not, for the time being, follow redirects correctly (302, 303)

A callback type is defined by an identity that is a common string. This allows us to easily create new callbacks without breaking the interface. The callback can have any number of parameters. It is the information to be communicated to the representative when a callback event occurs. The Representative register a URI that Resurs Bank will call, with placeholders for each parameter.

Here's an example of a callback URI

```
https://ws.host.com/Resursbank.aspx?orderId={paymentId}&status=UNFREEZE&partnerId=Resursbank&digest={digest}
```

Apparently, this event-type has two parameters: *paymentId* and *digest*. Read more about digest and checksums below.

When the call is done, we take the URI, and populate the placeholders with the actual values, and makes the call. All callbacks are HTTP/GET's with the exception of BOOKED and UPDATE that are HTTP/POST's. If we get a status that is less than 300 back, we consider the event conveyed. Otherwise, it lands in a queue with a fall-off mechanism: The gap between callback attempts increases until a month has passed, after which the callback is lost.

Digest and checksum

A representative can also request a checksum attached as a parameter in the call. This parameter is called *digest* (always, regardless of event type) and the agent specifies the parameters to be included in the checksum. He also indicates which algorithm is used (SHA-1 or MD5), and possibly a *salt*.

If we use the above URI such as:

```
SALT: iCanHasCheezeburger
ALGORITHM: MD5
CALLBACK URI: https://ws.host.com/Resursbank.aspx?orderId={paymentId}
&status=UNFREEZE&partnerId=Resursbank&digest={digest}
PARAMETERS WHICH SHOULD BE USED IN CHECKSUM CALCULATION: paymentId
paymentId: lePayment
```

This whould give the following string: lePaymentiCanHasCheezeburger
MD5 for this string is ED3381936CCAA2659CF3089F4AA83007

This would result in a call to

```
https://ws.host.com/Resursbank.aspx?
orderId=lePayment&status=UNFREEZE&partnerId=Resursbank&digest=ED3381936CCAA2659CF3089F4AA83007
```



The checksum is in **UPPERCASE!**

Register Callback

You can use either [SOAP](#) or [REST](#) to register callbacks.

Authentication

When the representative registers a callback, it's optional to provide a username and password. These will be used by Resurs Bank to perform a Basic Authentication at the merchant's site when doing a callback.

Alternatively is your site is open for calls.

Other authentication methods are not supported. Please note that the usage of a salt will be more secure if the channel isn't encrypted (SSL/TLS) (since basic authentication is plain text). For extra safety you can send in an unique SALT with every call.

Fallback - resending failed callbacks

When we try to send a callback and it fails, either if we can't contact the callback service or the correct status aren't returned, we will try to resend it 19 times, after the first failed one.

Attempt	Delay between attempts
1	0
2	30 sec
3	45 sec
4	1 min

5	1,5 min
6	2,5 min
7	4 min
8	5,5 min
9	8,5 min
10	13 min
11	20 min
12	30 min
13	45 min
14	1 h
15	1,5 h
16	2,5 h
17	4 h
18	5 h
19	8 h
20	12 h(total 36 h)

ANNULMENT

	Name	Description
ID	ANNULMENT	The annulment event ID.
Parameter	paymentId	Payment ID (was sent to us as <code>preferredPaymentId</code> in the bookPayment call)

Trigger

Will be sent once a payment is *fully* annulled at Resurs Bank, for example when manual fraud screening implies fraudulent usage. Annulling part of the payment *will not* trigger this event.

! If the representative is not listening to this callback orders might be orphaned (i.e. without connected payment) and products bound to these orders never released.

Example

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
    <ns0:registerEventCallback xmlns:ns0="http://ecommerce.resurs.com/v4/msg/configuration" xmlns:ns1="http://ecommerce.resurs.com/v4/msg/exception">
      <eventType>ANNULMENT</eventType>
      <uriTemplate>https://host.com/rest/resursbank_checkout/annulment/paymentId/{paymentId}/digest/{digest}</uriTemplate>
      <basicAuthUserName>UserNameIfBasicAccessAuthenticationIsUsed</basicAuthUserName>
      <basicAuthPassword>PasswordIfBasicAccessAuthenticationIsUsed</basicAuthPassword>
      <digestConfiguration>
        <digestAlgorithm>SHA1</digestAlgorithm>
        <digestParameters>paymentId</digestParameters>
        <digestSalt>SecretHashSalt</digestSalt>
      </digestConfiguration>
    </ns0:registerEventCallback>
  </S:Body>
</S:Envelope>
```

Resurs Bank will make a HTTP/GET call with query parameters (defined by the callback type) to the registered URL: <https://host.com/?merchant=ResursBank&event-type=ANNULMENT&paymentId=10000016&digest=4b6f2ddec947e6e5b6c4c3998081f3d7bce1f40d>

AUTOMATIC_FRAUD_CONTROL

Occurs when the automatic fraud control has been preformed. This will occur seconds after the bookPayment call.

If the shop needs this information the representative can either listen for this callback or run bookPayment in synchronous mode.

	Name	Description
ID	AUTOMATIC_FRAUD_CONTROL	The automatic fraud control event ID.
Parameter	paymentId	Payment ID (was sent to us as <code>preferredPaymentId</code> in the bookPayment call)
Parameter	result	The result of the automatic fraud control. Either FROZEN or THAWED.

Example

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
    <ns0:registerEventCallback xmlns:ns0="http://ecommerce.resurs.com/v4/msg/configuration" xmlns:ns1=""
http://ecommerce.resurs.com/v4/msg/exception">
      <eventType>AUTOMATIC_FRAUD_CONTROL</eventType>
      <uriTemplate>https://host.com/rest/resursbank_checkout/automatic_fraud_control/paymentId/{paymentId}
/result/{result}/digest/{digest}</uriTemplate>
      <basicAuthUserName>UserNameIfBasicAccessAuthenticationIsUsed</basicAuthUserName>
      <basicAuthPassword>PasswordIfBasicAccessAuthenticationIsUsed</basicAuthPassword>
      <digestConfiguration>
        <digestAlgorithm>SHA1</digestAlgorithm>
        <digestParameters>paymentId</digestParameters>
        <digestSalt>SecretHashSalt</digestSalt>
      </digestConfiguration>
    </ns0:registerEventCallback>
  </S:Body>
</S:Envelope>
```

Resurs Bank will make a HTTP/GET call with query parameters (defined by the callback type) to the registered URL: https://host.com/?merchant=ResursBank&event-type=AUTOMATIC_FRAUD_CONTROL&paymentId=10000016&result=FROZEN&digest=4b6f2ddec947e6e5b6c4c3998081f3d7bce1f40d

BOOKED

[Trigger] [Callback with dual requests]

	Name	Description
ID	BOOKED	The booking event ID.
Parameter	paymentId	Payment ID (was sent to us as <code>preferredPaymentId</code> in the <code>bookPayment</code> call or <code>orderReference</code> in the POST for Resurs Checkout)

Trigger

The order has been created in Resurs Bank system.

Resurs Bank will do a both a conditional POST and GET request call to a system with this callback registered. By means, if the POST request is successful, the GET won't fire up.

Parameters added to the URL is `paymentId` and the JSON below for `paymentSpec` (when the callback is based on method POST).

Example registerEventCallback

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
    <ns0:registerEventCallback xmlns:ns0="http://ecommerce.resurs.com/v4/msg/configuration" xmlns:ns1="http://ecommerce.resurs.com/v4/msg/exception">
      <eventType>BOOKED</eventType>
      <uriTemplate>https://host.com/rest/resursbank_checkout/booked/paymentId/{paymentId}/digest/{digest}</uriTemplate>
      <basicAuthUserName>UserNameIfBasicAccessAuthenticationIsUsed</basicAuthUserName>
      <basicAuthPassword>PasswordIfBasicAccessAuthenticationIsUsed</basicAuthPassword>
      <digestConfiguration>
        <digestAlgorithm>SHA1</digestAlgorithm>
        <digestParameters>paymentId</digestParameters>
        <digestSalt>SecretHashSalt</digestSalt>
      </digestConfiguration>
    </ns0:registerEventCallback>
  </S:Body>
</S:Envelope>
```

Resurs Bank will make a HTTP/POST call: https://host.com/rest/resursbank_checkout/booked/paymentId/11111111/digest/digest123xyz



Content-Type application/Json

Example of a POST call

```
{
  "addedPaymentSpecificationLines": []
}

--* Or like this where fee is added in Resurs payment administration gui*--
{
  "addedPaymentSpecificationLines": [
    {
      "id": "99999999",
      "artNo": "fff_999",
      "description": "Invoice fee",
      "quantity": 1,
      "unitMeasure": "pcs",
      "unitAmountWithoutVat": 16.00000,
      "vatPct": 25,
      "totalVatAmount": 4.00000,
```

```
        "totalAmount": 20.00000
    }
}
```

Callback with dual requests

The dual calls from this callback that may be observed in webserver logs is much about not breaking compatibility, where POST is a bit modern than the GET.

Usually there's a regular GET for the BOOKED with no parameters inside. The only thing it should do is to report that the order is booked.

The second POST is delivered with extra fees if they are present as shown above (this variant is currently unsupported by for example the Magento plugin).

Generally callbacks are also sent with two different user-agents, to secure that callbacks are really delivered. Some hosting providers usually blocks traffic from clients that are identified as a Java client.

FINALIZATION



This event is triggered when a payment is automatically finalized at Resurs Bank.
i.e.:

- finalizelfBooked set to TRUE in bookPayment method (Simplified Shop Flow Service and Resurs Hosted flow).
- Payment type is Swish (SE)

Once a payment is finalized automatically at Resurs Bank, this will trigger this event, when the parameter finalizelfBooked parameter is set to true in `paymentData`.

	Name	Description
ID	FINALIZATION	The finalization event ID.
Parameter	paymentId	Payment ID (was sent to us as <code>preferredPaymentId</code> in the bookPayment call)

Example

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
    <ns0:registerEventCallback xmlns:ns0="http://ecommerce.resurs.com/v4/msg/configuration" xmlns:ns1="http://ecommerce.resurs.com/v4/msg/exception">
      <eventType>FINALIZATION</eventType>
      <uriTemplate>https://host.com/rest/resursbank_checkout/finalization/paymentId/{paymentId}/digest/{digest}</uriTemplate>
      <basicAuthUserName>UserNameIfBasicAccessAuthenticationIsUsed</basicAuthUserName>
      <basicAuthPassword>PasswordIfBasicAccessAuthenticationIsUsed</basicAuthPassword>
      <digestConfiguration>
        <digestAlgorithm>SHA1</digestAlgorithm>
        <digestParameters>paymentId</digestParameters>
        <digestSalt>SecretHashSalt</digestSalt>
      </digestConfiguration>
    </ns0:registerEventCallback>
  </S:Body>
</S:Envelope>
```

Resurs Bank will make a HTTP/GET call with query parameters (defined by the callback type) to the registered URL: <https://host.com/?merchant=ResursBank&event-type=FINALIZATION&paymentId=10000016&digest=4b6f2ddec947e6e5b6c4c3998081f3d7bce1f40d>

Parameters and Callbacks

Depending on what value you use for the parameter `waitForFraudControl` in `bookPayment` you will have to implement different callbacks, if you are not monitoring the orders manually with the web tool [Payment Administration](#). The answer you get from `bookPayment` will differ depending on if you use the Simplified flow or the Hosted flow (the latter also explained [here](#)).

SIMPLIFIED FLOW

waitForFraudControl=FALSE

With this setting in the call `bookPayment` you run asynchronously and do not wait for the answer of the control. In `bookPaymentResponse` you will instead get an intermediate status `FROZEN` (in the old deprecated flow you will get `CONTROL_IN_PROGRESS`) or you will get `DENIED`.

When the automatic control has finished the order will get one of these statuses:

- `BOOKED` in the system - the control shows nothing wrong and you can debit the order. The callback `AUTOMATIC_FRAUD_CONTROL` is sent and says `THAWED`.
- `FROZEN` in the system - something is suspicious and the order goes to manual handling. The callback `AUTOMATIC_FRAUD_CONTROL` is sent and says `FROZEN`.

When an order is manually inspected it's also here two outcomes, both send their own callback that you have to listen to:

- The order is okay for debiting - this callback is sent: `UNFREEZE`.
- Something is wrong or suspicious - this callback is sent: `ANNULMENT`.

waitForFraudControl=TRUE

With this setting in the call `bookPayment` you run synchronously and wait for the control to run. In `bookPaymentResponse` you will get one of these statuses:

- `BOOKED` - the control shows nothing and you can debit the order.
- `FROZEN` - something is suspicious and it will be a manual inspection.

When an order is manually inspected it's also here two outcomes, both send their own callback that you have to listen to:

- The order is okay for debiting - this callback is sent: `UNFREEZE`.
- Something is wrong or suspicious - this callback is sent: `ANNULMENT`.

Our callbacks: <https://test.resurs.com/docs/x/LAAF>

HOSTED FLOW

waitForFraudControl=FALSE

With this setting in the call `bookPayment` you run asynchronously and do not wait for the answer of the control. The order will get an intermediate status `FROZEN` and you will only get back your `successUrl`.

When the automatic control has finished the order will get one of these statuses:

- `BOOKED` in the system - the control shows nothing wrong and you can debit the order. The callback `AUTOMATIC_FRAUD_CONTROL` is sent and says `THAWED`.
- `FROZEN` in the system - something is suspicious and the order goes to manual handling. The callback `AUTOMATIC_FRAUD_CONTROL` is sent and says `FROZEN`.

When an order is manually inspected it's also here two outcomes, both send their own callback that you have to listen to:

- The order is okay for debiting - this callback is sent: `UNFREEZE`.
- Something is wrong or suspicious - this callback is sent: `ANNULMENT`.

waitForFraudControl=TRUE

With this setting in the call `bookPayment` you run synchronously and wait for the control which will set the order in one of two statuses: You will only get your `successUrl` back and have to listen to callback `AUTOMATIC_FRAUD_CONTROL` to know which status the control has set the order in out of these two:

- `BOOKED` - the control shows nothing and you can debit the order.
- `FROZEN` - something is suspicious and it will be a manual inspection.

When an order is manually inspected it's also here two outcomes, both send their own callback that you have to listen to:

- The order is okay for debiting - this callback is sent: `UNFREEZE`.
- Something is wrong or suspicious - this callback is sent: `ANNULMENT`.

If you use the payment method "existing card" the automatic fraud control is not run, a control was made when you applied for the card.
This means that you can regard the successUrl as a successful booking if that is returned.
You might still be denied to place the order due to using a higher amount than your are allowed to.

Our callbacks: <https://test.resurs.com/docs/x/LAAF>

TEST

Test the callback mechanism. Can be used in integration testing to assure that communication works. A call is made to [DeveloperService \(triggerTestEvent\)](#) and Resurs Bank immediately does a callback. Note that TEST callback must be registered in the same way as all the other callbacks before it can be used.

	Name	Description
ID	TEST	The test event ID.
Parameter	param1, param2, param3, param4, param5	Parameters to test a successful roundtrip (see below)

Using the test callback

If a representative calls `DeveloperService.triggerTestEvent(...)` with `param1 = 1` `param2 = 2`, `param3 = 3`, `param4 = 4`, `param5 = 5`, and he has previously registered the following URL:

```
http://hosten.se/kontexten/funktionen.html?first={param1}&nr2={param2}&nr3={param3}&nr4={param4}&last={param5}
```

with no digest or checksum, then he will get a call to:

```
http://hosten.se/kontexten/funktionen.html?first=1&nr2=2&nr3=3&nr4=4&last=5
```



Note that the test event is placed in the queue, just like all other events. If they can not be delivered immediately, the system continues to try for a month ...

Example

```
<!--Register callback TEST-->
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:con="http://ecommerce.resurs.com/v4/msg/configure">
    <soapenv:Header/>
    <soapenv:Body>
        <con1:registerEventCallback xmlns:con1="http://ecommerce.resurs.com/v4/msg/configuration">
            <eventType xmlns="">TEST</eventType>
            <uriTemplate xmlns="">http://host.com/?merchant=ResursBank&event-type=TEST&par1={param1}&par2={param2}&par3={param3}&par4={param4}&par5={param5}</uriTemplate>
        </con1:registerEventCallback>
    </soapenv:Body>
</soapenv:Envelope>

<!--Get callback TEST-->
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:con="http://ecommerce.resurs.com/v4/msg/configuration">
    <soapenv:Header/>
    <soapenv:Body>
        <con:getRegisteredEventCallback>
            <eventType>TEST</eventType>
        </con:getRegisteredEventCallback>
    </soapenv:Body>
</soapenv:Envelope>

<!--Trigger callback TEST-->
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:dev="http://ecommerce.resurs.com/v4/msg/developer">
    <soapenv:Header/>
    <soapenv:Body>
        <dev:triggerEvent>
            <eventType>TEST</eventType>
            <param>Alfa</param>
            <param>Bravo</param>
            <param>Charlie</param>
            <param>Delta</param>
            <param>Echo</param>
        </dev:triggerEvent>
    </soapenv:Body>
</soapenv:Envelope>
```

```
</dev:triggerEvent>
</soapenv:Body>
</soapenv:Envelope>

<!--Result: http://host.com/?merchant=ResursBank&event-
type=TEST&par1=Alfa&par2=Bravo&par3=Charlie&par4=Delta&par5=Echo -->

<!--Remove callback TEST-->
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:con="http://ecommerce.resurs.
com/v4/msg/configure">
    <soapenv:Header/>
    <soapenv:Body>
        <con1:unregisterEventCallback xmlns:con1="http://ecommerce.resurs.com/v4/msg/configuration">
            <eventType>TEST</eventType>
        </con1:unregisterEventCallback>
    </soapenv:Body>
</soapenv:Envelope>
```

UNFREEZE

Informs when a payment is unfrozen after manual fraud screening. This means that the payment may be debited (captured) and the goods can be delivered.

	Name	Description
ID	UNFREEZE	The unfreeze event ID.
Parameter	paymentId	Payment ID (was sent to us as <code>preferredPaymentId</code> in the <code>bookPayment</code> call)

Example

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
    <ns0:registerEventCallback xmlns:ns0="http://ecommerce.resurs.com/v4/msg/configuration" xmlns:ns1="http://ecommerce.resurs.com/v4/msg/exception">
      <eventType>UNFREEZE</eventType>
      <uriTemplate>https://host.com/rest/resursbank_checkout/unfreeze/paymentId/{paymentId}/digest/{digest}</uriTemplate>
      <basicAuthUserName>UserNameIfBasicAccessAuthenticationIsUsed</basicAuthUserName>
      <basicAuthPassword>PasswordIfBasicAccessAuthenticationIsUsed</basicAuthPassword>
      <digestConfiguration>
        <digestAlgorithm>SHA1</digestAlgorithm>
        <digestParameters>paymentId</digestParameters>
        <digestSalt>SecretHashSalt</digestSalt>
      </digestConfiguration>
    </ns0:registerEventCallback>
  </S:Body>
</S:Envelope>
```

Resurs Bank will make a HTTP/GET call with query parameters (defined by the callback type) to the registered URL: <https://host.com/?merchant=ResursBank&event-type=UNFREEZE&paymentId=10000016&digest=4b6f2ddec947e6e5b6c4c3998081f3d7bce1f40d>

UPDATE

	Name	Description
ID	UPDATE	The update event ID.
Parameter	paymentId	Payment ID (was sent to us as <code>preferredPaymentId</code> in the bookPayment call)

Trigger

Will be sent when a payment is updated.

Resurs Bank will do a HTTP/POST call with parameter `paymentId` and the JSON for `paymentDiff` to the registered URL.

Example registerEventCallback

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
    <ns0:registerEventCallback xmlns:ns0="http://ecommerce.resurs.com/v4/msg/configuration" xmlns:ns1="http://ecommerce.resurs.com/v4/msg/exception">
      <eventType>UPDATE</eventType>
      <uriTemplate>https://host.com/rest/resursbank_checkout/update/paymentId/{paymentId}/digest/{digest}</uriTemplate>
      <basicAuthUserName>UserNameIfBasicAccessAuthenticationIsUsed</basicAuthUserName>
      <basicAuthPassword>PasswordIfBasicAccessAuthenticationIsUsed</basicAuthPassword>
      <digestConfiguration>
        <digestAlgorithm>SHA1</digestAlgorithm>
        <digestParameters>paymentId</digestParameters>
        <digestSalt>SecretHashSalt</digestSalt>
      </digestConfiguration>
    </ns0:registerEventCallback>
  </S:Body>
</S:Envelope>
```

Resurs Bank will make a HTTP/POST call: https://host.com/rest/resursbank_checkout/update/paymentId/10000016/digest/C60345B6E58FD0B363FD2904A39EBB03442CF778

Example of a POST call

```
--* Example 1 *--
{
  "paymentDiff": {
    "type": "ANNUL",
    "transactionId": null,
    "created": 1473258094600,
    "createdBy": null,
    "paymentSpecification": {
      "paymentSpecificationLines": null,
      "totalAmount": 24,
      "totalVatAmount": 6,
      "bonusPoints": 0
    },
    "orderId": null,
    "invoiceId": null
  }
}

--* Exempel 2 *--
{
  "paymentDiff": {
    "type": "DEBIT",
    "transactionId": "TrD-1473258203595-2404",
    "created": 1473258212019,
```

```
"createdBy": "B User",
"paymentSpecification": {
    "paymentSpecificationLines": [
        {
            "id": "1",
            "artNo": "NUT-001",
            "description": "Nut (M8)",
            "quantity": 10,
            "unitMeasure": "st",
            "unitAmountWithoutVat": 0.8,
            "vatPct": 25,
            "totalVatAmount": 2,
            "totalAmount": 8
        },
        {
            "id": "2",
            "artNo": "BOLT-002",
            "description": "Bolt (M8x125mm)",
            "quantity": 10,
            "unitMeasure": "st",
            "unitAmountWithoutVat": 1.6,
            "vatPct": 25,
            "totalVatAmount": 4,
            "totalAmount": 16
        }
    ],
    "totalAmount": null,
    "totalVatAmount": 6,
    "bonusPoints": 0
},
"orderId": "Ord-1473258203595-2404",
"invoiceId": null
}
}
```

Development

Getting started with self made integrations.

Table of contents

Table of contents

- [Downloadables](#)
- [Source code and support](#)
- [Heads up, please!](#)

Subpages

- [Errors, problem solving and corner cases](#)
- [General: Integration development](#)
- [PHP and development libraries](#)
- [API Types](#)
- [Create part payment widget](#)
- [Customer data - Regular expressions](#)
- [Rounding](#)
- [Recognized metadata](#)
- [Payment providers in simplified flow](#)
- [Permissions and passwords - Platform access](#)



Important about staging versus production

If you are planning to develop (or install) plugins, it is **required** that you are running both a test and a production environment. In this way you will first be able to test that everything work properly in a controlled environment before a production release.

If this is the case, also **make sure that your test and production environment are identical**. By doing this you will minimize the risk to have different results in your test compared to your production. For example: Running PHP 7.0 in test and an older version like PHP 5.6 or even 5.4, in production will probably generate unexpected results depending on the product you are using.

Downloadables

We're hosting a bunch of logos and images for our markdown documents that follows in our bitbucket-repos via <https://test.resurs.com/plugindevelopers/>

Source code and support

We use [Bitbucket](#) and [JIRA](#).

You can find our public source code repositories at <https://bitbucket.org/resursbankplugins/?visibility=public>.

To report any problems and view our current projects and issues, please visit JIRA at <https://resursbankplugins.atlassian.net/secure/Dashboard.jspa>.

Heads up, please!

Are you building your platform with callbacks? Be cautious; callbacks can be sent to your platform **rapidly**. By means, several callbacks can arrive to your platform the exact same time, causing race conditions in your platform - either in your backend parts or directly with your database. If you are fortunate, callbacks will be handled separately in a "standard" plugin. However, sometimes the best practice is to receive a callback and then handle them with a separate queue since database updates may not be fast enough to have the proper data when it is time to handle the next request.

Errors, problem solving and corner cases

Table of contents

- [How to handle errors](#)
- [EComPHP custom errorcodes](#)
- [Error handling \(Resurs error codes\)](#)

This section covers known problems, fixes for them, workarounds and other things related to solving problems.



Using ngrok or proxies?

Do not use publicly available proxies for your developing, like ngrok, and sites that are known to also spread malicious code and software as they normally is banned in our firewall.
This covers callbacks, but could affect other functions too.
One main reason could be that you are using a host or domain name that is flagged malicious in our systems. Ngrok is one of those that is usually scored to be blocked in outbound communications. If is highly recommended that you don't use such sites.

How to handle errors

As we do have more flows than one, with different "calling techniques" error handling may vary a bit between the flows. In its simplest way, you can say that HTTP-HEAD responses partially covers exceptions, however by only listening to a code from the http header may not reveal the cause many times. EComPHP covers most of the error handling in different steps, and an advice is to do something similar in other software too - if possible.

For example, SOAP errors can be handled by checking both head and body. Some exceptions sends a http head code above 500 (internal server error, and similar). This might give a part of the error, so in this case, you should consider checking for SoapExceptions too. In th body, details about the errors usually reside in the detail element, under an ECommerceError tag. Extracting this value could give you further information about the error. Such errors are documented at [Error handling \(Resurs error codes\)](#). EComPHP extracts this kind of errors to find the root cause, and when unable to find an error it uses its own - see [EComPHP custom errorcodes](#).

For the rest section, you could normally just check the http head responses.

EComPHP custom errorcodes

EComPHP, when errors occurs, renders own fault codes when something goes wrong. A sample of what codes that is used can be read from this point. A part of this list includes the SOAP faultcodes [here](#).

Except for this EComPHP passes errors from the engine that handles the communication with ecommerce rest/soap. Those exceptions can be found [here](#) and will not be included here as third party applications might change over time. This list does not cover the section ECOMMERCEERROR either, as that section covers external errors for Resurs Ecommerce.

CODE	CONSTANT	DESCRIPTION
1000	NOT_IMPLEMENTED	General errors
1006	UNKNOWN_SOAP_EXCEPTION_CODE_ZERO	Exceptions that occurs in EComPHP SOAP caller - postService() - where the postService has reached an exception, checked all possibilities of a proper error message but where the EComPHP for a reason can't catch a code. Instead, EComPHP gets an empty value or the value 0 as the errorcode. In this case, to ensure that the exception is thrown correctly, it generates this code instead of the "real" value.
1501	SSL_WRAPPER_MISSING	Thrown when EComPHP finds out that PHP is missing a SSL wrapper that is required for the services to work
6001	CALLBACK_TYPE_UNSupported	Occurs when a callback type is called that does not exist in the services or are unsupported
6003	CALLBACK_SALT_DIGEST_MISSING	Occurs when salt digest is missing in a callback registration
7000	BOOKPAYMENT_NO_BOOKDATA	Occurs when there is no payload to send into a rest- or SOAP service
7009	EXSHOP_PROHIBITED	Occurs when the user account prohibited exshop user credentials are used
7008	CREATEPAYMENT_NO_ID_SET	When preparing the payload for resurs checkout is used, and no payment id is properly set
7009	CREATEPAYMENT_TOO_FAST	When flooding prevention system is active and createPayment are running too fast in a shop, createPayment is throttled with this exception

Exception codes not in use or removed

CODE	CONSTANT	DESCRIPTION
1001	CLASS_REFLECTION_MISSING	REMOVED When reflection class could not be instantiated in older ECom-libs where WSDL stubs was included.
1002	WSDL_APILOAD_EXCEPTION	REMOVED When WSDL-files could not completely load into the core functions
1003	WSDL_PASSTHROUGH_EXCEPTION	REMOVED When EComPHP could not pass data through to the WSDL-stub library. Magic methods like __call, was probably involved here
1004	REGEX_COUNTRYCODE_MISSING REGEX_CUSTOMERTYPE_MISSING	DEPRECATED When country code or customer type is missing in getRegEx-request for form fields
1005	FORMFIELD_CANHIDE_EXCEPTION	DEPRECATED When a formfield is controlled canHideFormField() if it can be hidden or not and throwing exceptions are allowed and the field can not be hidden from the form
1500	SSL_PRODUCTION_CERTIFICATE_MISSING	REMOVED When EComPHP handled SSL errors internally
2000	NO_SERVICE_CLASSES_LOADED	REMOVED
2001	NO_SERVICE_API_HANDLED	REMOVED
6000		

	<code>CALLBACK_INSUFFICIENT_DATA</code>	REMOVED
6002	<code>CALLBACK_URL_MISMATCH</code>	REMOVED
2000	<code>NO_SERVICE_CLASSES_LOADED</code>	REMOVED
2001	<code>NO_SERVICE_API_HANDLED</code>	REMOVED
7001	<code>PAYMENTSPEC_EMPTY</code>	REMOVED
7002	<code>BOOKPAYMENT_NO_BOOK_PAYMENT_CLASS</code>	REMOVED
7003	<code>PAYMENT_METHODS_CACHE_DISABLED</code>	REMOVED
7004	<code>ANNUITY_FACTORS_CACHE_DISABLED</code>	REMOVED
7005	<code>ANNUITY_FACTORS_METHOD_UNAVAILABLE</code>	REMOVED
7006	<code>UPDATECART_NOCLASS_EXCEPTION</code>	REMOVED
7006	<code>UPDATECARD_DOUBLE_DATE_EXCEPTION</code>	REMOVED
7007	<code>PREPARECARD_NUMERIC_EXCEPTION</code>	REMOVED
7008	<code>BOOK_CUSTOMERTYPE_MISSING</code>	DEPRECATED Used in form fields generator, where the customertype NATURAL or LEGAL is missed to set

Error handling (Resurs error codes)

- Other errors
- SOAP related errors
 - Remarks

Other errors

We do have another collection of errors that might be good to take a look at that covers internal server problems and external errorhandlers that might warn you about the internal errors. See the list below.

[Error handling \(Resurs error codes\)](#)

[The "not a bug" list](#)

SOAP related errors

All errors/exception are returned as ECommerceError
Contains elements as defined in the following table.

Component	Type	Occurs	Nillable?	Description
errorTypeDescription	nonEmptyString	1..1	No	The textual description of the error type.
errorTypeld	int	1..1	No	Indicates which kind of error this is.
fixableByYou	boolean	1..1	No	"If this error has been avoided with some other input" = "It's a Resurs Bank problem"
userErrorMessage	nonEmptyString	1..1	No	An error message intended for the user of the web shop. This message must be shown!

Always use `userErrorMessage` to inform the customer about the error. It's translated to the shop's language. The fault string should be logged to help the shop owner to diagnose errors.

Do not under *any* circumstance show the `faultString` to the customer. It may contain sensitive information. But do log it!

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <soap:Fault>
      <faultcode>soap:Server</faultcode>
      <faultstring>Government identity invalid! : Malformed government id</faultstring>
      <detail>
        <ns2:ECommerceError xmlns:ns2="http://ecommerce.resurs.com/v4/msg/exception" xmlns:ns3="http://ecommerce.resurs.com/v4/msg/shopflow">
          <errorTypeDescription>ILLEGAL_ARGUMENT</errorTypeDescription>
          <errorTypeld>1</errorTypeld>
          <fixableByYou>true</fixableByYou>
          <userErrorMessage>Personnummer/Organisationsnummer 999999999 verkar vara inkorrekt</userErrorMessage>
        </ns2:ECommerceError>
      </detail>
    </soap:Fault>
  </soap:Body>
</soap:Envelope>
```

Remarks

All kind of errors will generate this kind of exception, but different groups of errors can be distinguished by their `errorTypeld`. For example, the `errorTypeld` 1 is "Invalid argument". You might want to take different actions depending on the `errorTypeld`, especially if the `fixableByYou` flag is true.

These are the current errorTypes, more might be added as time goes, but old error codes will remain unchanged.

Error Type Id	Error Type Description	Occurs	Notes

1	ILLEGAL_ARGUMENT		
3	INTERNAL_ERROR	Error server side, contact Resurs Bank.	
4	NOT_ALLOWED	You're not allowed to perform the requested action.	
8	REFERENCED_DATA_DONT_EXISTS	The data you're trying to use/get doesn't exist.	
9	NOT_ALLOWED_IN_ORDER_STATE		Lost from xsd.
10	CREDITAPPLICATION_FAILED		
11	NOT_IMPLEMENTED	The function is not implemented yet.	
14	INVALID_CREDITAPPLICATION_SUBMISSION	Something is wrong with credit/limit application	
15	SIGNING_REQUIRED	When signing is required and must be done to continue current operation.	
17	AUTHORIZATION_FAILED	Could not authorize debit on card/account.	
18	APPLICATION_VALIDATION_ERROR		
19	OBJECT_WITH_ID_ALREADY_EXIST		
20	NOT_ALLOWED_IN_PAYMENT_STATE		NOT_ALLOWED_IN_ORDER_STATE
21	CUSTOMER_CONFIGURATION_EXCEPTION		
22	SERVICE_CONFIGURATION_EXCEPTION		
23	INVALID_CREDITING		
24	LIMIT_PER_TIME_EXCEEDED		
25	NOT_ALLOWED_IN_CURRENT_STATE		
26	INVALID_FINALIZATION		
27	FORM_PARSING		
28	NOT_ALLOWED_INVOICE_ID		
29	ALREADY_EXISTS_INVOICE_ID		
30	INVALID_IDENTIFICATION		
31	TO_MANY_TOKENS		Yes, the constant has a typo.
32	CUSTOMER_ALREADY_HAVE_VALID_CARD		
33	CUSTOMER_HAS_NO_VALID_CARD		
34	CUSTOMER_HAS_MORE_THAN_ONE_VALID_CARD		
35	INVALID_AUTHENTICATION		
36	ANNUL_FAILED		
37	CUSTOMER_HAS_NO_VALID_ACCOUNT		
99	LEGACY_EXCEPTION		
502	WEAK_PASSWORD	When changing password and the new one is to weak to be accepted	
503	NOTAUTHORIZED	When trying to do something that requires <i>authenticated</i> (yeah, the code is wrong, bummer)	

The fixableByYou flag means that the system works as intended, and that some other input could have prevented the error from happening. For example, a customer not having enough funds on his card account renders an ECommerceException with the fixableByYou set to `true`. If, on the other hand, we have problems communicating with the database, the fixableByYou flag should be `false`. When this flag is `false` there is nothing you can do except showing the customer the error userErrorMessage.

The exception contains two error messages. One of them, the exception message, the faultString, is a technical description of what went wrong, and it's not suitable to show to the end user. Just log it, and use it for diagnosing and development. The userErrorMessage on the other hand, we more or less require you to show to the customer if something went wrong. The reason for requiring you to show it is that some of the messages we return have a legal meaning not to be fiddle with.

General: Integration development

- Shared plugins - Contribution
- New shared plugin - step by step building
- What to keep in mind
- Where to start
 - Configuration - Admin
 - Sections to create in the admin control panel
 - The payment methods section
 - The checkout and forms that needs to be filled in
 - Callbacks

Shared plugins - Contribution

The following text covers one way to create a shared plugin (meaning, a plugin that everybody can use), from scratch. It may also change over time as we find new ways to optimize our plugin-making. Our own goal with making new e-commerce-plugins is to make them streamlined. This means, we are trying to make all plugins act as much in the same way, in all platforms, as possible.

If you want to contribute, you may want to take a look at [Development](#) - this page summarizes our plugins and where they are located.

New shared plugin - step by step building

For example, when generating form fields for a checkout, we are always trying to utilize the e-commerce platform's own form fields as much as we can, so when a customer fills in address data, the parts of the payment booking which is sent to Resurs e-commerce are inherited from the primary store. This means that most of our form fields, except for government ID and a few more fields can be kept hidden. The lesser information customers has to fill in, the better.

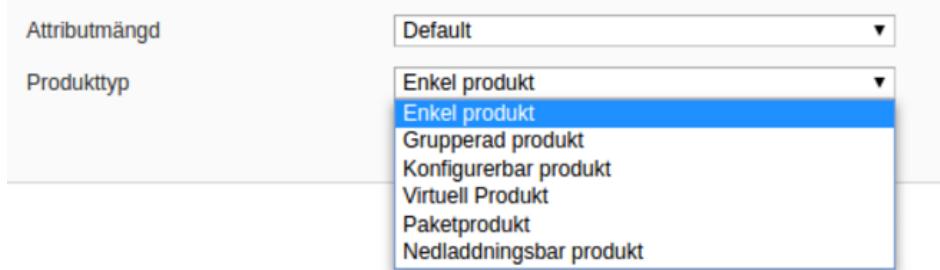


Testing in production?

Never test anything in production mode unless you are really sure what you are doing. We do not offer any test data for our production environment, due to legal regulations.

What to keep in mind

- If you're building your own checkout from scratch, where all payments are controlled by your checkout, make sure you'll handle both shipping, fees, discounts and article types that the shop provides.



Where to start

Configuration - Admin

If you are already familiar with the platform where you are going to create your plugin, our suggestion is to first prepare the administration interface to handle, for example, the payment methods that should be available for your customers. Our payment methods are dynamic, which sometimes may differ how other platforms actually are designed. In our "from scratch"-example, we start with no payment methods at all, and aiming at responsive configuration at this point, where we start with a very simple interface which initially always points at our test environment. In the screen dump you see the initial configuration from Prestashop.

 **Settings**

Module version: 1.0.0

Resurs Bank Webservices and API

Current environment: **Test** ▾

Production: <https://ecommerce.resurs.com/ws/V4/>

Test: <https://test.resurs.com/ecommerce-test/ws/V4/>

Resurs Bank login credentials

Customer ID: **phpapi**

Customer Password: **.....**

Resurs Bank API

Callback URL: <http://mock.prestashop.cte.loc/modules/resursbankpayment/resurscallback.php>

Callback events registered: *None. Update settings to update callback events*

[Update settings](#)

As soon as the credentials are filled in, the example plugin are starting to look for the current payment methods set up and the goal with the plugin is to make it as optionless as possible.

Sections to create in the admin control panel



Generic plugins and country settings

As our merchant credentials only supports one country per credential account, you need to set up configurations for the country the purchasing should be allowed in. This is especially important for merchants that allow shopping from multiple countries, where you have to define for which country our payment methods are allowed. Currently, our API's do not have any interface that tells which country the credentials belong to, so the country control must be done from the plugin. For example, a Norwegian customer can not show our payment methods in a Danish store. Therefore, this limitation must be set.

- URLs for where production and test environment are pointing
See below, for URLs to the shop flow
- Configurable URL, to a local place, for where callbacks are handled (Resurs Bank can send events to your shop, when payments are changed and this is where they are going to be handled)
- A toggler which by default always points at the test environment (Recommended)
- Login credentials for web services
- Section for payment methods
- Eventually a section for where each order status is set, depending on the webshop (An advice in this case is to have a status called "payment signing". At least in Magento this status is used to detect if a signing has been canceled by the customer or not)

Example:

When using a payment from Resurs Bank, an order should initially be set to "Holded", since we are waiting for the fraud control to be done. When the fraud control is done, Resurs Bank is sending a callback signal to the store to accept the payment as accepted, which will set the

order to "Processing". If the store supports this way to handle orders, you should let the store's own API's do the job for you as much as possible (like in the screenshot attached, from the Magento control panel).

New order status	New
Status for orders that will not be accepted	Canceled
Freeze status	On Hold
Unfreeze/thaw to status	Complete
Annulment order status	Canceled
Pending payment status	Processing
Signing status (Note: A status has to be created for this to work, in Magento's internal handler for statuses)	Payment signing

Web services are based on SOAP. It's recommended from this point to never add wsdl-parameters or URLs directly to each service provided by Resurs Bank. Since there are different kind of services (SimplifiedShopFlowService, AfterShopFlowService, etc), your plugin should instead generate the rest of the URLs by itself so everything can be called dynamically. In our PHP-based gateway each service are pre-loaded like below, so each of them can be called if they exists.

\$this->environment contains the above URL, depending on which is chosen in the admin panel. By doing this, all calls to webservices will be much easier to handle.

Services loader class (Example)

```
if (class_exists('Resurs_SimplifiedShopFlowService')) {$this->simplifiedShopFlowService = new Resurs_SimplifiedShopFlowService($this->soapOptions, $this->environment . "SimplifiedShopFlowService?wsdl");}
if (class_exists('Resurs_ConfigurationService')) {$this->configurationService = new Resurs_ConfigurationService($this->soapOptions, $this->environment . "ConfigurationService?wsdl");}
if (class_exists('Resurs_AfterShopFlowService')) {$this->afterShopFlowService = new Resurs_AfterShopFlowService($this->soapOptions, $this->environment . "AfterShopFlowService?wsdl");}
if (class_exists('Resurs_ShopFlowService')) {$this->shopFlowService = new Resurs_ShopFlowService ($this->soapOptions, $this->environment . "ShopFlowService?wsdl");}
```

The payment methods section

Payment methods are dynamically transferred from Resurs Bank e-commerce, so you need a small form where each payment method can be configured. Later on, this information will be used to display the correct methods in the checkout. This may differ from the "normal" behaviour in an e-commerce platform where payment service providers normally offers one (or a few) static payment methods and therefore only generates a simple form with configuration settings for the methods. The image below shows you the settings from Magento, and that's the primary configuration you probably need, building your settings base a plugin - and the best practice here is, as shown, to store data about them locally (cached), so no queries has to be made to e-commerce for each checkout-request.



Important notice about payment methods

You should also be aware that payment methods *can* disappear, depending on how they are used. For example, there are a minimum and a maximum payment amount bound to a method, so when a customer exceeds those amounts, they should normally not be shown in a checkout. If a method exceeds the amount, or expires/gets disabled (i.e. a method may be set up for a limited usage time, etc), and they are still shown by mistake in a checkout, this generates an error from Resurs.

Faktura privat

Betalningsstatusen är aktiv?	<input type="text" value="Yes"/>	[STORE VIEW]
Title	<input type="text" value="Faktura privatperson"/>	[STORE VIEW]
Sorteringsordning för betalmetod	<input type="text"/> ▲ Sorteringsprioritet: Lägre värde innebär högre placering.	[STORE VIEW]
Avgift för att använda betalningsalternativet	<input type="text"/>	[STORE VIEW]
Paymentmethod Logotype	<input type="text" value="/frontend/base/default/resursbank/images/resursbank.png"/> <i>Current logotype</i> 	[STORE VIEW]
Beskrivning för betalningsalternativets avgift (kommer visas på orderraden)	<input type="text"/>	[STORE VIEW]
Beskrivning för betalmetoden (kommer visas i kassan när betalmetoden väljs)	<input type="text"/>	[STORE VIEW]
Beloppsgränser för betalmetod	10 - 50000	[STORE VIEW]

The checkout and forms that needs to be filled in

Important notices

- When payment methods are shown to customers in a webshop, it is important that the "SEKKI" (Standardiserad Europeisk Konsumentkreditinformation) **always** is shown (the only exception is for an existing card, i.e. not a new one, where the customer most likely already has accepted an agreement).
- You should also consider making a "landing page" for your payment methods, when they require signing by the customer, where failed and successful signings should be handled.

- When creating payment forms, used for booking payments at e-commerce, make sure that you are also adding content validation to the fields ([see here](#)).

When configuration has been prepared in the administration panel, it's time to create a checkout interface for customers. Resurs Bank is building all plugins streamlined so they look and work, compared to each other, as similar as possible (the same rules are applied to the administration interface). Form fields that are offered by the webshop should inherit to the plugin's form fields as much as possible. Normally the fields we're thinking of are e-mail and phone numbers, but also (as feature) contact person when the payment method type is bound to a company. Payment methods should be shown or hidden depending on their min and max payment amounts (if a payment method is shown when it should not be, an error will occur).

Different payment methods require different kind of form field elements. They are listed below. Application full name is only used in the really old flow and can be removed.

Please take note on the optional phone number fields below. It is strongly recommended to always force a phone number at some point when calling the shopflow-API's even though they are not entirely required. If the numbers are left out of the payload, Resurs will redirect customers to another page to fill in data that the customer has forgotten to fill in. As Resurs in the end needs the phone number it is better that you handle this in your codebase.

Payment Method SpecificType	Applicant Government ID	Applicant Telephone Number	Applicant Mobile Number	Applicant Email Address	Contact Government ID	Card number	Applic Name
INVOICE <i>Natural</i>	REQUIRED	OPTIONAL (Read the red note)	OPTIONAL (Read the red note)	REQUIRED			OPTIONAL
INVOICE <i>Legal</i>	REQUIRED	OPTIONAL (Read the red note)	OPTIONAL (Read the red note)	REQUIRED	REQUIRED		REQUIRED /OPTIONAL
CARD <i>Existing card</i>	REQUIRED					REQUIRED	OPTIONAL
REVOLVING_CREDIT <i>New card</i>	REQUIRED		REQUIRED	REQUIRED			OPTIONAL
PART_PAYMENT <i>Part payment</i>	REQUIRED	REQUIRED	REQUIRED	REQUIRED			OPTIONAL

When creating forms for payment, make sure that you are also adding content validation (regex) to the fields ([see here](#)).

Address completion for filling in address data on an order page is available through [Get address](#). Currently, the service is only available for Sweden. Code examples for PHP (Magento) can be found at the [OldFlow Plugin - Coding examples for getAddress](#)-section.

Callbacks



Callback notices

Callbacks must be reachable externally. If you are testing your web store in a protected environment (which normally is something else than production) and you want to test callback functionality, you have to make sure that Resurs can reach the callback URLs.

An important thing, when building a plugin for Resurs Bank, is to make sure you are creating support for callbacks, so that Resurs' e-commerce services will be able to process the order after for example fraud controls on Resurs' side, when orders normally should get thawed.

Building plugins (settings)

- Configuration - Admin

Configuration - Admin

Module version: 1.0.0 Update settings

Resurs Bank
Resurs Bank Webservice and API

1 Module active

2 Environment

3 Production

3 Test

4 Resurs Bank login credentials
Country

5 Web service username

6 Web service password

7 Callback URLs registered at Resurs Bank

Type	URL
BOOKED	<code>https://ws2.yoursite.com.loc/?wc_api=WC_Resurs_Bank&event-type=BOOKED&paymentId=&digest=(digest)</code>
ANNULMENT	<code>https://ws2.yoursite.com.loc/?wc_api=WC_Resurs_Bank&event-type=ANNULMENT&paymentId=&digest=(digest)</code>
UNFREEZE	<code>https://ws2.yoursite.com.loc/?wc_api=WC_Resurs_Bank&event-type=UNFREEZE&paymentId=&digest=(digest)</code>
TEST	<code>https://ws2.yoursite.com.loc/?wc_api=WC_Resurs_Bank&event-type=TEST&prm1=(param1)&prm2=(param2)&prm3=(param3)&prm4=(param4)&prm5=(param5)</code>

Callback communication test performed: yyyy-mm-dd (hh:mm:ss) Environment: 'Test / Production'

1. Option to enable and disable the module.
2. Option to enable switch between test and production environment at Resurs Bank.
3. End points for **production** and **test** environment.
4. Country of merchant credentials.*
5. Username provided by Resurs Bank.
6. Password provided by Resurs Bank.
7. Callback implementation.**
You can use either **SOAP** or **REST** to register callback.
[Read more](#) how to use and implement callbacks.



* Country setting

Resurs Bank merchant credentials only supports one country per credential account. Merchants that allow shopping from multiple countries need to have one credential for each country (Sweden, Norway, Finland or Denmark). I.e., a Norwegian customer can not shop with payment methods in a Swedish store.



** Callback notices

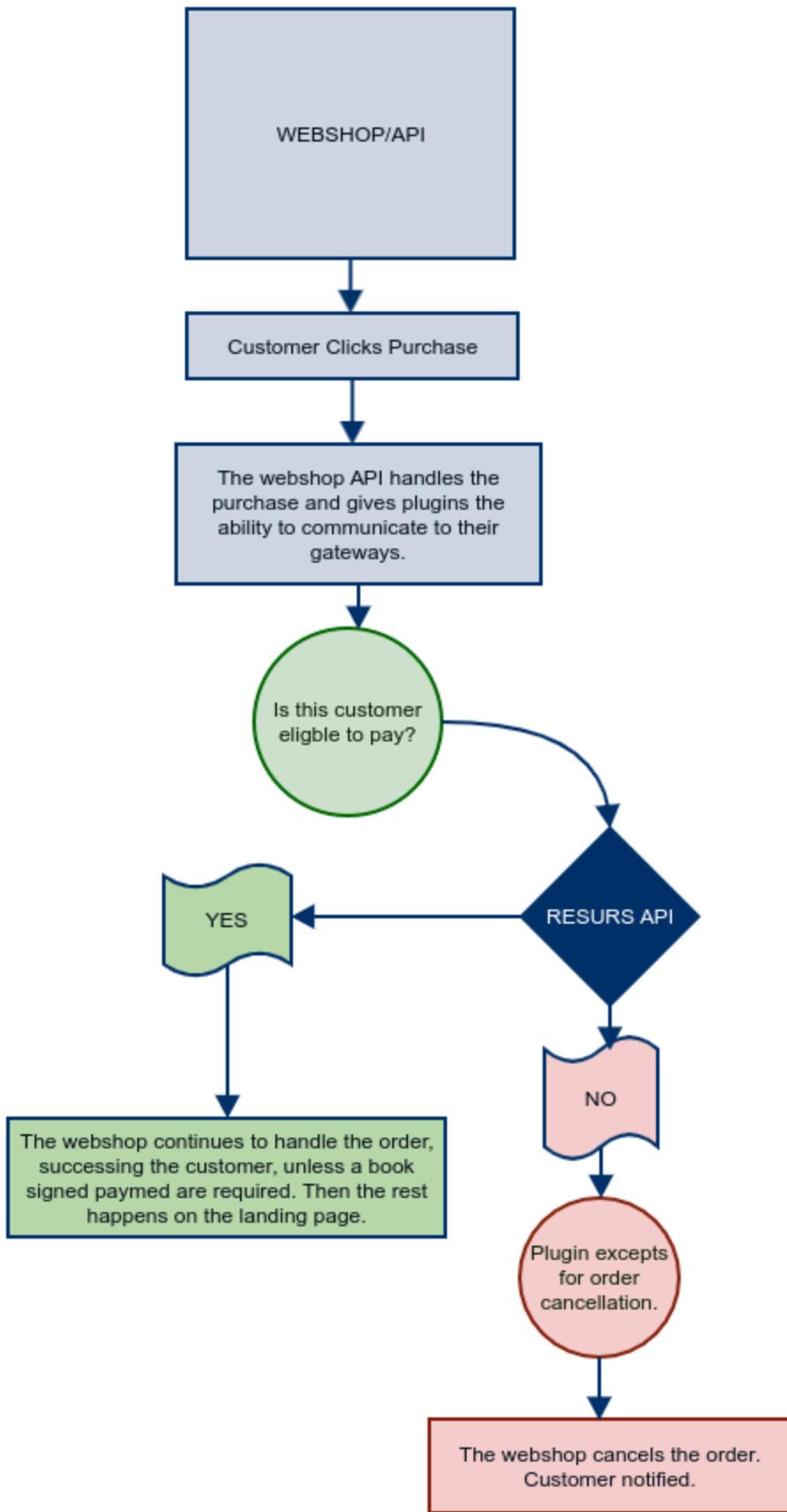
Callbacks must be reachable externally. If you are testing your web store in a protected environment and you want to test callback functionality, you have to make sure that Resurs can reach the callback URLs.

Handling checkout-flows bottle necks

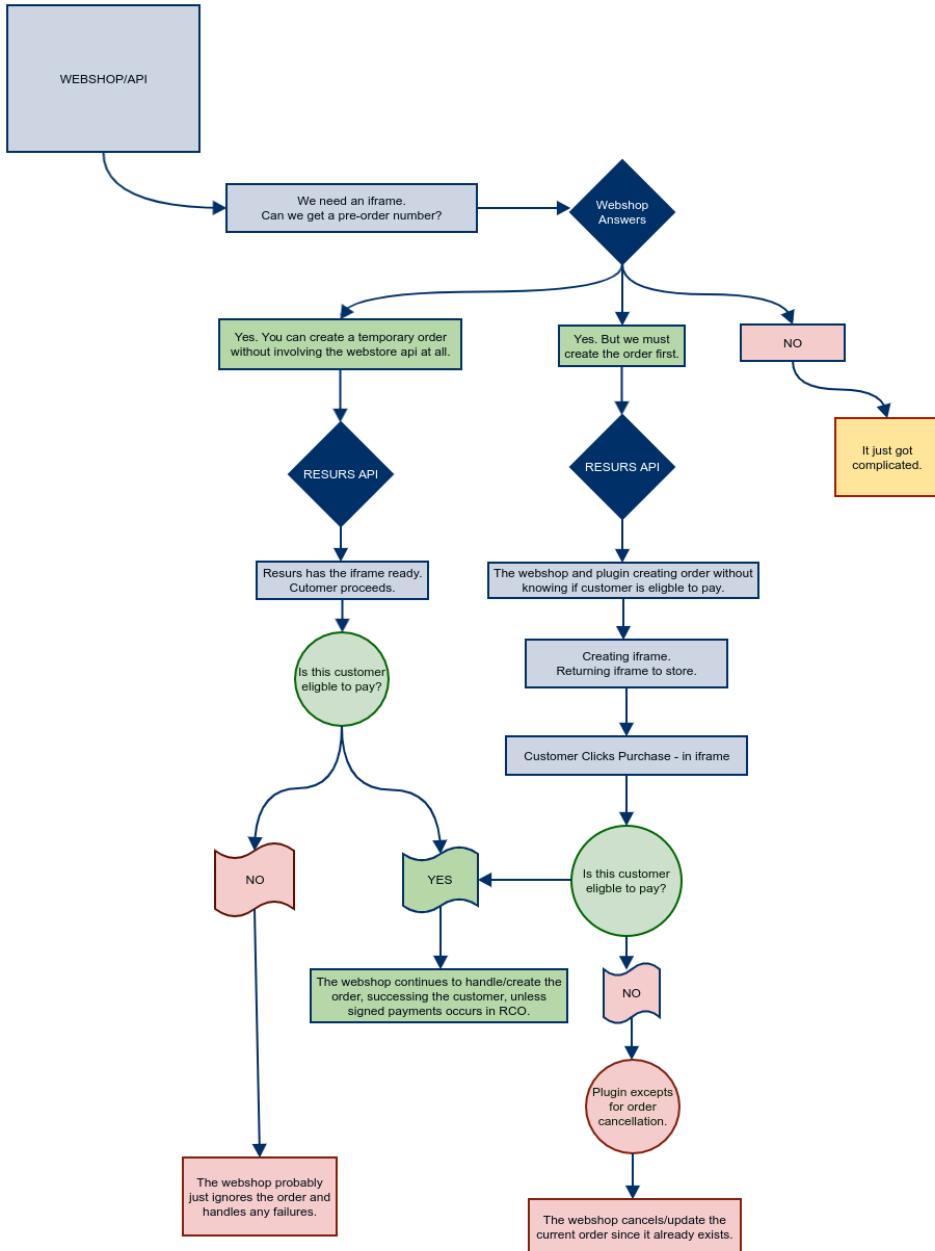
As described at the [Resurs Checkout page](#), there might be some bottlenecks ([moment 22](#)) to solve in some flows. This page is trying to visualize the respective flow.

Many stores are designed to take payment before creating an order. Bankwire and Check modules are designed to create an order, before the customer is able to pay, and that should be fairly obvious why, but real time payment methods (like paypal or credit card), an order is created AFTER the payment is completed ([Full quote source from Prestashop](#)).

This said, Resurs Bank Simplified flow is by best practice the flow that handles payment in a proper way:



Meanwhile RCO runs a more uncommon way



PHP and development libraries

- Supported code and links
- Discontinued releases

Supported code and links

Name/Docs	Link	Version	Status	Support	Dependencies	Shop API's
ECom v2.x	git	2.0	LTS	2023 -	PHP = 8.1	MAPI
ECom v1.x	git	1.3	LTS	2017 -	PHP >= 7.3 SOAP/XML curl+curl	SIMPLIFIED RESURS CHECKOUT HOSTED

Discontinued releases

Name	Version	Status	Notes
ECom1	1.1	DISCONTINUED	No longer supported. Last confirmed stable 1.1.50
ECom1	1.2	REMOVED	Bridge between 1.1 and 1.3. <i>Codebase removed.</i>
ECom1	1.0	DISCONTINUED	No longer supported. Last confirmed stable 1.1.50

Version 1.x (EComPHP)

This document covers information about EComPHP 1.x (Currently 1.3.x).



Resurs Bank in the community

If something does not work properly in EComPHP, feel free to [join the community via bitbucket](#), or contact us at onboarding@resurs.se. EComPHP is updated continuously to improve functionality.

To speed up the wsdl requests make sure you use `setWsdlCache(true)` on runs.

Description

Resurs EComPHP Gateway is a simplifier for our webservices, with functionality enough to getting started fast. It communicates with the [Simplified Flow API](#) for booking payments, [Configuration Service](#) and the [After Shop Service API](#) for finalizing, crediting and annulments etc. This full version of the gateway communicates with [Hosted Payment Flow](#) and [Resurs Checkout Web](#) (supporting both REST and SOAP). A PHP-reference for EComPHP is located at <https://test.resurs.com/autodocs>, if you want to take a look at our automatically generated documentation.

As EComPHP is continuously developed, you should take a look at our bitbucket repo to keep this information updated. It can be found at <https://bitbucket.org/resursbankplugins/resurs-ecomphp>.

Branches

Red branches is obsolete and no longer officially maintained.

- 1.3
- 1.1
- 1.0

Make sure to always upgrade!

EComPHP is continuously delivering upgrades that makes things work better. The current 1.3 is not fully PSR4-compliant, but our goal is to get rid of non-PSR4 code (which takes time). Below follows a list of some major non-breaking changes that takes us closer to such compliance.

- v1.3.47: Internal curlwrapper calls are now using proper class calls instead of deprecated classes from 6.0 - this also means that we no longer has to verify that they do exist before loading them. They are called on fly.

Preparing the library

Download

Take a look at the ongoing project at <https://bitbucket.org/resursbankplugins/resurs-ecomphp>

Requirements and dependencies

- Use composer
- Do not use other versions than EComPHP 1.3 as the prior versions is no longer guaranteed to work.
- PHP 7.3 - 8.0 are supported, the dialect is written for 5.6
- Even if you plan to only use Resurs Checkout, features are widely spread between many flows, which we also recommend to install the library so it can fully use all flows.
- [OpenSSL](#): For reaching Resurs Bank REST/SOAP
- [curl](#): php-curl with SSL support
- php-xml and streams (For the SOAP parts)
- EComPHP uses [NetCURL](#) for calls via both SOAP and REST. The packagist component is located [here](#)

Installing curl, xml, soapclient, etc

For Ubuntu, you can quickly fetch those with apt-get like below, if your system is missing them:

```
apt-get install php-curl php-xml php-soap
```

Installation

As mentioned above, we recommend that you use composer to install. How you install with composer is a bit up to you. But with all dependencies available, installing may look like this, depending on how you'd like to follow the release branches:

```
# Either this:  
composer require resursbank/ecomphp:1.3.*  
# Or this:  
composer require resursbank/ecomphp:^1.3
```

In its simplest form, composer.json could look like this:

composer.json

```
{  
    "require": {  
        "resursbank/ecomphp": "^1.3"  
    }  
}
```

Deploying plugins with EComPHP bulked

If you need to release a plugin with EComPHP, you can use composer to prefer-dist like the example below. However, the prefer-dist may not be entirely safe and components could be missing out. To be sure that everything stays put, the below example also shows how to clean up repositories that still contain a git reference.

Deploy and redeploy

```
#!/bin/bash  
composer clearcache  
rm -rf vendor/composer.lock  
composer install --prefer-dist  
find vendor/ -type d -name .git -exec rm -rf {} \; >/dev/null 2>&1  
find vendor/ -name .gitignore -exec rm {} \; >/dev/null 2>&1
```

Start building - minor examples

At its simplest level, this is the way how to start talking with the API. If you need detailed documents, take a look at the subsections in this space.

Simple start

```
use \Resursbank\RBEcomPHP\ResursBank;  
// [...] Other usages, like RESURS_ENVIRONMENTS, ETC [...]  
require_once("vendor/autoload.php");  
  
$flow = new ResursBank('storeUsername', 'storePassword', ResursEnvironments::ENVIRONMENT_TEST);  
// Since v1.3.40 caching wsdl has been made easier.  
$flow->setWsdlCache(true);  
$flow->getPaymentMethods();  
// [...] more code [...]  
$flow->getPayment('orderIdOfThePaymentYouJustCreated');
```

Registering callbacks

setRegisterCallback is set, by default, to register callbacks through REST instead of SOAP (to avoid using SoapClient).

Make sure that the callback digest key are saved in your endpoint also, so that it can be tested on incoming calls. Also make sure you are really using a secure URL configuration (i.e. you need to use the digest to make sure your orders can not be manipulated by someone).

getPaymentMethods

```
require_once(__DIR__ . "/classes/rbapiloader.php");  
$flow = new \Resursbank\RBEcomPHP\ResursBank('storeUsername', 'storePassword');
```

```
$flow->setWsdlCache(true);
$this->flow->setCallbackDigestSalt('St0res4lTkeY');
$paymentMethodObject = $this->flow->setCallback(ResursCallbackTypes::UNFREEZE, "http://shop.test.com/callbacks/?paymentId={paymentId}&digest={digest}");

```

EComPHP: ChangeLog

Upcoming issues

Key	Summary	T	Updated	P	Status
ECOMPHP-440	When unregistering/registering multiple callbacks, prevent further registration of callbacks on first timeout error	<input checked="" type="checkbox"/>	2023-06-14		TO DO
ECOMPHP-428	Adjust the way payment methods are handled		2022-04-22		TEST
ECOMPHP-429	Expired tokens vs secrets		2022-04-22		TEST
ECOMPHP-427	Stores and methods		2022-04-22		TEST
ECOMPHP-423	Getting callbacks by rest are not throwing unauth-errors		2022-04-22		TO DO
ECOMPHP-254	Improve design in Ecom		2018-05-18		TO DO

6 issues

Finished issues

Key	Summary	T	Updated	P	Status
ECOMPHP-460	isTrustly (Detect trustly cards)	<input checked="" type="checkbox"/>	2023-06-14		DONE
ECOMPHP-462	Find payment method variants in translation table (json)	<input checked="" type="checkbox"/>	2023-06-14		DONE
ECOMPHP-464	Feeding null into setCountryByCountryCode may cause garbage output which breaks checkout		2023-06-14		DONE
ECOMPHP-465	Minimize exception exposures	<input checked="" type="checkbox"/>	2023-06-14		DONE
ECOMPHP-454	Erormessages are too long due to prior ways of handling them	<input checked="" type="checkbox"/>	2023-04-06		DONE
ECOMPHP-453	Secure layer for bookSignedPayment	<input checked="" type="checkbox"/>	2022-09-26		DONE
ECOMPHP-97	Test suite for v1.1 is broken		2022-04-22		DONE
ECOMPHP-270	Bamboo: apiPaymentMethodsWithWrongCredentials		2022-04-22		DONE
ECOMPHP-443	Minor fixes for CheckoutType where is forgotten in rbapiloader		2022-04-22		DONE
ECOMPHP-417	Annuity factor calculations are handling duration and factors wrong		2022-04-22		DONE
ECOMPHP-271	Bamboo: generateSimpleSimplifiedInvoiceOrder		2022-04-22		DONE
ECOMPHP-441	Allow set up proxy on fly (before instantiation)	<input checked="" type="checkbox"/>	2022-04-22		DONE
ECOMPHP-4	Update EComPHP UnitTests and wsdl's	<input checked="" type="checkbox"/>	2022-04-22		DONE
ECOMPHP-384	cancelPayment and order filtering		2022-04-22		DONE
ECOMPHP-294	Patch getOrderStatusByPayment to handle FINALIZATION on "direct debits"		2022-04-22		DONE
ECOMPHP-314	updatePaymentReference gives misleading error codes when payments already exists	<input checked="" type="checkbox"/>	2022-04-22		DONE
ECOMPHP-295	Patch paymentFinalize to handle FINALIZATION on "direct debits"		2022-04-22		DONE

ECOMPHP-434	canDebit should include "frozen"-check	<input checked="" type="checkbox"/>	2022-04-22		<button>DONE</button>
ECOMPHP-366	Undefined \$renderCallback['eventType'] will throw "unknown eventType"		2022-04-22		<button>DONE</button>
ECOMPHP-451	Allow identical order/product rows to be split up on bookPayment	<input checked="" type="checkbox"/>	2022-04-22		<button>DONE</button>

Showing 20 out of 389 issues

Refactoring Notes for AfterShop in EComPHP

There is currently a plan of refactor EComPHP regarding to how it handles the AfterShop flow today. This page is a bunch of notes, that documents the old behaviour and the upcoming solution for a new flow.

- [Developer notes](#)
- [Things that happened during the finalization flow in EComPHP < 1.1.22:](#)
- [Things that happens during the renderPaymentSpecContainer\(\) \[deprecation\] in a finalization](#)
- [Methods for new release](#)
 - [sanitizeAfterShopSpec\(\)](#)
 - [FINALIZE](#)
 - [CREDIT](#)
 - [ANNUL](#)
 - [UPDATE \(OBSOLETE\)](#)

Developer notes

- None of the renderers are supporting the fact that not only artNo may differ. If there's more identical artNo with different description/price, they might be handled equally - so there might occur corner cases that are not handled by this engine.
This is normally not an issue as long as the renderers does not take care of very specific attributes.
- The behaviour of the getPaymentSpecByStatus should reflect what has once been created, so identical articles should not be a problem since they are keyed on artNo, description, price at Resurs Bank
- The magento plugin behaviour uses the methods that is about to get deprecated, so the old methods will probably be untouched until we reach at least v1.2

```
if (!connection->creditPayment($reference, $this->getCreditMemoItems($subject), array(), false, true))  
{}
```

- The easiest way to do future afterShop is to use the same method as the additionalDebitOfPayment is using, by simply adding the product rows into the internal payload.
This method should however support the prior payload system that sets a manual payload. By doing this, we could also add custom objects into the aftershop method.

Things that happened during the finalization flow in EComPHP < 1.1.22:

1. Adds metadata to the current order, that gets reflected to the invoice (CustomerID)
2. The inbound orderLine-array validates to match an array when only one article is sent inbound (and in the wrong way)
3. The payment id requested to finalize is fetched from Resurs
4. The payment specification from the client gets rendered, based on the fetched payment [[Why?](#)]
PARAMS(\$paymentId, ResursAfterShopRenderTypes::FINALIZE, \$paymentArray, \$clientPaymentSpec, \$finalizeParams, \$quantityMatch, \$useSpecifiedQuantity)
renderPaymentSpecContainer() - What does this do? See below in the
5. The new rendered payment container are posted to the finalizePayment-service during a try-catch moment

Things that happens during the renderPaymentSpecContainer() [deprecation] in a finalization

Parameters received

```
( $paymentId, ResursAfterShopRenderTypes::FINALIZE, $paymentArray, $clientPaymentSpec, $finalizeParams, $quantityMatch,  
$useSpecifiedQuantity );
```

- The payment id
- The type that should be rendered
- The customer client payment request
- *Finalizing parameters*
- *QuantityMatchSetup*
- *SpecifiedQuantitySetup*

Old behaviour

The renderer should not only render a correct payment specification.

1. Function is initialized with a specline render (renderSpecLine) for the original payment spec.
The specline renderer do the exact same as getPaymentSpecByStatus, except for that it also sanitizes orderlines based on their statuses.
Example: On finalizations, order rows that is not annulled or credited should be included in the payment spec. On annullments, rows that is not debited nor credited should be included.
2. Once finished with the primary renderer, the payment spec requested should be validated so that each row really exists in the specification.
A lot of logic are now calculated depending on prices, amounts and quantity that is probably not needed in this form.

3. A new totalAmount+totalVatAmount are now recalculated and stored in the new "compiled" payment spec
4. If the payment method is based on invoicing, further data are now included in the final payment specification:
createdBy, orderDate, invoiceDate, invoiceId (if not exists, create one by getNextInvoiceNumber())
Note: ourReference, yourReference, preferredTransactionId, orderId was never included in the first version.

Methods for new release

sanitizeAfterShopSpec()

This sanitizing function should, at least for unmanaged payment specs (full handling), make sure that each orderLine-bulk are clean. Sanitizing works much like `getPaymentSpecByStatus()`. However, this function should match a payment and filter out a specific set of orderLines. For example - for finalization, there's only one valid list of orderLines, and therefore each AUTHORIZE should be matched against what's in the other status blocks (DEBIT, ANNUL, CREDIT) - and if they do exist there in those blocks, they should not be added into the final returning AUTHORIZE block - so that only AUTHORIZE-only-rows remain.

FINALIZE

Read from block

AUTHORIZE

Sanitize

DEBIT, ANNUL, CREDIT (Everything in the AUTHORIZE list, that is not yet DEBIT, ANNUL, CREDIT)

CREDIT

Read from block

DEBIT

Sanitize

ANNUL, CREDIT (Everything in the DEBIT list, that is not yet ANNUL, CREDIT)

ANNUL

Read from block

AUTHORIZE

Sanitize

DEBIT, ANNUL, CREDIT (Everything that is not yet DEBIT, ANNUL, CREDIT)

UPDATE (OBSOLETE)

Returns AUTHORIZE (is still even active?)

EComPHP: Contribute

EComPHP is delived as open source, we encourage the community to develop and enhance the module. On this page, some tips is being held as there are "best practices" to use with it.

- Developing functions using Resurs Bank service
 - Initializing services
 - Building functions

Developing functions using Resurs Bank service

Initializing services

Most of the services provided by Resurs Bank are already implemented. If there are some missing, enhancing the module with those is welcome. For all function being built with EComPHP we usually need to initialize user authentication and such thigs, which is being made with **InitializeServices()**.

Building functions

If the services are provided from Resurs webservices (WSDL), you should take a look at \$ServiceRequestList, for where all function calls are linked to the service that is providing it. For example, getPaymentMethods could be reached from the SimplifiedShopFlowService, so in the array, it's defined as 'getPaymentMethods' => 'SimplifiedShopFlowService'. All methods are defined the same way. \$ServiceRequestList are then using the function postService() to handle the call. So, for this specific method, we can *almost* keep ourself with a oneliner:

getPaymentMethods example

```
public function getPaymentMethods($parameters = array()) {  
    $this->InitializeServices();  
    return $this->postService( "getPaymentMethods", array(  
        'customerType'    => isset( $parameters['customerType'] ) ? $parameters['customerType'] : null,  
        'language'        => isset( $parameters['language'] ) ? $parameters['language'] : null,  
        'purchaseAmount'  => isset( $parameters['purchaseAmount'] ) ? $parameters['purchaseAmount'] : null  
    ) );  
}
```

For Resurs Checkout and hosted flow, we will never get to use \$ServiceRequestList as the urls are not related to WSDL. Instead, at least for Resurs Checkout, we use getCheckoutUrl() to get the current URL for the checkout. As both production and test environments are available, getCheckoutUrl() handles this for you. To get a proper URL to for example the callbacks interface, you can do a very simple request that looks like this:

Get checkout URL

```
$this->getCheckoutUrl() . "/callbacks";
```

Since the checkout allows both POST, GET, PUT and DELETE, the calls are being handled "more manually" for each function, being implemented. In the example below we are preparing to register a callback for annulling payments.

Resurs Checkout calls

```
$renderCallback = array('array', 'that', 'contains', 'a', 'setup', 'for', 'registering', 'an', 'annulment',  
'callback');  
$renderedResponse = $this->CURL->doPost($this->getCheckoutUrl() . "/callbacks/ANNULMENT", $renderCallback,  
\TorneLIB\CURL_POST_AS::POST_AS_JSON);
```

As you can see here, we are speaking directly to the CURL interface, which handles everything. Since the checkout like JSON data, and registering callbacks is a POST method, we'll do a doPost() with the array, that is - when posting the data - is converted to a JSON string. If the rest call not requires a POST, a GET can be done in a similar way:

Fetching all registered callbacks for a merchant

```
$CallbackList = $this->CURL->getParsedResponse($this->CURL->doGet($this->getCheckoutUrl() . "/callbacks"));
```

EComPHP: createPayment [hostedFlow]

- Initialize the module
- Successful orders vs unsuccessful orders - Setting rules of signing (Flow requirement)
 - The syntax for setSigning is as follows:
- Set the preferred payment service (flow)
- Prepare the customer
 - Billing address
 - getPayload()
 - Additional required customer data
 - getPayload()
- Preparing the cart
 - Function: addOrderLine()
 - Function: getPayload()
- Special requirements
- Preparing a order reference
- Creating the order
 - Successful requests

The hosted flow is a clone of the simplified shop flow, with the difference that Resurs Bank is hosting the payment solution rather than it is integrated in the store; A paypal-like solution. To create a standard order, as shown in our test suite, the steps below are required to build a payment in the simplified flow. The examples are taken from the test suite, so we assume that you do run this in a similar environment.

Initialize the module

```
use \Resursbank\RBEcomPHP\ResursEnvironments;
$preferredEnvironment = ResursEnvironments::ENVIRONMENT_TEST;
$this->rb = new \ResursBank( $this->username, $this->password, $preferredEnvironment );
// or
$this->rb = new \ResursBank( $this->username, $this->password );
$this->rb->setEnvironment($preferredEnvironment);
```

Successful orders vs unsuccessful orders - Setting rules of signing (**Flow requirement**)

This is a part of customer preparing and required by the flow to properly work when it comes to signing/authorizing payments that requires this.

Before proceeding, we need to configure the payment to handle signings; if we for some reason require that your customer signs the payment - this happens for example, if you set a delivery address that mismatches with your billing address. If you have your own requirements (for example, you want to force signing every time a payment are created) this part is also important. Besides, without the rules of signing, Resurs Bank won't accept the payment. But the setup is quite simple. You need to set up two urls for this: One for successful signings and one for signings that fails during the payment process.

! Required data

Hosted flow, do require this setup, the same way as the simplified flow do as this part is also configuring the success, back and failurl for your store. In other words; this is where your landing page for successful payments is set up.

setSigning()

```
$this->rb->setSigning("https://mystore.test.com/signingSuccessful", "https://mystore.test.com/signingFailed",
false);
```

Running this will prepare for eventually letting the customer sign the payment. The boolean false tells Resurs Bank to not force any signing.

The syntax for setSigning is as follows:

Parameter	Type	Description
successUrl	string	Tells Resurs Bank where your payment, when successful, will land (<i>Thanks for your order</i> , normally).
failUrl	string	The opposite to successUrl. When customers for some reason fails through making the order this is where they should land.
forceSigning	boolean	If you want all orders to force signing, set it to true. Normally, this should be disabled.
backUrl	string	Backurls is normally used for hosted flow, where customers can return to the checkout without fulfilling anything.

encodeType	RESURS_URL_ENCODE_T YPES	Default: None. Do not use. This is for extreme cases only, where the url encoding with Resurs Bank Fails.
------------	-----------------------------	---

Set the preferred payment service (flow)

First, we need to prepare EComPHP for which payment flow we want to use.

```
$this->rb->setPreferredPaymentService(ResursMethodTypes::FLOW_HOSTED_FLOW);
```



Override functions keeping compatibility between flows

Note that setAdditionalDebitOfPayment() overrides this, as most of the payload are built to fit all flows, with the base from the simplified shopflow. For example, additionalDebitOfPayment is a simplified shopflow function and does not exist in Resurs Checkout. Therefore, adding a payment spec based on the Resurs Checkout payload, will fail. By this override, you can still add orderlines in the Checkout row format while EComPHP converts the specrows to a format that the simplified shopflow understands.

Prepare the customer

The second step to take is to prepare the customer from a shop. This can be done by manually adding a customer to the internal payload of EComPHP. The data below should be replaced with variables, so not all customers get the name Anders Andersson. This part of the hosted flow module is necessary, since Resurs Bank still wants customer data from the store. Besides, leaving customer fields empty in your store, will probably invalidate a payment anyway.

Billing address

```
$this->rb->setBillingAddress("Anders Andersson", "Anders", "Andersson", "Hamngatan 2", null, "Ingestans", "12345", "SE");
```

If you think that the getAddress-API in Resurs Bank is useful, you don't have to add the customer data manually. Instead, you can use this function to create your customer payload

```
$this->rb->setBillingByGetAddress("198305147715");
```

getPayload()

```
{
    "address": {
        "fullName": "Vincent Williamsson Andersson",
        "firstName": "Vincent",
        "lastName": "Andersson",
        "addressRow1": "Glassgatan 15",
        "postalArea": "G\u00f6teborg",
        "postalCode": "41655",
        "country": "SE"
    }
}
```

Additional required customer data

As we are dependent of the customer phone numbers, e-mail etc, you also need to set up that data for the customer. Since the billing (and delivery) address above is a separate section of the customer block, this additional data are added with setCustomer().

This function needs to set up some data from the client:

- governmentId
- phoneNumber
- cellPhone
- e-mail
- customer type (NATURAL for private shoppers and LEGAL for companies)

This very much depends on which payment method you are choosing and in the simplified flow you need to set up a web based form for the customer where this data can be entered. Take a look at the page [General integration](#), and see the rules to follow on how to set up this form. For some payment methods (like card) not all of the above data is required, while for example INVOICE-payments require them all. EComPHP 1.0 and 1.1 currently has a

collection of "long time deprecation functions", which much likely might help you in the form creation. One of the methods, **getTemplateFieldsByMethodType()**, collects the correct set of form fields for the chosen payment method. For this version serie, you will also get a collection with regular expression rules, for which you can use to validate the form fields so that they are filled in correctly. In the [WooCommerce plugin](#) (at least up to v2.1.0) those methods are used. However, since they are in deprecated: This will not be handled at all in the future (i.e. EComPHP 1.2.x does not contain this ruleset), mostly because forms are a very dynamic thing and payment methods at Resurs Bank may be configured differently compared how the payment solution are actually set up.

```
$this->rb->setCustomer("198305147715", "0808080808", "0707070707", "test@test.com", "NATURAL");
```

If you are using **setBillingByGetAddress**, you can set the first field (**governmentId**) to null, as the payload will set the government id with the help of **getAddress()**.

getPayload()

```
{
    "governmentId": "198305147715",
    "email": "test@test.com",
    "phone": "0808080808",
    "cellPhone": "0707070707",
    "type": "NATURAL"
}
```

Preparing the cart

A payment is nothing without the cart. In our case, the cart is your final cart from your store - the payment specification (orderlines). To create an orderline, you can use **addOrderLine()** for your help, and as Resurs Checkout this function only requires minimum of data to render a correct orderline. The data needed is the following:

Variable	Description
\$articleNumberOrId	The article number or the id (normally a short string that sets the article in your store)
\$description	The longer article description of your
\$unitAmountWithoutVat	The price of your article, as it was single product (one unit)
\$vatPct	The VAT of the product without decimals. If the VAT is 25%, you give 25 as value
\$unitMeasure	Pcs, pieces, st, styck, kg, etc. Used to describe the quantity on the invoice "8 pcs of house hold items", "3 kg of meat"
\$articleType	(Optional) Used for the checkout to specify if the article is shipping, fee or regular. Only needed when it's not a regular article type. Null or empty string works to specify no type.
\$quantity	The quantity

Function: addOrderLine()

```
$this->rb->addOrderLine(
    "HORSE",
    "Stallponny",
    4800,
    25,
    "st",
    null,
    1
);
```

Function: getPayload()

```
{  
    "specLines": [  
        {  
            "artNo": "HORSE",  
            "description": "Stallponny",  
            "quantity": 1,  
            "unitMeasure": "st",  
            "unitAmountWithoutVat": 4800,  
            "vatPct": 25,  
            "type": "",  
            "id": 1,  
            "totalVatAmount": 1200,  
            "totalAmount": 6000  
        }]  
}
```

Special requirements

If you have special requirements of your order, like you need to wait for the fraudcontrol before proceeding, if you need to annul an order immediately if it's frozen, etc, such values goes with extra parameters in the payload. In this step you may want to add things that is normally not included via EComPHP, like how the payment are handled together with callbacks. In short; you can add your own payload before proceeding with the payment. Like this:

```
$myPayLoad = array(  
    'paymentData' => array(  
        'waitForFraudControl' => false,  
        'annulIfFrozen' => false,  
        'finalizeIfBooked' => false  
    ),  
    'metaData' => array(  
        'key' => 'CustomerId',  
        'value' => '1337'  
    ),  
    'customer' => array(  
        'yourCustomerId' => '1337'  
    )  
) ;
```

Note: In EComPHP, the function addMetaData() is used after a created payment, not to confuse with.

Preparing a order reference

Sometimes we refer this as the "preferred payment id". You can set it by yourself, with setPreferredId(), but sometimes - when you don't need this to be specific - you can use the internal function **getPreferredId()** instead. This function are delivered with a default setup, creating order references (string) with a maximum length of 25 characters and an optional prefix. If you have a very high load of shopping in your store, and are worried about the unique ids created with this function, it is also possible to double the uniqueness of the order reference. Giving the command

```
$this->rb->getPreferredId(25, "MYSTORE", true);
```

Will for example create a reference that looks like: **20170512110317-MYSTORE466** (the default format of references is set to **YYYYMMDDHHMMSS-PREFIXRAND**). To get the current set payment reference, you can use **getPreferredPaymentId()**. Running this function with a boolean value of true, the preferred id will be set if it does not already exists.

Creating the order

After following the steps above, you are now ready to initiate the booking. This is done by simply use the function createPayment(). In this example, we have a payment method called "INVOICE", so we will use that as the first parameter in.

```
$Payment = $this->rb->createPayment("INVOICE");
```

Having your own payload as described above? Sure. Just add it:

```
$Payment = $this->rb->createPayment("INVOICE", $myPayload);
```

Successful requests

For hosted flow, the response from \$Payment above, will be an url like [https://test.resurs.com/ecommerce-test/hostedflow/customer-landing-page
/paymentSessionId](https://test.resurs.com/ecommerce-test/hostedflow/customer-landing-page/paymentSessionId) - for your solution, this is the place where you will redirect your customers to finished their order.

On your successUrl, you can now proceed creating the order with either the customer session, a getPayment()-request, or any other solution that your store might offer.

EComPHP: createPayment [RCO]

- Initialize the module
- Successful orders vs unsuccessful orders - Setting rules of signing (Flow requirement)
 - The syntax for setSigning is as follows:
- Set the preferred payment service
 - Preparing for ResursCheckoutJS or similar solutions
- Preparing the cart
 - Function: addOrderLine()
 - Function: getPayload() returns
- Full example

Communicating with the iFrame

Creating payments behaviour for the checkout is very much like the [Simplified shopflow](#), except for the fact that you *almost* only need to display an iframe on the webpage instead of integrated web forms.

Payload notice

As of v1.1.22 (+1.0.22), the payload will reset after each successful call to this method. Prior versions just cleans up the internal orderlines. The new way to handle the payload, opens for multiple calls in the same "EComPHP session".

Initialize the module

```
use \Resursbank\RBEcomPHP\ResursEnvironments;
$preferredEnvironment = ResursEnvironments::ENVIRONMENT_TEST;
$this->rb = new \ResursBank( $this->username, $this->password, $preferredEnvironment );
// or
$this->rb = new \ResursBank( $this->username, $this->password );
$this->rb->setEnvironment($preferredEnvironment);
```

Successful orders vs unsuccessful orders - Setting rules of signing ([Flow requirement](#))

This part works much like [the simplified shopflow](#).

Before proceeding, we need to configure the payment to handle signings; if we for some reason require that your customer signs the payment - this happens for example, if you set a delivery address that mismatches with your billing address. If you have your own requirements (for example, you want to force signing every time a payment are created) this part is also important. Besides, without the rules of signing, Resurs Bank won't accept the payment. But the setup is quite simple. You need to set up two urls for this: One for successful signings and one for signings that fails during the payment process.

setSigning()

```
$this->rb->setSigning("https://mystore.test.com/signingSuccessful", "https://mystore.test.com/signingFailed",
false);
```

In EComPHP v1.1.22 or later you can solve this with lesser confusion (see below). Basically, setCheckoutUrls are passing over the variables to the setSigning()-function.

```
$this->rb->setCheckoutUrls("https://mystore.test.com/signingSuccessful", "https://mystore.test.com
/signingFailed");
```

Running this will prepare for eventually letting the customer sign the payment. The boolean false tells Resurs Bank to not force any signing.

The syntax for setSigning is as follows:

Parameter	Type	Description
successUrl	string	Tells Resurs Bank where your payment, when successful, will land (<i>Thanks for your order</i> , normally).
failUrl	string	The opposite to successUrl. When customers for some reason fails through making the order this is where they should land.
forceSigning	boolean	

		Forced signing. Deprecated and won't have any effect in future releases of the API. DEPRECATED
backUrl	string	Backurl is normally used for hosted flow, where customers can return to the checkout without fulfilling anything.

Set the preferred payment service

First, we need to prepare EComPHP for which payment flow we want to use. In this case, we'll tell EComPHP to use the checkout. To maintain some kind of backwards compatibility, both `METHOD_RESURSCHECKOUT` and `METHOD_OMNI` still works.

```
$this->rb->setPreferredPaymentService(ResursMethodTypes::METHOD_CHECKOUT);
```

Compared to the [simplified shopflow](#) you do not need to prepare much more than some basic data. For example, you can skip all inbound customer data at this point. Data about the customer is handled by the checkout iframe, and could be caught with some help from for example [ResursCheckoutJS](#), depending on how the integration looks.

One other thing you however need though, is a payment reference (orderid) to create the order with. If your store supports the ability to create an order reference before creating the order, you should use this method to set up the iframe. If this is not supported, you can do this in some ways:

1. Create a random, temporary order reference, to build the iframe. During the iframe order completion, lock the submit button ([Get inspired with ResursCheckoutJS](#)), create the order in the store and return the handle to the checkout.
 - a. If you need to rename the order reference to match with your store, you can in this moment also send a backend-signal to Resurs Checkout with help from `updatePaymentReference`
2. Create the order on the way back to the checkout (preferably through a customer landing page)

Both of the above method have their own advantages. Some of them are listed here:

- Creating the order in the store **before** returning the handle might create junk orders in the store that will be left untouched if something happens in the checkout before the landing page (customer aborts, fails or just backs out of the checkout without proceeding)
- Creating the order in the store **after** the checkout process might not create the order fully, if the customer aborts or fails in the process before the landing page

You can of course create the order after the checkout process, if you don't want to script anything, and instead use the advantage of callbacks. This, however, leaves you completely alone with the store API and you need to take control over anything that can happen in the order process by yourself. The only thing you can trust at this moment is the BOOKED callback. When this callback is running, you can do a `getPayment()` with EComPHP and fetch both the required customer info and the orderlines. However, like said here, you need to be very familiar with your stores API and now how to complete a full order without the help of the customer session.

Preparing for ResursCheckoutJS or similar solutions

If you decide to use the "[ResursCheckoutJS](#)"-method, you need to be aware that you also need to include an url in the payload handler that tells the web-browser on which parent URL it can find the iframe source. EComPHP does this for you, by setting a default URL through the webserver variables `$_SERVER['HTTP_HOST']`. However, if you require a "non standard URL" you'll need to use the internal function `setShopUrl()`. As of version 1.0.22/1.1.22 /1.2.0, validation of the shopUrl are included in EComPHP, to make sure that the shopUrl are properly formatted. However, this feature must be activated to get effect.

Setting this:

```
$this->rb->setShopUrl("https://my.shop.url/failing/uri/", true);
```

Will render this, in the final payload:

```
https://my.shop.url
```

You can also run this, to get similar result:

```
$this->rb->setValidateCheckoutShopUrl(); // As of 1.1.22
$this->rb->setShopUrl("https://my.shop.url/failing/uri/");
```

Note: This validation does not validate if the host is real.

Preparing the cart

A payment is nothing without the cart. In our case, the cart is your final cart from your store - the payment specification (orderlines). To create an orderline, you can use `addOrderLine()` for your help, and as Resurs Checkout this function only requires minimum of data to render a correct orderline. The data needed is the following:

Variable	Description
\$articleNumberOrId	The article number or the id (normally a short string that sets the article in your store)
\$description	The longer article description of your
\$unitAmountWithoutVat	The price of your article, as it was single product (one unit)
\$vatPct	The VAT of the product without decimals. If the VAT is 25%, you give 25 as value
\$unitMeasure	Pcs, pieces, st, styck, kg, etc. Used to describe the quantity on the invoice "8 pcs of house hold items", "3 kg of meat"
\$articleType	(Optional) Used for the checkout to specify if the article is shipping, fee or regular. Only needed when it's not a regular article type. Null or empty string works to specify no type.
\$quantity	The quantity

Function: addOrderLine()

```
$this->rb->addOrderLine(
    "HORSE",
    "Stallponny",
    4800,
    25,
    "st",
    null,
    1
);
```

Function: getPayload() returns

```
{
    "orderLines": [
        {
            "artNo": "HORSE",
            "description": "Stallponny",
            "quantity": 1,
            "unitMeasure": "st",
            "unitAmountWithoutVat": 4800,
            "vatPct": 25,
            "type": ""
        }
    ]
}
```

Full example

This example has partially been taken from the EComPHP UnitTest suite and shows a very basic set up for the checkout, to be enabled:

```
$this->rb->setPreferredPaymentService(ResursMethodTypes::METHOD_CHECKOUT);
$iframePaymentReference = $this->rb->getPreferredPaymentId(30, "CREATE-");
$this->rb->addOrderLine(
    "Product",
    "This text describes our product",
    500,
    25,
    "st",
    null,
    1
);
$this->rb->setShopUrl("https://my-iframe-shop.test.com");
$this->rb->setCheckoutUrls("https://google.com/?q=signingSuccessful", "https://google.com/?q=signingFailed",
```

```
false);
$theFrame = $this->rb->createPayment($iframePaymentReference);
// Display the frame
echo $theFrame;
```

Note that the shopUrl must be set in <protocol>://host.name only, or it may fail on the iframe communications level. However, if this rule is honored in later versions of EComPHP, like below, the hostname will be validated and corrected automatically.

```
$this->rb->setShopUrl("https://my-iframe-shop.test.com/this/uri/will/be/removed/in/the/final/result");
```

Output Result for above will be <https://my-iframe-shop.test.com>

Communicating with the iFrame

postMsg is deprecated

Please have in mind that the old legacy iframe communication is deprecated.

To get a more responsive behaviour for your store, it is possible to communicate with the iFrame and create an order on the fly, instead of waiting until Resurs Bank is finished with the order creation. Basically, this method makes it possible to create an order in the background (front-end) before the checkout process are taking place in Resurs e-commerce.

Some advantages of this is that ...

- Your customers may be able to fulfill the order even if the internet connection is broken while completing the flow.
- The checkout will be more responsive and support updates of the shop cart before the checkout

You should read the section [Iframe communication \(v2API\)](#) if you want to know more about this, since you need to handle the incoming data from the iFrame.

Order flow and denied credits while using OmniJS

Since the order has to be created in the shop before it is confirmed and created in e-commerce, it is important that you, if you get any orders where the customer has a denied credit, update the previously created order with the proper address data from the customer. If this is not properly done, there's a risk that a bad customer could hijack a good customer's government ID to fulfill an order.

Failurls in RCO-payload (EComPHP)

The failurl sometimes requires encoding to properly work with redirects from Resurs Bank.

Using a failurl that is fully encoded may however not be entirely compatible with the ecommerce solutions especially when it comes to "not nice urls". A "nice" url is a rest-looking url where no ? and & parameters are required. Instead of <http://www.test.com/index.php?parameter1=true>, a nice URL looks like this:

```
http://www.test.com/index/parameter1/true/
```

To avoid such problems, we can as of march 2019, push encoded urls into the payload. But when we push a **http%3A%2F%2F**-formatted urlstring into the signing/failurl-blocks (in this case in the iframe), Resurs ecommerce responds with a "malformed url"-exception instead of accepting it. So we need to properly encode the url by the path only. However, to make it possible to change this in the future (if this is at some point fixed at Resurs Bank) we use bitwise methods to decide **how** we'd like to encode the url. This is an example from PrestaShop on how we do handle encoded urls:

Using setSigning()

```
$this->ecom()->setSigning(  
    $this->getSuccessUrl(),  
    $this->getBackUrl(),  
    false,  
    null,  
    RESURS_URL_ENCODE_TYPES::FAILURL +  
    RESURS_URL_ENCODE_TYPES::PATH_ONLY  
) ;
```

In this case we ask EComPHP to set the successUrls and failurl, but *not* backurl (it will go null here, since hosted flow is the only flow that wants to know about a backurl).

Besides of this, the last parameter setup tells EComPHP to only encode the failurl. However, since Resurs did not support fully encoded urls, we ask EComPHP to only encode the pathway of the url. By means, the url will be encoded like this:

```
https://www.test.com/%3Fparameter1%3Dvalue1%26parameter2%3Dvalue2.
```

When we go through a test flow where we decide do "mockfail", the redirect will be done as is above. In worse cases, we can also add the bitflag **LEAVE_FIRST_PART**. This means that the PATH will be encoded, but only after the first part of the string. By means, our encoded url will look like this:

```
https://www.test.com/?parameter1=value1%26parameter2%3Dvalue2
```

Both above cases can be tested here:

<https://omnitest.resurs.com/web/dist/fail.html?redirectUrl=https://identifier.tornevall.net/%3Fjson%3Dtrue%26queryParam1%3DTest1%26queryParam2%3DTest2>
<https://omnitest.resurs.com/web/dist/fail.html?redirectUrl=https://identifier.tornevall.net/?json=true%26queryParam1%3DTest1%26queryParam2%3DTest2>

EComPHP: createPayment [simplifiedShopFlow]

- Initialize the module
- Successful orders vs unsuccessful orders - Setting rules of signing (Flow requirement)
 - The syntax for setSigning is as follows:
- Set the preferred payment service (flow)
- Prepare the customer
 - Billing address
 - getPayload()
 - Additional required customer data
 - getPayload()
- Preparing the cart
 - Function: addOrderLine()
 - Function: getPayload()
- Special requirements
- Preparing a order reference
- Creating the order
 - The booked payment response
- Handle signed payments

As said in another chapter of this documentation, EComPHP 1.0.2+ and 1.1.2+ is furthermore simplified. To create a standard order, as shown in our test suite, the steps below are required to build a payment in the simplified flow. The examples are taken from the test suite, so we assume that you do run this in a similar environment. The steps below should render a huge payload, but instead of creating it by yourself, EComPHP can do this for you. If you want to look at the payload rendered by EComPHP after each step, you can use the **getPayload()**-function to view what's creating during the process.

Initialize the module

```
use Resursbank\RBEcomPHP\RESURS_ENVIRONMENTS;
use Resursbank\RBEcomPHP\ResursBank;
$preferredEnvironment = RESURS_ENVIRONMENTS::TEST;
$this->rb = new ResursBank( $this->username, $this->password, $preferredEnvironment );
// or
$this->rb = new ResursBank( $this->username, $this->password );
$this->rb->setEnvironment($preferredEnvironment);
```

Successful orders vs unsuccessful orders - Setting rules of signing (Flow requirement)

This is a part of customer preparing and required by the flow to properly work when it comes to signing/authorizing payments that requires this.

Before proceeding, it is **required** configure the payment to handle signings; if we for some reason require that your customer signs the payment - this happens for example, if you set a delivery address that mismatches with your billing address. If you have your own requirements (for example, you want to force signing every time a payment are created) this part is also important. **Besides, without the rules of signing, Resurs Bank won't accept the payment.** But the setup is quite simple. You need to set up two urls for this: One for successful signings and one for signings that fails during the payment process.

setSigning()

```
$this->rb->setSigning("https://mystore.test.com/signingSuccessful", "https://mystore.test.com/signingFailed",
false);
```

Running this will prepare for eventually letting the customer sign the payment. The boolean false tells Resurs Bank to not force any signing.

The syntax for setSigning is as follows:

Parameter	Type	Description
successUrl	string	Tells Resurs Bank where your payment, when successful, will land (<i>Thanks for your order</i> , normally).
failUrl	string	The opposite to successUrl. When customers for some reason fails through making the order this is where they should land.
forceSigning	boolean	If you want all orders to force signing, set it to true. Normally, this should be disabled.
backUrl	string	Backurls is normally used for hosted flow, where customers can return to the checkout without fulfilling anything.
encodeType		

RESURS_URL_ENCODE_T YPES	Default: None. Do not use. This is for extreme cases only, where the url encoding with Resurs Bank Fails.
-----------------------------	---

Set the preferred payment service (flow)

First, we need to prepare EComPHP for which payment flow we want to use.

```
use Resursbank\RBEcomPHP\RESURS_FLOW_TYPES;
//
$this->rb->setPreferredPaymentService(RESURS_FLOW_TYPES::SIMPLIFIED_FLOW);
```

Override functions keeping compatibility between flows

Note that setAdditionalDebitOfPayment() overrides this, as most of the payload are built to fit all flows, with the base from the simplified shopflow. For example, additionalDebitOfPayment is a simplified shopflow function and does not exist in Resurs Checkout. Therefore, adding a payment spec based on the Resurs Checkout payload, will fail. By this override, you can still add orderlines in the Checkout row format while EComPHP converts the specrows to a format that the simplified shopflow understands.

Prepare the customer

The second step to take is to prepare the customer from a shop. This can be done by manually adding a customer to the internal payload of EComPHP. The data below should be replaced with variables, so not all customers get the name Anders Andersson.

Billing address

```
$this->rb->setBillingAddress("Anders Andersson", "Anders", "Andersson", "Hamngatan 2", null, "Ingestans", "12345", "SE");
```

If you think that the getAddress-API in Resurs Bank is useful, you don't have to add the customer data manually. Instead, you can use this function to create your customer payload

```
$this->rb->setBillingByGetAddress("198305147715");
```

getPayload()

```
{
    "address": {
        "fullName": "Vincent Williamsson Alexandersson",
        "firstName": "Vincent",
        "lastName": "Alexandersson",
        "addressRow1": "Glassgatan 15",
        "postalArea": "G\u00f6teborg",
        "postalCode": "41655",
        "country": "SE"
    }
}
```

Additional required customer data

As we are dependent of the customer phone numbers, e-mail etc, you also need to set up that data for the customer. Since the billing (and delivery) address above is a separate section of the customer block, this additional data are added with setCustomer().

This function needs to set up some data from the client:

- governmentId
- phoneNumber
- cellPhone
- e-mail
- customer type (NATURAL for private shoppers and LEGAL for companies)

This very much depends on which payment method you are choosing and in the simplified flow you need to set up a web based form for the customer where this data can be entered. Take a look at the page [General integration](#), and see the rules to follow on how to set up this form. For some payment methods (like card) not all of the above data is required, while for example INVOICE-payments require them all. EComPHP 1.0 and 1.1 currently has a

collection of "long time deprecation functions", which much likely might help you in the form creation. One of the methods, **getTemplateFieldsByMethodType()**, collects the correct set of form fields for the chosen payment method (**NOTE: The template handlers that has been imported from shopFlow, the old one is deprecated!**). For this version serie, you will also get a collection with regular expression rules, for which you can use to validate the form fields so that they are filled in correctly. In the [WooCommerce plugin](#) (at least up to v2.1.0) those methods are used. However, since they are in deprecated: This will not be handled at all in the future (i.e. EComPHP 1.2.x does not contain this ruleset), mostly because forms are a very dynamic thing and payment methods at Resurs Bank may be configured differently compared how the payment solution are actually set up.

```
$this->rb->setCustomer("198305147715", "0808080808", "0707070707", "test@test.com", "NATURAL");
```

If you are using `setBillingByGetAddress`, you can set the first field (`governmentId`) to null, as the payload will set the government id with the help of `getAddress()`.

getPayload()

```
{
    "governmentId": "198305147715",
    "email": "test@test.com",
    "phone": "0808080808",
    "cellPhone": "0707070707",
    "type": "NATURAL"
}
```

Preparing the cart

A payment is nothing without the cart. In our case, the cart is your final cart from your store - the payment specification (orderlines). To create an orderline, you can use `addOrderLine()` for your help, and as Resurs Checkout this function only requires minimum of data to render a correct orderline. The data needed is the following:

Variable	Description
<code>\$articleNumberOrId</code>	The article number or the id (normally a short string that sets the article in your store)
<code>\$description</code>	The longer article description of your
<code>\$unitAmountWithoutVat</code>	The price of your article, as it was single product (one unit)
<code>\$vatPct</code>	The VAT of the product without decimals. If the VAT is 25%, you give 25 as value
<code>\$unitMeasure</code>	Pcs, pieces, st, styck, kg, etc. Used to describe the quantity on the invoice "8 pcs of house hold items", "3 kg of meat"
<code>\$articleType</code>	(Optional) Used for the checkout to specify if the article is shipping, fee or regular. Only needed when it's not a regular article type. Null or empty string works to specify no type.
<code>\$quantity</code>	The quantity

Function: addOrderLine()

```
$this->rb->addOrderLine(
    "HORSE",
    "Stallponny",
    4800,
    25,
    "st",
    null,
    1
);
```

Function: getPayload()

```
{  
    "specLines": [  
        {  
            "artNo": "HORSE",  
            "description": "Stallponny",  
            "quantity": 1,  
            "unitMeasure": "st",  
            "unitAmountWithoutVat": 4800,  
            "vatPct": 25,  
            "type": "",  
            "id": 1,  
            "totalVatAmount": 1200,  
            "totalAmount": 6000  
        }]  
}
```

Special requirements

If you have special requirements of your order, like you need to wait for the fraudcontrol before proceeding, if you need to annul an order immediately if it's frozen, etc, such values goes with extra parameters in the payload. In this step you may want to add things that is normally not included via EComPHP, like how the payment are handled together with callbacks. In short; you can add your own payload before proceeding with the payment. Like this:

```
$myPayLoad = array(  
    'paymentData' => array(  
        'waitForFraudControl' => false,  
        'annulIfFrozen' => false,  
        'finalizeIfBooked' => false  
    ),  
    'metaData' => array(  
        'key' => 'CustomerId',  
        'value' => '1337'  
    ),  
    'customer' => array(  
        'yourCustomerId' => '1337'  
    )  
);
```

Note: In EComPHP, the function addMetaData() is used after a created payment, not to confuse with.

Preparing a order reference

Sometimes we refer this as the "preferred payment id". You can set it by yourself, with setPreferredId(), but sometimes - when you don't need this to be specific - you can use the internal function **getPreferredId()** instead. This function are delivered with a default setup, creating order references (string) with a maximum length of 25 characters and an optional prefix. If you have a very high load of shopping in your store, and are worried about the unique ids created with this function, it is also possible to double the uniqueness of the order reference. Giving the command

```
$this->rb->getPreferredId(25, "MYSTORE", true);
```

Will for example create a reference that looks like: **20170512110317-MYSTORE466** (the default format of references is set to **YYYYMMDDHHMMSS-PREFIXRAND**). To get the current set payment reference, you can use **getPreferredPaymentId()**. Running this function with a boolean value of true, the preferred id will be set if it does not already exists.

Creating the order

After following the steps above, you are now ready to initiate the booking. This is done by simply use the function createPayment(). In this example, we have a payment method called "INVOICE", so we will use that as the first parameter in.

```
$Payment = $this->rb->createPayment("INVOICE");
```

Having your own payload as described above? Sure. Just add it:

```
$Payment = $this->rb->createPayment("INVOICE", $myPayload);
```

The booked payment response

This is an example of the response that is returned after a successful `createPayment()`. At this moment, the below `bookPaymentStatus` can give you a bunch of different answers. Take a look at [bookPaymentStatus](#) for more information about this.

```
{  
    "paymentId": "20170512112617-0195646528",  
    "bookPaymentStatus": "BOOKED",  
    "signingUrl": null,  
    "approvedAmount": "6000",  
    "customer": {  
        "governmentId": "8305147715",  
        "address": {  
            "fullName": "Vincent Williamsson Alexandersson",  
            "firstName": "Vincent",  
            "lastName": "Alexandersson",  
            "addressRow1": "Glassgatan 15",  
            "postalArea": "G\u00f6teborg",  
            "postalCode": "41655",  
            "country": "SE"  
        },  
        "phone": "+46707070707",  
        "email": "test@test.com",  
        "type": "NATURAL"  
    }  
}
```

Handle signed payments

In some cases, the `bookPaymentStatus` will be "SIGNING". This means you need to handle signings also (which has been mentioned above in the "rules of signing"-section). How? You put up a landing page for where you can handle successful signings. When your customers sign payments, they are redirected back to your store, where you can handle the final part of a signed payment. Normally, Resurs Bank are attaching the parameter you set up in the signing step, so do not forget to add a payment reference when you do this. The payment reference is supposed to be picked back up at the landing page.

```
$Payment = $this->rb->bookSignedPayment("prior_set_payment_id");
```

When this is done, you'll get a new [response](#) from the signing sent back to Resurs Bank. If everything is successful, you may start handling the final parts of the order (like changing order statuses or whatever you need to do to set the order as confirmed). When this is also done, you may redirect your customer to your final "successful landingpage".

EComPHP: DEBIT, CREDIT, ANNUL [afterShopFlow]

- Aftershop Setup
 - Payment specs and finalizations
 - skipSpecValidation (**)
- How to use this
 - Example
 - Partial handling example
- Customizing invoice
- Cancelling full orders

Forcefully change orderlines with other prices

Helper functions

Bad invoiceId setup

Backward compatible functionality (Unsupported - Do no use)

This is a document on how to use the afterShopFlow in EComPHP.



Best practices

- It is a good idea to surround all calls with try- and catch methods, since the NetCurl-library may throw more serious errors during the aftershop handling; i.e. if something goes really wrong, this will happen.
- No second parameter with payload data is required if you plan to use the addOrderLine() functions. Doing this, EcomPHP will render own customized orderRows (so, yes, addOrderLine can be used both for booking and aftershop functions).

Aftershop Setup

Old method	MaxParm	Parameters	Extra information
paymentFinalize	4	paymentId (string, required) customizedOrderRows (array, optional) runOnce (boolean, optional) skipSpecValidation(boolen, optional)	runOnce =Tell EComPHP to only run finalization once. Default is false (*). skipSpecValidation =Allows developers to override the getPayment orderRow validation (v1.3.23+) during aftershop (**)
paymentAnnul	3	paymentId (string, required) customizedOrderRows (array, optional) skipSpecValidation(boolen, optional)	skipSpecValidation =Allows developers to override the getPayment orderRow validation (v1.3.23+) during aftershop (**)
paymentCredit	3	paymentId (string, required) customizedOrderRows (array, optional) skipSpecValidation(boolen, optional)	skipSpecValidation =Allows developers to override the getPayment orderRow validation (v1.3.23+) during aftershop (**)
paymentCancel	3	paymentId (string, required) customizedOrderRows (array, optional) skipSpecValidation(boolen, optional)	skipSpecValidation =Allows developers to override the getPayment orderRow validation (v1.3.23+) during aftershop (**)

* = The runOnce parameter is default false, which normally is used to prevent invoice numbering issues. Running once means that EComPHP will throw an exception if finalization throws an exception in the first run, and the error code is 29=invoice already exists. Running this twice (default) will make EComPHP try to correct the invoice numbering problem.

Payment specs and finalizations

Normally when finalizing orders, Resurs Bank API explicitly wants each order row set properly on finalization and the API don't accept "anonymous order row finalizations" even though that it is possible to send a finalization request without the orderlines. When this is done in the API, an extra order row will be forced on Resurs Bank side. Invoices handled this way will look like the image below.

Betalningsreferens/OCR
8538792891

Betalningsvillkor
Dröjsmålsränta

Förfallodatum
2021-09-30

Betalnummer
SUB000000132

Dröjsmålsränta
20 %

Art.Nr	Beskrivning	Antal	A-pris inkl.moms	Moms	Belopp
1	-	1	100.00	0.00 %	100.00
			Total SEK exkl. moms		100.00 kr
			Momsbelopp		0.00 kr
			Totalt SEK inkl. moms		100.00 kr

By default EComPHP won't accept this either. But if there are circumstances that really require this, it is possible to enforce this in EComPHP from version 1.3.56 like this - with a flag set with `setFinalizeWithoutSpec()`:

```
// Perform partial debit.  
if ($this->isPartial($commandSubject, $data)) {  
    // Flag ECom to drop specLine data (remove payment lines).  
    $connection->setFinalizeWithoutSpec();  
    // Add payment line for entire amount to debit.  
    $connection->addOrderLine(  
        '',  
        '',  
        $this->getAmount($commandSubject)  
    );  
}  
// Capture payment.  
$connection->finalizePayment($paymentId);
```

skipSpecValidation (**)

When running payment credit/annul/debit, we usually want to use a getPayment-object to validate the order content. This means you can run the above functions in a minimalistic mode (**like: paymentCredit(orderid)**). EComPHP handles the rest of the order and credits the correct orderrows in the payment, by validating quantity and articles in the order. If you run the same function with a custom set of articles, EComPHP will still validate that you are allowed to credit a payment, with proper quantity and price. However, some ecommerce platforms has customized discounts that is rather spread as a one-row discount, where the order in "payment admin" differs from the payment in the platform orderview. With the skipSpecValidation EComPHP enters a sloppy mode where it allows you to change the price and quantity to whatever you need to perform the action with minor interferences with the payment.

All methods are set to return a boolean - true for successful and false for failed. However, it is a good idea to try and catch the calls.

The new methods works basically the same as the prior versions. We've removed all unnecessary parameters from the function (creditParams, quantityMatch, useSpecifiedQuantity do no longer have any effect on the calls). This means, finalizePayment() will only pass necessary parameters to the replacements and the trust, that ordrows are sent properly, lies at the developer. The goal is to handle the calls as wrapper, where we pass what you want, to Resurs e-commerce.

During the deprecation period, the old methods will keep the current syntax. From v1.2.0 **both xPayment() and paymentX() will act in the same way**.

How to use this

The new way to handle the aftershop can be done in two ways:

1. Use the internal functions, the same way as createPayment are executed (with addOrderLine, then send your wishes to each function)
2. Run the flow with a custom order array (backward compatibility), where you put your specrow as before, in an array, and send it (both can be combined, however, unexpected result **may** occur)

To run "safe" with the afterShopflow, it is currently recommended, that you use internal functions. The aftershop flow supports following aftershop behaviours:

- Handling of the full order (this is the simplest way, to finalize-credit-annul a complete order)
- Handling partial orders (finalize-credit-annul parts of an order, based on order rows)
- Handling partial orders (finalize-credit-annul parts of a row, based on quantity)



Payload notice

As of v1.1.22 (+1.0.22), the payload will reset after each successful call to this method. This opens for multiple calls in the same "EComPHP session".

Example

Since the functions act the same, independently on the request you're making, here's a short example on how to use the finalize function. If you need more inspiration, should take a look at the test suite, where there's a lot of tests for this flow.

To finalize one order, regardless of the content, the only command you need to run, is this (with the payment id):

```
// Finalization
$this->rb->paymentFinalize($paymentId);
// Annul
$this->rb->paymentAnnul($paymentId);
// Credit
$this->rb->paymentCredit($paymentId);
// Let ecom decide (if the payment contains partially various statuses)
$this->rb->paymentCancel($paymentId);
```

In this case, EComPHP will finalize whatever it finds in the current payment, based on what's not already finalized.

Partial handling example

To finalize a part of an order, you can add the rows you'd like to finalize like in the example below. EComPHP will in this case handle the orderLine-array for you - and your code might look a little bit more structured:

```
$this->rb->addOrderLine( "myAdditionalManualFirstOrderLine", "One orderline added with addOrderLine", 100, 25,
'st', 'ORDER_LINE', 2 );
$this->rb->addOrderLine( "myAdditionalManualSecondOrderLine", "One orderline added with addOrderLine", 100, 25,
'st', 'ORDER_LINE', 2 );
// Finalization
$this->rb->paymentFinalize($paymentId);
// Annul
$this->rb->paymentAnnul($paymentId);
// Credit
$this->rb->paymentCredit($paymentId);
```

Customizing invoice

If you need to customize the customer invoice with for example a customer id, you can use setCustomerId(), before running the commands listed here. Basically, adding any metadata could be used through the addMetaData()-function, but setCustomerId() makes all necessary moves on fly.

Cancelling full orders

It is still possible to cancel orders, though the special function paymentCancel (**cancelPayment** in the old versions). In the contrary to the other three functions (debit, credit, annul), this function validates the content you send into the payload. Mostly since this function is made for annulling rows that has only been authorized and credit rows that have been debited.

Forcefully change orderlines with other prices

The above example shows how to "just push in orderdata into a finalization". But what happens when the orderlines only should be partially updated? Like this:

```
$this->rb->addOrderLine( "myAdditionalManualFirstOrderLine", "One orderline added with addOrderLine", 50, 25,
'st', 'ORDER_LINE', 1 );
$this->rb->addOrderLine( "myAdditionalManualSecondOrderLine", "One orderline added with addOrderLine", 50, 25,
'st', 'ORDER_LINE', 1 );
```

Normally a finalization in this state, where the original orderrows has another values, EComPHP will try to fix this problem and change the amounts to corrected values. However, in this special case, the price is also different to the original above. EComPHP normally tries to correct this to, so to force new price values into a payload like this you can instead do this:

```
$finalizeResult = $this->rb->paymentFinalize( $paymentId, null, false, true );
```

As one of the parameters in the finalization is different to credit/annul, there's an extra boolean value to pass over to EComPHP (the false value).

```
* @param $paymentId
* @param array $customPayloadItemList
* @param bool $runOnce Only run this once, throw second time
* @param bool $skipSpecValidation Set to true, you're skipping validation of orderrows.
```

But as you can see at the in-parameters above, since v1.3.23, a new parameter (\$skipSpecValidation) is added. Using this you can bypass internal validation and push in your own values.

This works with both crediting/annulling too, but the \$runOnce-parameter is not required for the other calls.

Helper functions

There are a bunch of helper function available in EComPHP to find out the status of a payment. This helps the developer, for example, to quickly find out if a payment is debitible or not. The function are listed below

Function	Description
canCredit	Returns true if a payment is still creditable
canDebit	Returns true if a payment is still debitible
canAnnul	Returns true if a payment is still annullable This function is based on canDebit - if a payment is debitible, the order can also be annullable
getIsDebited	Returns true if a payment is debited
getIsCredited	Returns true if a payment is credited
getIsAnnulled	Returns true if a payment is annulled

Bad invoiceId setup

As of EComPHP v1.3.27, due to some discovered problems with how invoice id's sometimes are handled we change the way how this is set during afterShop. In 1.3.7, we will in default mode drop much of the support of how invoice id's are set in the flow. Prior versions statically located an invoice id by [Peek Invoice Sequence](#). When, i.e., finalization was running and ECom got a SoapException (code 29), ECom also tried to heal itself by looking for a new invoice id automatically. This behaviour is now changed and ECom will only try to set the ID if no prior ID's are set before. If something goes wrong, ecom is dependent on exceptions, and we believe that it is safer to run this way. Besides, normally, merchants should not touch basic settings in the portals. However, if you really require ECom to do the prior checkups and set static invoice ids in the payload, you can easily do this by flagging the steps. The flags follow below. The values are always boolean.

setFlag(key)	Behaviour
AFTERSHOP_STATIC_INVOICE	Fall back to basic behaviour. Find invoice id, set it, run afterShop with this. On exception code 29, ECom tries to repair by looking up prior invoice id's and go with it. Once. If this fails, a major exception are thrown back. And then it stops there.
AFTERSHOP_RELEASE_INVOICE	Failsafe mode for AFTERSHOP_STATIC_INVOICE. Aftershop acts the same above, but till try to search and rescue for a second id.

Backward compatible functionality (**Unsupported - Do no use**)

If you decide to go with the backward compatibility way, more responsibility lands on the integrator that has to make sure that the orderlines are correct. Using this method, a payment finalization, can look like this (**not best practice**). This example is based on finalization:

```
$orderLineArray = array(
    array(
        'artNo'           => 'myAdditionalManualFirstOrderLine',
        'description'     => "One orderline added with addOrderLine",
    )
);
```

```

        'unitAmountWithoutVat' => 100,
        'vatPct'                => 25,
        'quantity'              => 1
    ),
    array(
        'artNo'                 => 'myAdditionalManualSecondOrderLine',
        'description'           => "One orderline added with addOrderLine",
        'unitAmountWithoutVat' => 100,
        'vatPct'                => 25,
        'quantity'              => 1
    ),
);
$this->rb->paymentFinalize($paymentId, $orderLineArray);

```

The \$orderLineArray in this example do support recursive arrays as above. If you by mistake miss this (when you only have one article to finalize for example) EComPHP will handle this too, by first convert the array into recursion.



Can I do both?

Yes. This method, can be combined with the above examples with addOrderLine()

Instant FINALIZATION / Bitwise constants (EComPHP)

Table of contents

- What is this, really?
 - I don't want it
 - I do want to use it
- How to programmatically use this solution
 - Best practice with switch/case
 - Is the order the values are handled in important?

What is this, really?

The method we're using here is based on the binary system and each value in the bitmasking table is set with a value. Normally, this is just "regular numbers" that could be used out of the box.

For instant finalizations the table could look like this:

128	64	32	16	8	4	2	1
off	off	on	off	off	on	off	off
-	-	autodebited	credited	annulled	completed	processing	pending
0	0	1	0	0	1	0	0

If you have started using payment methods where customer payments are instantly transferred to you as a merchant (SWISH, Vips, direct bank transfers etc), you'd probably already realized that the flow behaviour have started to act different. In some platforms finalization means that orders are fully handled, delivered and debited which also means, if a callback acts like this your order might be updated long before you've actually handled it (the woocommerce plugin works like this). In the above example, EComPHP will return status 36 (since $32+4=36$) if this function is enabled. This enables for status deviations at Resurs Bank, that shop systems normally will be able to handle otherwise.

Since: EComPHP 1.3.14, 1.0.41 and 1.1.41 via RESURS_PAYMENT_STATUS_RETURNCODES.

I don't want it

Then you don't have to read furthermore after this section. By sending a `$ECOM->setAutoDebitableTypes(false)` you can make ECom behave as in prior version. By means, you disable the behaviour. However, this way of handling instant finalizations is idle whatsoever, as long as you use payment methods that is not actively supports the behaviour (SWISH, INTERNET, "Direkt Bankbetalning", etc).

I do want to use it

Everything starts in `getOrderStatusByPayment()` regardless of the callback.

PHP-modules that uses ECom correctly triggers this method on each inbound callback so that ECom, based on the inbound payment, decides which status that will be used. There are some exceptions - FINALIZATION is one of them. When FINALIZATION is triggered, the normal behaviour to follow is to just return `RESURS_PAYMENT_STATUS_RETURNCODES::PAYMENT_COMPLETED`. ECom will continue to do this, and in newer versions of ECom `PAYMENT_COMPLETED` has the constant value **4** instead of **30**.

- During the FINALIZATION process, ECom will also check the current payment method that was used in the order, but only if the type was defined as `PAYMENT_PROVIDER`.

This means we'll experiencing as less performance as possible. Once finding a PAYMENT_PROVIDER, ECom will also make a short background check of the method to validate of which extended type it is and if it is potentially a "automatic finalizer"-method. If that's the case,

- ECom will together with `PAYMENT_COMPLETED` (4) also return the constant `PAYMENT_AUTOMATICALLY_DEBITED` (32) in the same (bit) value, if it discovers that the payment method is based on `PAYMENT_PROVIDER` and the payment method is flagged as a method that instantly finalizes the payments.

The final result gives us the value 36 (`RESURS_PAYMENT_STATUS_RETURNCODES::PAYMENT_COMPLETED | RESURS_PAYMENT_STATUS_RETURNCODES::PAYMENT_AUTOMATICALLY_DEBITED`). For "normal" payments, the value will still be 4 (`PAYMENT_COMPLETED`).

How to programmatically use this solution

Best practice with switch/case

Bitmask Comparisons

```
// $suggestedStatus returned from ecomphp

switch (true) {

/* Other cases */

// If the suggested status from EComPHP matches with the flag completed (FINALIZED or fully debited) act from
the same condition
case $suggestedStatus & (RESURS_PAYMENT_STATUS_RETURNCODES::PAYMENT_COMPLETED |
RESURS_PAYMENT_STATUS_RETURNCODES::PAYMENT_AUTOMATICALLY_DEBITED):

    // If the suggested status from EComPHP has been flagged with PAYMENT_AUTOMATICALLY_DEBITED
    if ($suggestedStatus & RESURS_PAYMENT_STATUS_RETURNCODES::PAYMENT_AUTOMATICALLY_DEBITED) {
        $order->update_status('another_status_than_completed');
    } else {
        // ... or act normally
        $order->update_status('completed');
    }

    // Return the status code to someone that needs it.
    return $suggestedStatus;

/* Other other cases */

}

}
```

Before the release of the changes in ECom, the below solutions are examples on the effect with older solutions. In the first example, the instant finalization will be ignored and the code will preferably ignore everything if the order has been both instantly finalized and debited. In another way, this only covers regular payments without instant debits.

The simple old format

```
if ($finalizeStatus === RESURS_PAYMENT_STATUS_RETURNCODES::PAYMENT_COMPLETED) {
    // The normal control
}
```

And finally, this is the hard way, which compares the values separately "as is", checking both with old syntax:

```
if ($finalizeStatus === RESURS_PAYMENT_STATUS_RETURNCODES::PAYMENT_COMPLETED || 
    $finalizeStatus === RESURS_PAYMENT_STATUS_RETURNCODES::PAYMENT_COMPLETED +
RESURS_PAYMENT_STATUS_RETURNCODES::PAYMENT_AUTOMATICALLY_DEBITED
) {
    // Act regardless of finalization status
}
```

Is the order the values are handled in important?

It depends on how you use them. If you want to break on first matching status, the order could make a difference. If you look for more results than just one - for example in the "Instant Finalization" (where two values can be set) - and you split each status into separate cases, you should definitely not use breaks after the first matching status.

EComPHP: getCostOfPurchaseHtml (Automated)

Reference: Get Cost of Purchase (HTML)

A method to retrieve detailed cost of purchase information in HTML format. EComPHP has a function to automate this with proper HTML-headers and CSS.

CSS

If \$returnBody is true and \$callCss is in use, this is the base content of the CSS that is needed by default:

- body { }
- .header { }
- .headHeader { }
- .headText { }
- .priceTable { }
- .evenRow { }
- .oddRow { }
- .content { }
- .tailText { }
- .legalInfoLink { }

CSS generated by \$returnBody

- .cost-of-purchase-box { }

Usage example

As the function states here, you may set \$returnBody to true to get standard HTML-tags with head and body from the call.

```
getCostOfPurchase($paymentMethod = '', $amount = 0, $returnBody = false, $callCss = 'costofpurchase.css', $hrefTarget = '_blank')
```

Sample CSS file for getCostOfPurchase (from the WooCommerce-plugin)

getCostOfPurchaseHtml()

```
$costOfPurchaseHtml = $flow->getCostOfPurchase("INVOICE", 1337, true, "/mysite/css/costofpurchase.css", "_blank");
return $costOfPurchaseHtml;
```

EComPHP: setAdditionalDebitOfPayment() - Update payments with additional data

In the simplified flow, [Additional Debit of Payment](#) will give you the opportunity to add more orderlines to your orders after it has been created. As of EComPHP 1.0.3 and 1.1.3, this is also supported by the library. To make it easy to add more orderlines to an order, we are using our own internal function `addOrderLine()`. You might want to take a look at [EComPHP: createPayment \[simplifiedShopFlow\]](#) to find out more about this. In 1.0.3/1.1.3, after creation of an order, EComPHP will clean up the internal storage of the payment spec, as soon as it has been successfully booked. This means that you can reuse that function to put in additional lines:

```
$paymentId = "myPaymentId";
$this->rb->addOrderLine(
    "extraLine",
    "One another forgotten orderline",
    150,
    25,
    "st",
    null,
    1
);
// With setLoggedInUser() you can also set up a user identification for the createdBy-parameter sent with the
// additional debig. If not set, EComPHP will use the merchant credentials.
$this->rb->setLoggedInUser("Shrek");
$this->rb->setAdditionalDebitOfPayment($paymentId);
```

Ad you can see, `setAdditionalDebitOfPayment` does not require any other parameter in than the payment that needs the extra orderline. Making sure we're using the `setLoggedInUser()`, EComPHP will also set the `createdBy` parameter to the administrator that added the extra order line. If this is not being set, EComPHP will instead use the configured merchant credentials (which might be a little bit of lacking sometimes).

Payload notice

As of v1.1.22 (+1.0.22), the payload will reset after each successful call to this method. This opens for multiple calls in the same "EComPHP session".

EComPHP and localization

Table of contents

- Current (May 2022) translation string buildup
 - Keywords to look for
- Description
- Language Names
- How to use

Current (May 2022) translation string buildup

The translation segment in EComPHP is very complex. From an external point of view, translations may look hardcoded as phrases and keying in the language files looks quite inconsistent. There is a non-hardcoded-ish structure, but it is mostly driven internally and it is therefore impossible to statically decide which payment method is connected to a specific phrase, since they are probable to move around during development. Also, the phrases are mainly built for Resurs Checkout, so internally this will never become a problem. There are plans to restructure this lineup of code, so in an ECom aspect, this has to be taken care with caution. If the languages in future releases are changing it is highly important that the requesting methods can handle them as of today.

Below is an explanation of how the phrases are built *today*. The translation place is entirely depending on the internal location, country and system in use:

- Resurs-owned payment methods is used to be built on subTitle. Normally, ecomphp is extracting **infoText1** and **infoText2**, which is sometimes not the correct keys to use when it comes to the internals. The characteristics of an internal method is based on the initial name RESURS. For example, part payments comes both as RESURS_PART_PAYMENT and partPayment in the language files now. To extract the proper phrase, the best way is to match type with what's not PAYMENT_PROVIDER and when not, use RESURS_PART_PAYMENT as the proper payment method.
- The above method leads us into the PAYMENT_PROVIDER handing. Most of the named payment methods contains psp as an initial string. Like pspTrustly, pspCard, pspSwish, etc. For pspTrustly there are however some very specific rules applied, at least for when we use getPaymentMethods; Trustly are identified as specificType INTERNET, not TRUSTLY. But since there are also a psplInternet defined in the language files there may be inaccurate hits on that phrase collection. For those currently checking payment methods, the phrase should be easiest to fetch from pspTrustly.
- During translation fetching, we look for data stored based by **specificType**. There was an idea to also check **type**, when specificType does not return data. But in the most cases methods based on type is also returning empty data when specificType is not present. But we usually do a failovercheck on this too, in case there is changes for this in the future.
- Is the payment trusty? Go for pspTrustly instead of psplInternet, since the info string is **currently** located in the pspTrustly-block.

For more information of how this is digged through, most of the above (and below) is described in swedish at

[P17-307](#) - Synka betalmetodernas texter del 3 DONE where it was discovered first.

Keywords to look for

As this text is written, we are working on the PrestaShop module (simplified) for this to work properly. Normally when we're looking for phrases we use ecom internals, which is using infoText1 and infoText2 to find the strings. However, the key **info** is also sometimes necessary (pspTrustly is using that for SE, but not for the other countries). Also, for internal methods, we look for subTitle. So to sum up this we collect:

- infoText1
- infoText2
- info
- subTitle

Description

Resurs Bank is giving us the opportunity to export a bunch of locales with help from language files where phrases are stored. We've picked this feature up, and those language files are embedded in the EComPHP source. The files are stored in src/Service/Container and for Swedish, Norwegian and Finnish phrases are stored in json containers. Danish is stored as XML.

```

1 {
2   "common": {
3     "Change": "Ändra",
4     "ChoosePaymentMethod": "Betalsätt",
5     "Get": "Hämta",
6     "GetYourInformation": "Dina uppgifter",
7     "NotYouQ": "Inte du?",
8     "close": "stäng"
9   },
10  "userInfo": {
11    "manualInputText": "Du kan också",
12    "manualInputLinkText": "fylla i dina uppgifter manuellt",
13    "govIdInputText": "Mycket att fylla i? Låt oss",
14    "govIdInputLinkText": "hämta dina uppgifter",
15    "placeholderGovId": "ååååmmdd-nnnn",
16    "helpGovId": "Med hjälp av ditt personnummer kan vi slå upp dina adressuppgifter.\nDessa behövs för",
17    "labelFirstName": "Förnamn *",
18    "labelLastName": "Efternamn *",
19    "labelAddress": "Adress *",
20    "labelAddressInfo": "Adressinformation",
21    "labelPostalCode": "Postnummer *",
22    "labelCity": "Stad *",
23    "labelEmail": "E-postadress *",
24    "labelPhone": "Mobiltelefon *",
25    "labelAlternativeDeliveryAddress": "Annan leveransadress",
26    "infoAddressBlock": "Dessa uppgifter behövs för att vi ska kunna skicka din beställning. Var extra",
27    "infoDeliveryAddressBlock": "Att skicka en beställning till en annan leveransadress än din folkbok"
28  }

```

Language Names

Resurs are using ISO 639-1 based locale names (sv, da, no, fi, etc), instead of ISO 3166 (SE, DK, NO, FI, etc). The translation class is however covers both of the formats when requesting a locale (see below).

How to use

Our testsuite is covering two ways fetching translation phrases. The simplest way is to just request for a key and to make ECom choose the proper language you instantiate a class called Translation.

```
$helper = new Translation();
$helper->setLanguage('sv');
```

When the locale is selected, you can start request a phrase by doing one of the below:

```
$closePhrase = $helper->getPhrase('Change');
$validationForEmptyGovId = $helper->getPhrase('forEmpty', ['validation', 'govId']);
```

By looking at the json files you can also see where the data is stored and if you find (like the above example) that the phrases are stored recursively where ECom can not find the phrases itself (it won't do the recursion for you), the second argument helps you define a array-path to the phrase location.

If you know what you are looking for, like payment methods, there's however faster ways getting some of the phrases, by the customized getMethodInfo and getPhraseByMethod. getPhraseByMethod will look for a payment method and return a block of data relating to that method (very often based by type or specificType). getPaymentMethodInfo if a very strict checker and will only return data on very specific keys. Currently the keys are based on "infoText1" and "infoText2".

```
$partPaymentInfo1 = $helper->getMethodInfo('partPayment');
$partPaymentInfo2 = $helper->getMethodInfo('partPayment', 2);
$pspTrustly = $helper->getPhraseByMethod('pspTrustly');
```

EComPHP features and tips

This page will contain all special features that is not always obvious that EComPHP supports

- This is EComPHP
- List of special features
 - Opening for more order line freedom
 - Tweaking
 - Feature list
 - Payment methods in simplified shopflow vs Resurs Checkout
 - SOAP-calls is cachable
 - getPayment is cached
 - getCostOfPriceInformation alternative to getCostOfPurchaseHtml
 - Showing price information for denmark
 - Invoice sequence handling are changed
 - Reachable curl handle
 - Units
 - Other fixes
 - Debugging createPayment
 - Custom shopUrl for Resurs Checkout
 - Preferred payment ID's
 - setPreferredId(<string>)
 - getPreferredPaymentId()
 - setCountry
 - Special payment functions and aftershop features
 - getAnnuityPriceByDuration
 - getPaymentDiffByStatus
 - getPaymentSpecCount
 - sanitizeAfterShopSpec
 - User and client
 - EComPHP Client identification
 - Client Identification - SubFeature
 - User and application identification
 - Resurs Bank payment order Statuses, how do I handle them?
 - getOrderStatusByPayment
 - RESURS_PAYMENT_STATUS_RETURNCODES and how they are used
 - Callbacks
 - Multiple stores
 - Extended features
 - getCostOfPurchase
 - setCostOfPurchaseHtmlBefore and setCostOfPurchaseHtmlAfter
 - External extended URL validation
 - Invoice sequence numbering
- Undocumented features
 - setCustomerIpProxy
 - ShopFlow cloned features (deprecated)
 - setFormTemplateRules
 - getRegEx
 - canHideFormField
 - getTemplateFieldsByMethodType
 - getTemplateFieldsByMethod
 - getFormFieldsByMethod

This is EComPHP

- EComPHP is a wrapper that transforms data into the right format before handling it. The library of curl used is side-by-side-compatible with both rest and soap so EComPHP can utilize and jump freely between both old and new services.
- In some cases, for example when working with aftershop features, EComPHP runs multiple commands with Resurs Bank to get current states (i.e. getPayment, getPaymentMethods)
- Some aftershop functions have shopflow overriders (Example: Running in Resurs Checkout-mode, still requires parts of the simplified flow, when running aftershop - ecomphp have overrides to handle this automatically)
- Many features are cross-flow based
 - Like above example: the afterShopFlow webservice is using simplifiedShopFlow to find out states in the payments
 - Registering callbacks in checkout rest mode is sometimes using SOAP and configurationService to handle callbacks

List of special features

Opening for more order line freedom

As of v1.3.76 we've been opening the ability to manipulate order rows outside the ruleset of ECom's own business logics. There are moments when you really need to push data through the API without the validations the ECom does on each order row (which is built in to prevent bad actions). There are an example in the test suite that we have been running manually to confirm moments when this is required (named `afterShopOverride`) and includes two specific methods to disable internal validation rows:

Function	Description
<code>setGetPaymentValidation</code>	Feature designed for afterShop actions like creditPayment, annulPayment, etc. Disables order row validation entirely so that any data will be accepted as valid.
<code>setBookPaymentValidation</code>	Feature designed for bookPayment. Disables validations on duplicate rows, where the duplicate validator usually merges identical order rows into one, with updated quantity and order totals. This is more or less a fix from a legacy system where the specLine-setup was built differently and where PHP could not handle identical product lines properly. By disabling this validation, you can push through whatever you need, line by line.

Tweaking

EComPHP is tweakable by its internal flagset. When initializing the library you can use a `$library->setFlag()`, to enable features that is normally not active. Here's a list of those flags.

As of 1.0.23, 1.1.23 and 1.2.0, the ECom library supports setup of configurable flags that could be picked up again, in external applications. This makes it possible to control parts of ECom from time to time and from 1.2.2/1.3.2 (1.1.29/1.0.29), we've started up to use those flag internally too. From those versions of the library we also starting to use session variables, so some of the flags can be passed between different calls in a web application. The internal flags and their usage are presented below, and can be set by `setFlag($key, $value)`, `getFlag($key)`, `deleteFlag($key)`. Using session variables, is being made with `setSessionVar($key, $value)`, `getSessionVar($key)` and `deleteSessionVar($key)`.

As NetCURL 6.1 was released a "generation 2 flagset" became available through the `Flag::class`.

	inValue	Information/Since	Description
<code>PREVENT_EXEC_FLOOD</code>	true	DEPRECATED	This flag prevents the <code>createPayment()</code> -method to run too fast. It uses session variables (PHP: <code>\$_SESSION</code>) to set a timestamp for which <code>createPayment</code> was last runned, and throttles the payment creation if next <code>createPayment</code> is executed within 5 seconds (default)
<code>PREVENT_EXEC_FLOOD_TIME</code>	integer	DEPRECATED	If 5 seconds is to little to prevent <code>createPayment()</code> -execution, this flag will replace the default of 5 seconds to another value. Example: <code>setFlag('PREVENT_EXEC_FLOOD_TIME', 10);</code> In this example, next <code>createPayment()</code> for the current session might not be executed during a time of 10 seconds instead.
<code>PREVENT_EXEC_FLOOD_EXCEPTIONS</code>	true	DEPRECATED	Activating this flag with true, will allow exceptions to be thrown during <code>createPayment()</code> - otherwise, the next <code>createPayment()</code> if it is in the range of <code>PREVENT_EXEC_FLOOD_TIME</code> , will be dropped silently. If you do set this value to true, you must handle exceptions yourself.
<code>SKIP_AFTER_SHOP_INVOICE_CONTROL</code>	false	N/A	(Default: false) Prevents - if set to true - afterShop functions finalize, annul and credit to repair invoice numbering if error 29 occurs during the process. If enabled <code>ALREADY_EXISTS_INVOICE_ID</code> is checked, and EComPHP will try to find the last increment value and set things right.
<code>GET_PAYMENT_BY_REST</code>	true	N/A	Asks EComPHP to use <code>getPayment</code> via REST instead of afterShop-SOAP. In prior versions this setting was reverse (<code>GET_PAYMENT_BY_SOAP</code>), but as there was too much backfires in the errorhandling, we chose to use SOAP as default instead.
<code>CREATED_BY_NO_CLIENT_NAME</code>	true	N/A	Allow clients to skip clientname (if the client name is confusing in payment admin) by setting flag <code>CREATED_BY_NO_CLIENT_NAME</code> . If flag is unset <code>ecomphp-<DECIMAL_VERSION_NUMBER_></code> will be shown. If unset EComPHP will first try to use a name by a logged in user (if set by client) and then fall back to the username <code>EComPHP-RemoteClientAction</code> .
<code>ALWAYS_RENDER_TOTALS</code>	true	N/A	Makes EComPHP, in <code>renderPaymentSpec()</code> , to always render new total amounts instead of trusting inbound payload data. However, the new functions that handles <code>paymentDiffs</code> usually also recalculates the totals (since 1.3.23) since quantity often tend to change during aftershop actions.
<code>USE_AFTER_SHOP_RENDERING</code>	true	N/A	Internally used by EComPHP to be able to skip exceptions when there are no payload set, where the internal methods still needs the rest of the payload (specLines, etc).
<code>KEEP_RCO_BILLING</code>	true	N/A	In Resurs Checkout mode, make EComPHP not strip off <code>[customer=>'billing']</code> from the payload. Currently not in use, just prepared. The fields used is the same as those that you can find in the simplified flow.
<code>KEEP_RCO_DELIVERY</code>	true	N/A	In Resurs Checkout mode, make EComPHP not strip off <code>[customer=>'deliveryAddress']</code> from the payload. Use to prefill (push in own customer information) the iframe. The fields used is the same as those that you can find in the simplified flow.
<code>SKIP_AFTER_SHOP_VALIDATION</code>	true	N/A	Used to override <code>getPayment</code> orderrows during aftershop handling (for example, when you have own prices on prior rows that has to be forcefully changed).
<code>HEAL_URL</code>	true	1.3.47	If you during a payment creation in hosted flow gets a landingpage-url from the API that contains http instead of https, this flag will rewrite the url to a proper https-link before returning it to the client. The features is currently only for hosted flow as we've only seen it there.

Feature list

Payment methods in simplified shopflow vs Resurs Checkout

Historically (for ecom 1.3 at least), simplified shopflow has not always supported payment providers. To not break standard compatibility by displaying unsupported methods in a checkout, EComPHP chose to hide them by default. Today, this is different but the `getPaymentMethods`-method called from ECom is still honoring PSP methods. This happens since payment providers supports government-id-less submission forms, which Resurs internals require. To activate full support for all payment methods (requires that you can handle the lack of government id), you can use `setSimplifiedPsp(true)`.

```
$connection->setSimplifiedPsp(true);
```

if you'd like to follow the prior restrictions in simplified even when you run RCO-mode, you can use the `setStrictPsp(true)` instead. As we today support PSP in the simplified shopflow, those parts are deprecated but still active. This could change in future releases.

SOAP-calls is cachable

With start at 1.3.40, the communication interface with Resurs Bank is updated, which means there is a big raise in performance. Support for cached wsdl is enabled with this kind of method call:

```
$connection->setWsdlCache(true);
```

Furthermore the ECom will be able to set itself in production- or testmode, which means cached SOAP calls will be enabled in production environments, while test will still run uncached in case of updates in the webservices.

getPayment is cached

In EComPHP a bigger optimization has been done with the `getPayment` method. This means that if you do a regular `getPayment` out of the box, the response is stored unless you request specifically to make a whole new request. The primary reason for this is to limit request per browser lookup. In WooCommerce for example `getPayment` are being made in several places during one load, making EComPHP make at least 8 requests to the API. If EComPHP store cached payment information this won't happen. This is considered a performance fix. To make uncached requests, the call should look like this:

```
$ecom->getPayment('paymentid', false);
```

- Observe that this does not apply to new instantiated calls. For each instance of EComPHP that is created, a new request will be sent through the API.
- Also observe that each call to status update features in ecom is not cached.

getCostOfPriceInformation alternative to getCostOfPurchaseHtml

As of 1.3.30, we are starting to utilize `priceInfo` as an alternative to `getCostOfPurchaseHtml`. Three standard templates are added to the library that executes a similar view as `getCostOfPurchase`. The function itself looks like this:

```
/*
 * @param string $paymentMethod Payment method as string or object (multiple methods allowed, due to DK).
 * @param int $amount The amount to show the priceInformation with.
 * @param bool $fetch If ecom should try to download the content from the priceinfoLink.
 * @param bool $iframe Pushes the priceinfoLink into an iframe. You should preferably have $fetch false here.
 * @param bool $limitByMinMax By default, ecom only shows priceinformation based on the $amount.
 */
public function getCostOfPriceInformation(
    $paymentMethod = '',
    $amount = 0,
    $fetch = false,
    $iframe = false,
    $limitByMinMax = true
)
```

This means that you can display the priceinformation in different ways.

- Priceinfo is extracted from the legal-links block in `getPaymentMethods`.
- The first parameter allows you to send in the payment method as a string or the object directly. Do not forget to add the amount to the request as the second argument.

- The third and fourth arguments allows you to either get the url or the content of the priceInfo. With the fourth argument you can choose to contain the urls in an iframe so the modal data fits better in your store as the css and design resides remotely at Resurs Bank.
- If both third and fourth arguments are true, iframe is chosen instead of colliding the view.
- The fifth argument is by default true, so the method follows the rules of minAmount and maxAmount. This means the priceinformation will be rendered with compatible payment methods within the range of min and max. Normally, we'd be happy with this, but this can be overridden in corner cases where all methods should be shown regardless of their limits.
- To remember: All views are based on a single payment methods. If you want to show price information for denmark it is recommended to push all payment methods in the call (see below).

Showing price information for denmark

The example below renders a full "tabbed view" for denmark based on all payment methods available that has price information available.

```
$costOfPurchaseHtml = $flow->getCostOfPriceInformation($flow->getPaymentMethods(), $amount);
```

Invoice sequence handling are changed

Due to limitations in the API, invoice id's are now handled differently to prior versions. Invoice id's are set one time - if the ID is unexistent on the side of Resurs Bank. Next time aftershop are running, ECom won't force any more invoice id into the system as it this could potentially stop stores not increment id's properly. To restore the old behaviour you could use the practically undocumented flag features AFTERSHOP_RESCUE_INVOICE and AFTERSHOP_RESCUE_INCREMENT but normally, this should not be necessary.

Reachable curl handle

If any changes has to be made directly to the curl library, the support for this is opened further. Test environment is no longer a required parameter.

Units

Besides of this, pipeline tests should now work properly with each version of phpunit without backward compatible installations (as setUp() function in the units changes after PHP 7.1).

```
public function setUp(): void {
    // This is not compatible with prior versions of PHP.
}

public function __setUp() {
    // The replacement that unfortunately need to be added in each test case.
}
```

Other fixes

The full changelog could be found here: [EComPHP: ChangeLog](#).

Debugging createPayment

You can do a createPayment() without really creating it - by simply pause the final execute of bookPayment/creating the iframe. This section has however its own documentation, that you may find interesting: [Important notes and troubleshooting/exceptions \(EComPHP\)](#)

Custom shopUrl for Resurs Checkout

Normally, when creating the iframe for Resurs Checkout, not setting shopUrl will render a default URL that defines how the iframe handles the communication between the end user and the iframe. This is absolutely not necessary, but to make the features constantly open and reachable for developers EComPHP sets up this for you, in the format: **PROTOCOL://HTTP_HOST**. The protocol is based on what's returned in the web server HTTPS-variable, and if it is set, the URL might for example look like this <https://test.com> - you should here note that trailing slash and eventually the following full uri is stripped from the url, så even if you have a site laying on https://test.com/this/sub/structure the shopUrl should still be https://test.com.

You can customize this URL to point to another domain name. For example, your site should be test.org instead of test.com:

```
$rb->setShopUrl("https://test.org/");
```

Take a note on the bad formatted trailing slash. This will be stripped off in the final result.

In a near future, host validation (by resolvers, to make sure that the shopUrl is by DNS confirmed as valid) will become available.

Another thing to take note of here, is that the shopUrl should be validated even when the payload are sent into EComPHP manually. This is currently not done by default, so by setting up following code, validation will become active at payload level too:

```

$rb->setValidateCheckoutShopUrl();
// ... code ...
$myPayLoad = array('shopUrl' => 'https://test.com/badly/formed/uri/link');
// ... code ...
$rb->createPayment( /* Your data here */ );

```

Preferred payment ID's

EComPHP is designed to handle payment/order id's where such ids is not important for the store to be correct. In a normal scenario, the payment id is created by the shop, and just transferred through to EComPHP and then the order can be fulfilled. However, if this is ignored by developers or the store, EComPHP creates this by default.

setPreferredId(<string>)

This function is used when there is a prepared order id, that EComPHP should use.

getPreferredPaymentId()

This function returns a payment id. If it is pre-set by client, it will return the set id. Otherwise, it returns a randomly generated id, based on the current timestamp and a random number (to avoid conflicts between many end users).

setCountry

Setting up a default country by setCountry might help a bit with things like default unitMeasure-data. While using addOrderLine, the unitMeasure is set to be "st" (styck). However, setting up the country with this function allows you to pass \$unitMeasure empty. Doing this, will change (try) the values of unitMeasure to a proper measure for each country.

Country value	Default unitMeasure
ResursCountry:: COUNTRY_DK	st
ResursCountry:: COUNTRY_NO	st
ResursCountry:: COUNTRY_FI	kpl
ResursCountry:: COUNTRY_SE	st (default)

Special payment functions and aftershop features

This section is mostly made for the aftershop functions. You might also take a look at the bottom section for [EComPHP: DEBIT, CREDIT, ANNUL \[afterShopFlow\]](#), where there are some functions listed for which you can find out which status an order is set in.

getAnnuityPriceByDuration

EComPHP has special features that should make it easy to show how much a customer needs to pay monthly (usually shown at single-product pageviews). This method is called getAnnuityPriceByDuration and is in details described at the section [Calculations with getAnnuityFactors/part payments \(EComPHP\)](#).

getPaymentDiffByStatus

This is a master renderer and has replaced the prior method name **getPaymentSpecByStatus**, which sorts out what happened to each paymentDiff in the getPayment service API. It will list all authorized, debited, credited and annulled orderrows separately. The newer release not only give you statuses from the paymentDiff (of what happened to the order). It also renders a per-product-row-table where it shows what's left in the order referring to debiting, annulling and crediting. Together with **getPaymentDiffByAbility** it has a high value when it comes to partially crediting and annulling orders which EComPHP hasn't entirely supported before. This comes to a reality in v1.3.23+ (1.0.48/1.1.48) for prior setups.

It looks like this:

```

$orderLinesByStatus = array(
    'AUTHORIZE' => array(),
    'DEBIT' => array(),
    'CREDIT' => array(),
    'ANNUL' => array(),
);

```

The formatted table itself can look like the table below, where the new values ANNULLABLE, DEBITABLE and CREDITABLE keys shows up which quantity that is left in for each item row. So, if you plan to credit PR01, this table tells you that there are 50 more in the quantity that you can credit, etc. There's also a bunch of new features for which ECom can handle paymentdiffs and compare items. We primarily use [array_intersect](#) to compare valid orderrows those days instead of the prior "removeFromArray" that is from v1.3.23 completely removed from the source.

```

Array
(
    [0] => Array
    (
        [artNo] => PR01
        [description] => PR01
        [unitMeasure] => st
        [unitAmountWithoutVat] => 37.00000
        [vatPct] => 25.00000
        [AUTHORIZE] => 100.00000
        [DEBIT] => 50.00000
        [CREDIT] => 0
        [ANNUL] => 50.00000
        [ANNULLABLE] => 0
        [DEBITABLE] => 0
        [CREDITABLE] => 50
    )

    [1] => Array
    (
        [artNo] => PR02
        [description] => PR02
        [unitMeasure] => st
        [unitAmountWithoutVat] => 57.00000
        [vatPct] => 25.00000
        [AUTHORIZE] => 100.00000
        [DEBIT] => 0
        [CREDIT] => 0
        [ANNUL] => 0
        [ANNULLABLE] => 100
        [DEBITABLE] => 100
        [CREDITABLE] => 0
    )

    [2] => Array
    (
        [artNo] => PR03
        [description] => PR03
        [unitMeasure] => st
        [unitAmountWithoutVat] => 55.00000
        [vatPct] => 25.00000
        [AUTHORIZE] => 100.00000
        [DEBIT] => 0
        [CREDIT] => 0
        [ANNUL] => 0
        [ANNULLABLE] => 100
        [DEBITABLE] => 100
        [CREDITABLE] => 0
    )

    [3] => Array
    (
        [artNo] => PR04
        [description] => PR04
        [unitMeasure] => st
        [unitAmountWithoutVat] => 68.00000
        [vatPct] => 25.00000
        [AUTHORIZE] => 100.00000
        [DEBIT] => 0
        [CREDIT] => 0
        [ANNUL] => 0
        [ANNULLABLE] => 100
        [DEBITABLE] => 100
        [CREDITABLE] => 0
    )
)

```

```
)  
})
```

getPaymentSpecCount

This function was built primarily for unittesting. It uses getPaymentSpecByStatus() and instead of returning the order rows, returns each diff-status-object with the total count of rows for each status:

```
$orderLinesByStatus = array(  
    'AUTHORIZE' => count,  
    'DEBIT' => count,  
    'CREDIT' => count,  
    'ANNUL' => count,  
) ;
```

sanitizeAfterShopSpec

Sanitizes a payment spec from a payment id or a prepared getPayment object and return filtered depending on the requested aftershop type

User and client

There are several functions in EComPHP that is used to identify users and clients. This is some of them, listed.

EComPHP Client identification

When communicating with Resurs Bank webservices and rest, you can tell EComPHP to identify itself with a customized USER_AGENT-header. This opens for the possibility to identify your plugin name, the web store name/version and any other variable you need to use to simplify error checking and so.

```
$rb->setUserAgent('MyPlugin-2.1.77 Magento-XXX.YYY');
```

Client Identification - SubFeature

If you have great needs of also transmitting the end user web-browser with this USER-AGENT-header, you can also activate another feature by using setPushCustomerUserAgent(true). In this case, EComPHP will also add the end user HTTP_USER_AGENT into the user-agent string, as a compressed string, base64-encoded.

User and application identification

This function is mostly used by AfterShop features in EComPHP. If there are any needs of identifying who annulled, credited or debited a payment setLoggedInUser() can be used for setting more data than just "EComPHP -versionNumber-". The data can be seen, amongst others, in Resurs Payment Admin. The following example will render what's shown in the screen capture below (except for the realClientName parts as this has been built in after the screenshots, which by the way also needs to be renewed to match MP).

Note: setRealClientName is used to assign a name in the aftershop parts, which identifies an application if the aftershop process has been automated.

```
$rb->setLoggedInUser('myAdminUserName');  
$rb->setRealClientName('myApplicationName');  
$rb->annulPayment($paymentId);
```

Orderspecifikation [-]

= Auktoriserat = Annulerat = Debiterat = Krediterat

Artikelnummer	Beskrivning	Styckepris exkl. moms	% moms	Summa	Varav moms	Auktoriserat	Annulerat	Debiterat	Krediterat
Art 1705	Beskrivning 4037	1905,00	25	2381,25	476,25		1 st		
Art 1589	Beskrivning 3033	1907,00	25	2383,75	476,75		1 st		
Art 1045	Beskrivning 2109	1014,00	25	1267,50	253,50		1 st		
Art 2029	Beskrivning 3944	1190,00	25	1487,50	297,50		1 st		
Art 1318	Beskrivning 3484	1904,00	25	2380,00	476,00		1 st		
Art 1892	Beskrivning 2753	1860,00	25	2325,00	465,00		1 st		
Art 1342	Beskrivning 2161	1993,00	25	2491,25	498,25		1 st		
Art 1857	Beskrivning 2904	1179,00	25	1473,75	294,75		1 st		

Dokument [+]

Metadata [+]

Betalningshistorik [-]

Auktorisation	2017-10-09 10:56	Skapad av: SHOP_FLOW	Totalt belopp: 16190,00		Total moms: 3238,00	
Artikel nummer	Beskrivning	Belopp	Kvantitet	Moms	Totalt belopp	Total moms
Art 1705	Beskrivning 4037	1905,00	1 st	25%	2381,25	476,25
Art 1589	Beskrivning 3033	1907,00	1 st	25%	2383,75	476,75
Art 1045	Beskrivning 2109	1014,00	1 st	25%	1267,50	253,50
Art 2029	Beskrivning 3944	1190,00	1 st	25%	1487,50	297,50
Art 1318	Beskrivning 3484	1904,00	1 st	25%	2380,00	476,00
Art 1892	Beskrivning 2753	1860,00	1 st	25%	2325,00	465,00
Art 1342	Beskrivning 2161	1993,00	1 st	25%	2491,25	498,25
Art 1857	Beskrivning 2904	1179,00	1 st	25%	1473,75	294,75

Annulering	2017-10-09 10:56	Skapad av: EComPHP_010123/myAdminUserName	Totalt belopp: 16190,00		Total moms: 3238,00	
Artikel nummer	Beskrivning	Belopp	Kvantitet	Moms	Totalt belopp	Total moms
Art 1705	Beskrivning 4037	1905,00	1 st	25%	2381,25	476,25
Art 1589	Beskrivning 3033	1907,00	1 st	25%	2383,75	476,75
Art 1045	Beskrivning 2109	1014,00	1 st	25%	1267,50	253,50
Art 2029	Beskrivning 3944	1190,00	1 st	25%	1487,50	297,50

Resurs Bank payment order Statuses, how do I handle them?

getOrderStatusByPayment

How you configure your store is very individual and are often also dependent on what your store can handle. However, there is a suggested "best practice" you could follow when setting up how to handle local order statusen during for example receiving callbacks or updating orders depending on the status of the order at Resurs Bank side. There is fragment of code in EComPHP that suggests how to update the orders, via **getOrderStatusByPayment**. Here's a small example (taken out of its context) on how to use it.

```
$storeOrderId = "1337";
$paymentIdOrPaymentObject = "1337";
$byCallbackEvent = RESURS_CALLBACK_TYPES::CALLBACK_TYPE_AUTOMATIC_FRAUD_CONTROL;
$callbackResult = "THAWED";
// Callback AUTOMATIC_FRAUD_CONTROL arrives here
$suggestedStatus = $ecom->getOrderStatusByPayment($paymentIdOrPaymentObject, $byCallbackEvent, $callbackResult);
```

```

switch ($suggestedStatus) {
    case RESURS_PAYMENT_STATUS_RETURNCODES::PAYMENT_PROCESSING:
        setStoreLocalStatus($storeOrderId, "processing");
    default:
        setStoreLocalStatus($storeOrderId, "on-hold");
}

```

To use this without the callback control, you can just run this:

```
$suggestedStatus = $ecom->getOrderStatusByPayment($paymentIdOrPaymentObject);
```

In this case, the control will process lesser analyzing on the current order. However, it is not always interesting to run this control with the callback control. The returned value is based on [RESURS_PAYMENT_STATUS_RETURNCODES](#).

RESURS_PAYMENT_STATUS_RETURNCODES and how they are used

The calculations below are based on the [advices from the aftershop service API here](#) (scroll to the bottom). From around v1.3.16 the behaviour of the return codes has changed to bitwise data, as SWISH payment method practically introduced a new kind of status: Instant debits. The means of this is that when you get the status DEBITED from a getPayment, you won't be able to determine if the finalization has been instant or if someone manually fired a finalization. EComPHP tries to find out this on its own, by comparing the payment with the payment method.

If the payment method is based on SWISH (this is configurable) or other "instant payment methods" (INTERNET or "direkt banköverföring" is also based on this) EComPHP must have the ability to set to statuses in the same time. By using bitwise masking in this setup you can either use the values as before - separately - or check for a combined set of data. For example, if this feature is enabled with SWISH, the return code will be 4 & 32 (36) rather than just 4 according to the table below.

Status	Integer value	Description	Calculations (AND)
NOT_IN_USE	0	When the analyze of the current payment could not be define anything	Nothing will happen
PAYMENT_PENDING	1	Invoked on BOOKED and AUTOMATIC_FRAUD_CONTROL <ul style="list-style-type: none"> • BOOKED: getPayment(), frozen is true • AUTOMATIC_FRAUD_CONTROL, not thawed (FROZEN) 	
PAYMENT_PROCESSING	2	Invoked on AUTOMATIC_FRAUD_CONTROL, BOOKED and UNFREEZE <ul style="list-style-type: none"> • BOOKED: getPayment(), frozen is false • AUTOMATIC_FRAUD_CONTROL: Trusting THAWED • UNFREEZE: Trusting the event 	
PAYMENT_COMPLETED	4	Invoked on FINALIZATION and UPDATE <ul style="list-style-type: none"> • FINALIZATION: Trusting the event • UPDATE: getPayment() - See calculations 	<ul style="list-style-type: none"> • ! canDebit(\$paymentData) • getIsDebited(\$paymentData) • \$resursTotalAmount > 0
PAYMENT_ANNULLED	8	Invoked on ANNUL and UPDATE <ul style="list-style-type: none"> • ANNUL: getPayment() - Trusting the event • UPDATE: getPayment() - See calculations 	<ul style="list-style-type: none"> • getIsAnnulled(\$paymentData) • ! \$this->getIsCredited(\$paymentData) • \$resursTotalAmount == 0
PAYMENT_CREDITED	16	Invoked on UPDATE and ANNUL <ul style="list-style-type: none"> • Using getPayment() - See calculations 	<ul style="list-style-type: none"> • getIsCredited(\$paymentData) <ul style="list-style-type: none"> ◦ \$resursTotalAmount == 0
PAYMENT_AUTOMATICALLY_DEBITED	32	Most often invoked on FINALIZATION <ul style="list-style-type: none"> • Using getPayment() and getPaymentMethods() 	Based on PAYMENT_COMPLETED but with a twist.
PAYMENT_MANUAL_INSPECTION	64	-	
PAYMENT_STATUS_COULD_NOT_BE_SET	128	When something goes wrong. This was changed from 0 for a while ago since 0 is also considered a value in the bitwise-verse.	

Callbacks

When callbacks are handled, you can either code the digest validation yourself by, something like this (examples is from the WooCommerce plugin):

```

$currentSalt = get_transient( 'resurs_bank_digest_salt' );
if ( $event_type == 'AUTOMATIC_FRAUD_CONTROL' ) {
    $check_digest = $request['paymentId'] . $request['result'] . $currentSalt;
} else {
    $check_digest = $request['paymentId'] . $currentSalt;
}
$check_digest = sha1( $check_digest );
$check_digest = strtoupper( $check_digest );
$resursPaymentData = null;
try {
    $resursPaymentData = $this->flow->getPayment( $request['paymentId'] );
} catch (\Exception $ignorePaymentExceptions) {}
if ( $request['digest'] === $check_digest ) {
    // Do whatever you need with the callback, on correct matches
}

```

Or you can use EComPHP internal validation (available from 1.0.33/1.1.33/1.2.6/1.3.6):

```

$currentSalt = get_transient( 'resurs_bank_digest_salt' );
if ( !$this->flow->getValidatedCallbackDigest(isset($request['paymentId']) ? $request['paymentId'] : null,
$currentSalt, isset($request['digest']) ? $request['digest'] : null, isset($request['result']) ? $request
['result'] : null) ) {
    // Digest mismatch, so stop handle the callback from here
    die;
}

```

Multiple stores

If you have multiple stores in a single merchant, you can set up a deal with Resurs Bank to use storeId's. In this case, you can add an extra row in the payload sent with createPayment: **setStoreId(<integer>)**.

Extended features

This section contains a list of features that either has been extended or is very specific for EComPHP as helpers.

getCostOfPurchase

The getCostOfPurchase-function is based on Resurs Bank webservice getCostOfPurchaseHtml. However, instead of just returning the html-code generated by Resurs Bank, the internal function also returns a properly set html-body and - if requested - a linked CSS that refers to the page content, so it could be easily designed with a standard look.

Parameter	Type	Description
paymentMethod	string	The basic set up, required to generate the correct data
amount	int	The base amount for which Resurs Bank will build the costOfPurchase-page
returnBody	boolean	Setting this to true, will add the proper <html>-tags in the returned html code
callCss	string	URL to the CSS layout
hrefTarget	string	Sets up, for each href element in the html body, a target="_blank" (default) so that each click in the information box opens a new window rather than just redirects them. To change the _blank-behaviour to something else, the string can just simple be replaced to a custom value.

setCostOfPurchaseHtmlBefore and setCostOfPurchaseHtmlAfter

If you have greater needs to add data into the getCostOfPurchase, that is prepended or appended to the returned body you can use those two functions to add additional data to the returned html body.

External extended URL validation

This feature is very out of the box of EComPHP and uses a third party API, that do not belong to Resurs Bank. In some cases, there have been problems with site reachability - for example, when callbacks are running there is a web hosting company that blocks the connectivity to your store. Or, another one, where you use internal links that is normally not reachable at all for our callback services. For such cases, you can run this function to find out whether your site is reachable or not.

Usage example (partially taken from our unittest suite):

```
$testUrl = "https://test.resurs.com/signdummy/index.php?isCallback=1&event=ANNULMENT&digest={digest}&paymentId={paymentId}&lastReg=171009083933";
$validateResponse = $rb->setValidateExternalCallbackUrl( $testUrl );
if ( $Reachable !== ResursCallbackReachability::IS_FULLY_REACHABLE ) {
    // Our site is not fully reachable on this response
}
```

The API used for this function supports **IPv6**.

Invoice sequence numbering

Using our aftershop flow means you can handle all of your orders on your ecommerce-site instead of via our merchant portals, like payment admin. The first time you run through those functions - you can read about them here [EComPHP: DEBIT, CREDIT, ANNUL \[afterShopFlow\]](#) - there will be a need for an invoice number on Resurs side. This is automatically set up by EComPHP if not already set. As of v1.0.27/1.1.27 (and 1.2.0) you can reset the invoice sequence (it will be nullified), with the function call **resetInvoiceNumber()**. If this is being made by mistake, the prior invoice sequence numbering might be restored again with the help from **getNextInvoiceNumberByDebits()** - which is a function that scans through a number of already debited payments, from which it uses the last invoice number it can find and restores the invoice sequence number with this value (+1).

Undocumented features

setCustomerIpProxy

ShopFlow cloned features (deprecated)

The methods in this section is "long time deprecated". They have a history in our deprecated shopflow that, in startPaymentSession, got most of the form data from Resurs Bank, neceassy to create a proper order. To make things easier in early versions of our WooCommerce-plugin, this part of EComPHP was used to simlate the behaviour from the flow. EComPHP is here set up to return customer field data, required by end users to complete orders. It can also return default regexes for which developers can validate proper customer fields like phone numbers, e-mail, etc. In future version it is completely up to developers to make this right.



Class changes

As of newer releases of EComPHP, the deprecated form controller has been moved to its own class.

This documentation section is incomplete.

setFormTemplateRules

getRegEx

canHideFormField

getTemplateFieldsByMethodType

getTemplateFieldsByMethod

getFormFieldsByMethod

Prefilled customer data (EComPHP)

To pre-set customer data in Resurs Checkout from for example logged in user data you could do something like below. This is an example grabbed from the WooCommerce plugin. As described below, to give effect to another billing address for RCO you push data into the delivery address the same way you normally do in the simplified flow. Also remember that if you want to add other data like phone, e-mail (and government id, that is just an empty string in the below example) you need to set this data via setCustomer() in EComPHP.

*Currently as the feature is quite new, it has to be enabled with the ECom-flag **KEEP_RCO_DELIVERY**. See below. There is also a **KEEP_RCO_BILLING** available in case something else should be acutalized in the future. However, the plan is rather to make those two flags unused once reaching production properly.*

```
private function getDataFromCustomerObject($billingObject, $key) {
    $return = '';
    if (isset($billingObject[$key])) {
        $return = $billingObject[$key];
    }
    return $return;
}

/// Another function goes here.

if (is_user_logged_in()) {
    $this->flow->setFlag('KEEP_RCO_DELIVERY', true);
    $loggedInCustomer = new WC_Customer(get_current_user_id());
    $loggedInBilling = $loggedInCustomer->get_billing();
    $this->flow->setCustomer(
        '',
        '',
        $this->getDataFromCustomerObject($loggedInBilling, 'phone'),
        $this->getDataFromCustomerObject($loggedInBilling, 'email'),
        'NATURAL',
        ''
    );
    $this->flow->setDeliveryAddress(
        $this->getDataFromCustomerObject(
            $loggedInBilling,
            'first_name'
        ) . ' ' . $this->getDataFromCustomerObject(
            $loggedInBilling, 'last_name'
        ),
        $this->getDataFromCustomerObject($loggedInBilling, 'first_name'),
        $this->getDataFromCustomerObject($loggedInBilling, 'last_name'),
        $this->getDataFromCustomerObject($loggedInBilling, 'address_1'),
        $this->getDataFromCustomerObject($loggedInBilling, 'address_2'),
        $this->getDataFromCustomerObject($loggedInBilling, 'city'),
        $this->getDataFromCustomerObject($loggedInBilling, 'postcode'),
        $this->getDataFromCustomerObject($loggedInBilling, 'country')
    );
}
```

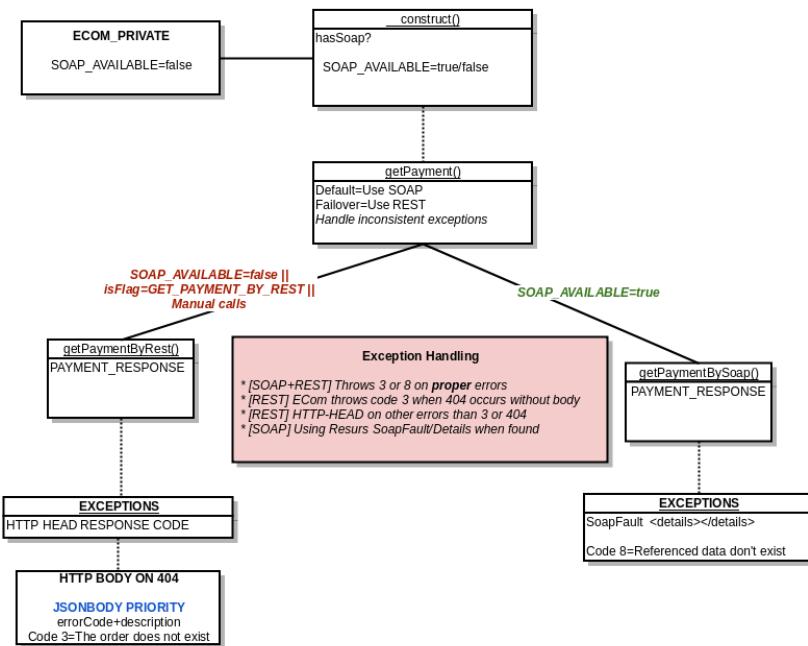
getPayment-magics

When using our API's you'll soon realize that the checkout/hosted-flow API contains most of the functions you actually need. There are however exceptions when you have to fall over to SOAP-calls - getPaymentMethods, and such methods only exists in the webservices API (SOAP).

getPayment however, exists in both environments, but the two API's works slightly different.

- getPayment in the SOAP-call and the REST-API gives you the same information when orders exists at Resurs Bank
- getPayment, on failures, in the SOAP-call will most likely throw a soapfault with extended information about the error (in a details block). When the payment does not exist, the error code is **8**.
- getPayment, on failures, in the REST-API does not throw a soapfault (of course), but an HTTP HEAD error **404** (not found) or internal code **3** which contains an extended body with the "more correct error response" (which also might confuse exception handling)

The fact that the rest functions actually returns a http-404 might confuse a bit, as this error type is actually used for more than just unexistent payments. If you build tests based on getPayment with rest, you need to consider that there's also an extended json-based body, when you're using the call and gets errors.



Calculations with getAnnuityFactors/part payments (EComPHP)

EComPHP has internal functions to calculate annuity factors (or "part pay from" on for example single product page views where customers wants to know how much they need to pay per month). The function called by EComPHP is basically getAnnuityFactors, but instead of letting intergrators/developers calculate the factors by themselves, EComPHP does this for you. The primary function to do this is getAnnuityPriceByDuration, where you push in the total price of a product, the factors you want to use when you calculate the amount, and finally the duration time of how long the payment should be "alive" (12 months, 24 months and so on).

```
$payFrom = $flow->getAnnuityPriceByDuration(10000,  
    'PARTPAYMENT',  
    12  
) ;
```

In this example we will get 833 SEK out of the response based on our chosen payment method.

No fees are added in this question

No extra set up fees are calculated in this call. If you want complete information about monthly payments including fees, the alternative to use is getCostOfPurchaseHtml. Unfortunately, that function is not an automated function that gives values separately. It generates a complete html table (exactly as the method name are called).

Further description of what getAnnuityPriceByDuration do in its calls

As you can see below, you're quite free of what you send into the question. You can either send the getAnnuityFactors-object directly into the method and save some time with it (if you for example have it cached). If you don't have this and want as live information as possible, the \$paymentMethodIdOrFactorObject can be used to send for example the payment method name "PARTPAYMENT". Doing this, will first resolve which factors that belongs to the payment method and THEN after this start the calculation.

```
/**  
 * Get annuity factor rounded sum by the total price  
 *  
 * @param $totalAmount  
 * @param $paymentMethodIdOrFactorObject  
 * @param $duration  
 *  
 * @return float  
 * @since 1.1.24  
 * @throws \Exception  
 */  
public function getAnnuityPriceByDuration($totalAmount, $paymentMethodIdOrFactorObject, $duration)  
{  
    $return = 0;  
    $durationFactor = $this->getAnnuityFactorByDuration($paymentMethodIdOrFactorObject, $duration);  
    if ($durationFactor > 0) {  
        $return = round($durationFactor * $totalAmount);  
    }  
  
    return $return;  
}  
  
/**  
 * Get annuity factor by duration  
 *  
 * @param $paymentMethodIdOrFactorObject  
 * @param $duration  
 *  
 * @return float  
 * @throws \Exception  
 * @since 1.1.24  
 */  
public function getAnnuityFactorByDuration($paymentMethodIdOrFactorObject, $duration)  
{  
    $returnFactor = 0;  
    $factorObject = $paymentMethodIdOrFactorObject;  
    if (is_string($paymentMethodIdOrFactorObject) && !empty($paymentMethodIdOrFactorObject)) {
```

```
    $factorObject = $this->getAnnuityFactors($paymentMethodIdOrFactorObject);
}
if (is_array($factorObject)) {
    foreach ($factorObject as $factorObjectData) {
        if ($factorObjectData->duration == $duration && isset($factorObjectData->factor)) {
            return (float)$factorObjectData->factor;
        }
    }
}

return $returnFactor;
}
```

Bitwise features (EComPHP)

At some places in EComPHP there are constants based on the integers 1, 2, 4, 8, 16, etc - rather than 1, 2, 3, 4. By means, you can set up one value that represents many events. Below follows a list with examples where we practice active bitmasking in EComPHP.

Table of contents

- General usage of bitmasking
 - Principles
- RESURS_PAYMENT_STATUS_RETURNCODES
 - RESURS_CALLBACK_TYPES
 - Other classes based on bitmasking

General usage of bitmasking

If we use **RESURS_PAYMENT_STATUS_RETURNCODES** below in this description, a payment can - regardless of what Resurs are returning in the `getPayment()`-statement - in the below example have a maximum value of 63.

If this happens, something of course went terribly wrong, but as an example it is just fine. The value 63 indicates that the order is *pending*, *processing*, *completed*, *annulled*, *credited* and *automatically debited* at the same time. Another example, if the code is 12, we know that the order is both annulled and finalized, and so on. By using bitvalues, instead of static constants, we can add as many of those flags as we need. If the value is 27, the status will become *Credited, Annulled, Processing and Pending* ($16+8+2+1$) and so on.

The only thing we need to think of, is to use each bit as the flag that we want to extract:

Bit	Value	ReturnCode
1	1	Pending
2	2	Processing
3	4	Completed
4	8	Annulled
5	16	Credited
6	32	Automatically debited
7	64	Another flag
8	128	Yet another flag
...	Etc	Add as much as you want

Principles

This method normally allows developers to store at least 8 different flags as one byte in a file, so even if there are 8 different settings in a configuration, the file will just be 1 byte big and the single character ascii code can be anything between 0-255.

If the flags are more than 8 bits, the value will require 2 byte (16 bit = 1024 - 2 bytes, 24 bit = 2048 - 3 bytes, and so on).

RESURS_PAYMENT_STATUS_RETURNCODES

RESURS_PAYMENT_STATUS_RETURNCODES is refactored as of october 2018. Instead of returning a static constant during payment status checks, it returns a value that represents a set of flags for a payment. In this case the flags only covers payments that is actually finalized immediately after the order has been created. You can read more about the usage [here](#). In short a payment can have more than one status code due to the payment behaviour. If an order is completed only, after a set of actions from the merchant, the status usually will be 4.

```
abstract class RESURS_PAYMENT_STATUS_RETURNCODES
{
    const PAYMENT_STATUS_COULD_NOT_BE_SET = 0;      // Waiting for callback or frozen
    const PAYMENT_PENDING = 1;           // Waiting for callback or frozen
    const PAYMENT_PROCESSING = 2;        // Booked, waiting for next action
    const PAYMENT_COMPLETED = 4;         // Fully finalized (debited)
```

```

const PAYMENT_ANNULLED = 8;      // Fully annulled
const PAYMENT_CREDITED = 16;     // Fully credited
const PAYMENT_AUTOMATICALLY_DEBITED = 32;    // Fully credited
}

```

RESURS_CALLBACK_TYPES

RESURS_CALLBACK_TYPES is normally not necessary to bitmask as you might want to use different urls for different callbacks, when registering them. It might get handy when you'd like to register many callbacks in one call. As this part of EComPHP has been used for long time, there is a slight risk that compatibility breaks on the road, so for registering callbacks this is not fully implemented. However, for unregistering the callback events, it is since there are no need for adding urls to that method call.

BitMask

```

abstract class RESURS_CALLBACK_TYPES
{
    const NOT_SET = 0;
    const UNFREEZE = 1;
    const ANNULMENT = 2;
    const AUTOMATIC_FRAUD_CONTROL = 4;
    const FINALIZATION = 8;
    const TEST = 16;
    const UPDATE = 32;
    const BOOKED = 64;
}

```

For those constants, you might apply bitmasking features. Bitmasking is still experimental but the example above shows constants for how to use callbacks in this way. In this particular one, you can unregister multiple callbacks with only one touch, instead of calling the unregisterEventCallback multiple times.

The following example will unregister FINALIZATION, AUTOMATIC_FRAUD_CONTROL and TEST as callbacks. The second parameter is set to true and will indicate, to Ecom, that the command covers multiple callbacks:

```

$this->TEST->ECOM->unregisterEventCallback(RESURS_CALLBACK_TYPES::FINALIZATION | RESURS_CALLBACK_TYPES::AUTOMATIC_FRAUD_CONTROL | RESURS_CALLBACK_TYPES::TEST, true);

```

Another way to do the exact same request is to use the value directly. However, this is not recommended, since there is always a risk that changes is made in core systems.

```

$this->TEST->ECOM->unregisterEventCallback(28, true);

```

Other classes based on bitmasking

RESURS_AFTERSHOP_RENDER_TYPES is prepared for the use of bitmasking but currently not entirely tested. The purpose is to extract payment specs from an order with specific types. For example, to extract rows from an order that has been annulled or credited (**(RESURS_AFTERSHOP_RENDER_TYPES::CREDIT | RESURS_AFTERSHOP_RENDER_TYPES::ANNUL)**) can be used. To extract rows that has been debited and credited, you can in the same way use (**(RESURS_AFTERSHOP_RENDER_TYPES::CREDIT | RESURS_AFTERSHOP_RENDER_TYPES::FINALIZE)**).

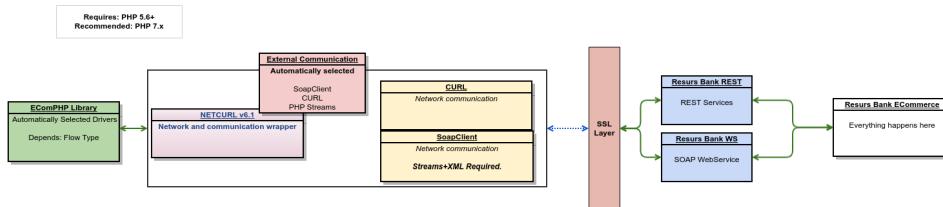
EComPHP Visualization and backstory

Table of contents

[[Communication drivers](#)] [[Overlapping flows](#)] [[getPayment Magics](#)] [[Aftershop Flow Magics](#)] [[Was and is - Backstory](#)]

Communication drivers

This image shows what EComPHP actually is in a visual style. Observe that SoapClient has an entry - "internal curl" - in this scenario. The reason is that SoapClient is internally using curl to communicate with Resurs Bank Webservices (SOAP).

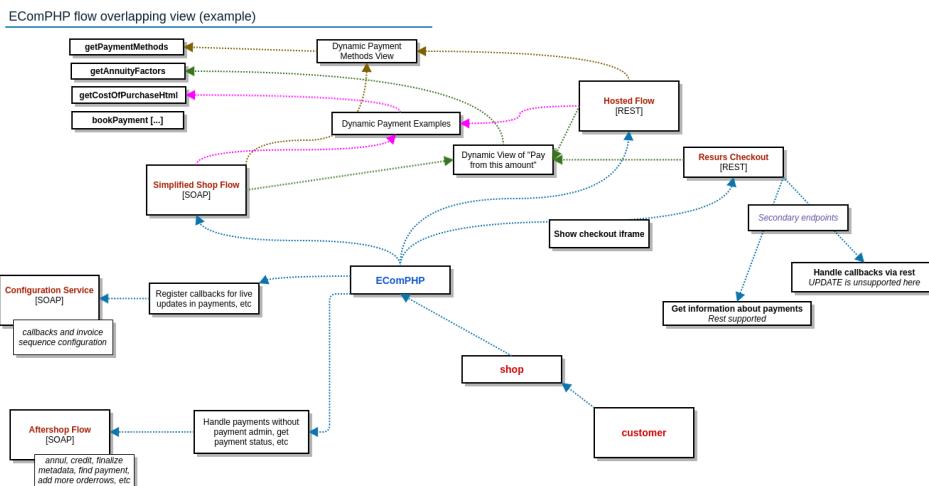


Overlapping flows

The below view is a shortened (as EComPHP do much more than this) overview on how the payment flows may overlap each other, when EComPHP is fully integrated into a system.

As you can see here, there are mixed communication environments:

- The simplified flow and hosted flow allows checkouts, but parts in hosted flow can use the simplified system. For example: If the checkout that uses the hosted flow, needs to show payment methods dynamically, the simplified flow is used to fetch and list the payment methods
- If your store allows separate "pay from NNN SEK" for each item in the store, the simplified flow is used, even if you have Resurs Checkout or Hosted flow configured
- If your store runs the checkout with rest-only in the shop, but you want to handle all payments from the shops internal order administration, SOAP and aftershop services are still required (unless you find it more comfortable to use Resurs Payment admin)
- Callbacks has to be configured somewhere, if you want to handle events that occurs in payment admin. It could be done either via [rest \(checkout\)](#) or [SOAP \(fully supported as the rest does not support UPDATE\)](#)



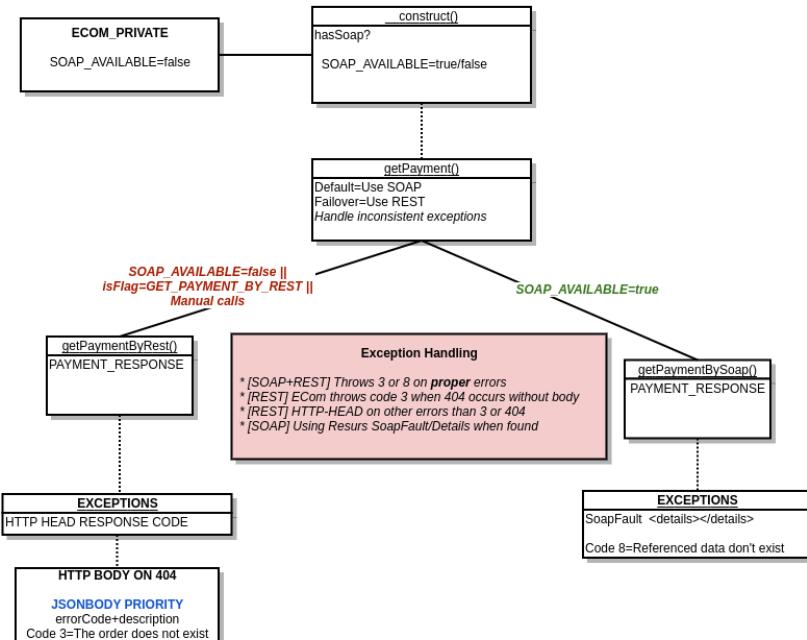
getPayment Magics

When using our API's you'll soon realize that the checkout/hosted-flow API contains most of the functions you actually need. There are however exceptions when you have to fall over to SOAP-calls - `getPaymentMethods`, and such methods only exists in the webservices API (SOAP).

`getPayment` however, exists in both environments, but the two API's works slightly different.

- `getPayment` in the SOAP-call and the REST-API gives you the same information when orders exists at Resurs Bank
- `getPayment`, on failures, in the SOAP-call will most likely throw a soapfault with extended information about the error (in a details block). When the payment does not exist, the error code is 8.
- `getPayment`, on failures, in the REST-API does not throw a soapfault (of course), but an HTTP HEAD error 404 (not found) which contains an extended body with the "more correct error response" (which also might confuse exception handling)

The fact that the rest functions actually returns a http-404 might confuse a bit, as this error type is actually used for more than just unexistent payments. If you build tests based on `getPayment` with rest, you need to consider that there's also an extended json-based body, when you're using the call and gets errors.



Aftershop Flow Magics

Coming soon.

Was and is - Backstory

EComPHP was, historical, built around a number of stub files generated by wsdl2phpgenerator as this was a recommended way of simplifying communication with Resurs Bank Ecommerce. Using the stubs directly sometimes was a bit tricky to get along with, so the ending result was EComPHP, that handled all basic communication for the developer. There were several methods built in, that EComPHP handled, and the goal was to simplify the flows that much, so only one method call was needed to initiate a payment - and even if the shop flows changed, all initiations should be identical since some fields in the payload sometimes mismatched between the flows. EComPHP fixed this too, and still are.

Time passed and Resurs Bank different shop flows were extended with the hosted flow and Resurs Checkout which, in difference to the simplified flow, uses rest calls.



Community support

If something does not work properly in EComPHP, feel free to [join the community via bitbucket](#), or contact us at onboarding@resurs.se. EComPHP is updated continuously to improve functionality.

Important notes and troubleshooting/exceptions (EComPHP)

- Payment methods in simplified shopflow vs Resurs Checkout
- Why is NOT_ALLOWED_IN_PAYMENT_STATE happening?
- Why is "-" added to customerId in metadata?
- The SOAP-calls are not cached and therefore slow
- SSL Issues
- ECommerce throws/returns errors during createPayment
- Exceptions

Payment methods in simplified shopflow vs Resurs Checkout

Historically (for ecom 1.3 at least), simplified shopflow has not always supported payment providers. To not break standard compatibility by displaying unsupported methods in a checkout, EComPHP chose to hide them by default. Today, this is different but the getPaymentMethods-method called from ECom is still honoring PSP methods. This happens since payment providers supports government-id-less submission forms, which Resurs internals require. To activate full support for all payment methods (requires that you can handle the lack of government id), you can make this kind of call to EComPHP:

```
$connection->setSimplifiedPsp(true);
```

Why is NOT_ALLOWED_IN_PAYMENT_STATE happening?

Normally, this error is considered a "cornercase trap" which can happen if you handle your payment integration wrong based on payment methods that does not require signing and customers gets their payments FROZEN on checkout. bookSignedPayment can not usually be used on orders in that "payment state" and therefore the soap will refuse the action. If you integrate your system, if nothing else works, you can do a getPayment before the bookSignedPayment, since this SOAP-call only is needed when the payment is not yet fully created at Resurs Bank. So if you are using getPayment and gets a response from the API that gives you an existing order - you don't have to book any signed payment as it is already present.

Why is "-" added to customerId in metadata?

See [ECOMPHP-334](#). Swedish description below.

ECom autogenererar ett kundnummer under aftershop-hanteringen (se nedan). Frågan har ställts varför det är så och av allt att döma är den enda skillnaden just nu att kundnummer är helt tomt när man inte sätter det, jämfört med när man sätter ett "-" i fakturan (se dumpt). Antingen har detta varit en fix för att motverka en tidig bugg med fakturorna, men det förtäljer inte historien då metoden i princip inte har detta dokumenterat.

Vad som möjligtvis kan ha varit anledningen till att kundnummer sätts så tidigt i processen kan ha att göra med att det inte går att sätta kundnummer senare, utan måste göras direkt. Detta förvarande är i skrivande stund otestat dock. Det är bara fakturautseendet i sig som har genomgått test. Men eftersom det är metadata är det mest troligt att det sätts tidigt för att inte påverka prestandamässigt vid själva finaliseringen, genom att skicka extra apianrop just när en faktura skall debiteras. Dessutom kan det vara så att metadata-apiet inte fungerar under tiden som finalizePayment går iväg och därmed riskerar finaliseringen att inte heller fungera om felet inte fångas ordentligt.

```
/**  
 * Split function for aftershop: This was included in each of the deprecated function instead of running from  
 * a  
 * central place  
 *  
 * @param $paymentId  
 *  
 * @return bool  
 */  
private function aftershopPrepareMetaData($paymentId)  
{  
    try {  
        if (empty($this->customerId)) {  
            $this->customerId = "-";  
        }  
        $this->addMetaData($paymentId, "CustomerId", $this->customerId);  
    } catch (\Exception $metaResponseException) {  
    }  
  
    return true;  
}
```

The SOAP-calls are not cached and therefore slow

You may be able to speed up things in ECom by enabling wsdl caching, just remember that things are currently not built for handling this properly. The history goes back to an unstable SoapClient where the results of the soap-requests could act very strange when the WSDL was cached. However, caching wsdl may give the opportunity to save a few seconds as the SoapClient tend to ask for the wsdl during each request otherwise. Just remember, if you follow the below advise you're kind of on your own, as the current module is built on trying to not do anything wrong - without caching the wsdl.

If you are a developer, make sure you have the latest release of EcomPHP and netcurl v6 and add something like this.

netcurl cache_wsdl

```
// For versions above 1.3.40, use:  
$this->ECOM->setWsdlCache(true);  
  
// For older versions of the plugin, use:  
$CH = $this->TEST->ECOM->getCurlHandle();  
$CH->setCurlOpt(['cache_wsdl' => 3]);  
$this->TEST->ECOM->setCurlHandle($CH);
```

The behaviour of the above command is not entirely natural. Normally, from netcurl 6.0.25, an error occurred when trying to push options into something that is used by curl. From version 6.0.26 and 6.1.0 this is however allowed to do, even if the outcome of it MAY destroy something else in the ecom-core.

Note: How this is done in netcurl-6.1 is still not documented. However, having direct access to the each netcurl core (preferably NetWrapper as this is auto selective on http communication), allows - regarding to the test suite - something like this:

SoapClientWrapper

```
(new SoapClientWrapper($this->no_wsdl))  
    ->setWsdlCache(WSDL_CACHE_DISK)  
    ->setAuthentication(  
        $this->rEcomPipeU,  
        $this->rEcomPipeP  
    )->getPaymentMethods();
```

SSL Issues

In some rare cases, when your web environment for example are customized, it happens that ssl certificate bundles can not be found, loaded, or things that makes SSL URL calls unsafe. Normally, EcomPHP prohibits all calls where certificates of SSL can not be validated as real, since anything else are considered invitations to man-in-the-middle-attacks. There is however a workaround for this, for test environments. When switching to production, this won't work.

```
$initFlow->setDebug(true);  
$initFlow->setSslValidation(false, false);
```

Those two lines will shut down SSL validation completely. Please note, that setSslValidation does not work if debug mode is shut off, so those two variables must work together. In our woocommerce plugin, you can set the special flag **DISABLE_SSL_VALIDATION**, in the advanced section to bypass the SSL validations in test mode.

ECommerce throws/returns errors during createPayment

Sometimes, when creating your payment, things go wrong. Sometimes it's all about the payload. Debugging the payload if that is the failing part, it might be tricky to find out what happened. Besides, before 1.0.2, reviewing the payload before completing the payment wasn't the easiest to do as the bookPayment()-function always did everything by itself in one step. From 1.0.2, payments can be "delayed" and put up in two steps:

```
$this->rb->setRequiredExecute(true);  
$delayPayment = $ECOM->createPayment( "PRIVATE_INVOICE" );
```

Having your payment ready to go like this, will prevent that the primary webservice call is firing during the createPayment()-process. Instead, to really invoke the booking, the call has now been set up to wait for your call. \$delayPayment will, in this mode return a small array that looks like

```
array(  
    'status' => 'delayed'  
) ;
```

In this state, you could do whatever sidestep you need to do. For example, if you need to check your payload before sending it, you can now run this, to see what it really contains...

```
var_dump($ECOM->getPayload());
```

And when ready too book the payment:

```
$finalBooking = $ECOM->Execute();
```

Exceptions

The curl library currently used usually passes exceptions through and when they are handled internally and separately, it will rethrow an exception with the previous one. The reason is that some exceptions sometimes (especially during the SOAP web services) throw more data that you might want to look at. For example, some exceptions are reached the following way (which also can be seen in the unitTest suite):

```
try {
    $wsdl->someThingWentWrong();
} catch ( \Exception $e ) {
    $previousException = $e->getPrevious();
    $exceptionMessage = $e->getMessage();
    $previousExceptionMessage = $previousException->getMessage();
    $details = isset($previousException->detail) ? $previousException->detail : null; // Resurs detailed
exceptions not always available
    if (!is_null($details)) {
        // The details always contains this collection of data
        $fixableByYou = $details->fixableByYou;
        $errorTypeDescription = $details->errorTypeDescription;
        $errorTypeId = $details->errorTypeId;
        $userErrorMessage = $details->userErrorMessage;
    }
}
```

Errors and EComPHP

EComPHP is not only a ecommerce wrapper/helper, for making orders. It also assures that you get as accurate error information as properly. Different flows has different ways to handle errors. We will try to explain how errors are thrown from ecommerce and how they will be handled by EComPHP. In newer releases of EComPHP we aim at helping furthermore to throw correct errors through the interfaces.

Table of contents

- [HTTP Errors](#)
 - [Extended HTTP Errors](#)
- [SOAP Errors](#)
 - [Extended SoapFaults](#)
- [Other standard errors](#)

HTTP Errors

If nothing else than a HTTP error code can be fetched, this will be primarily used. Such exceptions will first try to fetch the code and the message received in the HTTP header. For example,

HTTP/1.1 400 Bad Request

can be caught like this:

```
catch (\Exception $e) {  
    $e->getMessage(); // Bad Request  
    $e->getCode(); // 400  
}
```

Extended HTTP Errors

However, in some cases, when HTTP 400 errors are thrown (for example) there might also be a body delivered with the error that is not empty. Such errors is mostly common in the rest interfaces and might look like this:

```
{"timestamp":1527160469786,"status":400,"error":"Bad Request","exception":"java.lang.IllegalArgumentException","message":"totalAmount '340284' must not be greater than minLimit '50000.00'"}
```

If EComPHP catches a HTTP error, it will also check the body for this kind of JSON-strings and extend the errors thrown to the integrated platform.

Available from EComPHP 1.3.11, 1.0.38, 1.1.38 (and 2.0.0)

SOAP Errors

When running through SOAP, several errors might occur. The most common way to catch errors is through SoapClient standard error interface (which can be handled as a regular Exception or a SoapFault).

Extended SoapFaults

SoapFaults might in some cases be extended by ecommerce itself (See the constant codes at [Error handling \(Resurs error codes\)](#)). If this part of the exception exists in a SoapFault, EComPHP will set its priority to those parts. It will also add the popular string "fixableByYou", that might indicate if the problem has been created by you or if it comes from ecommerce (in some cases there might however be false alarms).

Other standard errors

The modules sometimes throw their own exceptions. Some of them are [listed here](#), and covers amongst others "SOAP Auth Errors", SSL and resolving problems, etc.

EComPHP Error Adaption

EComPHP has been built to follow and normalize Resurs flows (and when it's needed convert unsupported data to valid supported data), so that we in the end always get the same kind of answers regardless of what we'd like to do with Resurs API.

As described in [the notices about updatePaymentReference-section](#) it is highly valued that everything looks the same way all the time. However, depending on the flow, the acting might differ a bit. This also includes error handling. The later versions of EComPHP is written to be able to handle stringified exception codes. Normally, [the Exception-class in PHP](#) only accept long integers as a valid error code when exceptions are thrown. In some cases Resurs Bank may throw exceptions that is against this standard, with alpha numeric instead. EComPHP tries to handle this, with an extended exception class.

Default exception constructor

```
public Exception::__construct ([ string $message = "" [, int $code = 0 [, Throwable $previous = NULL ]]] )
```

EComPHP exception constructor

As of version 1.3.18 (or 1.1.45 in the older branches) our intention is to start reusing our recent ResursException again (meaning, it is no longer deprecated). The syntax for ResursException looks like this:

```
public function __construct(
    $message = 'Unknown exception',
    $code = 0,
    \Exception $previous = null,
    $stringifiedCode,
    $fromFunction = ''
)
```

ResursExceptions parameters

Parameter	Description
message	The regular error message thrown in exceptions.
code	The regular error code thrown in exceptions.
previous	The regular previous exception that was thrown in inherited exceptions.
stringifiedCode	Everything that may be thrown by an API that is not a long integer.
fromFunction	Extra variable that allows backtracing of the function where the exception was thrown.

Examples

With this exception handler we may be able to bypass exceptions that is normally not allowed to use by PHP. It is still possible, referring to our tests, to catch this exception with the standard exception syntax, like this (from the test suite):

```
try {
    throw new \ResursException('Fail', 0, null, 'TEST_ERROR_CODE_AS_STRING', __FUNCTION__);
} catch (\Exception $e) {
    $firstCode = $e->getCode();
}
try {
    throw new \ResursException('Fail', 0, null, 'TEST_ERROR_CODE_AS_STRING_WITHOUT_CONSTANT', __FUNCTION__);
} catch (\Exception $e) {
    $secondCode = $e->getCode();
}
```

When running those two exceptions in the test, the responses should look like this, in the end:

code	\$exception->getCode() result	Why?
TEST_ERROR_CODE_AS_STRING	1007	Every constant that exists in the RESURS_EXCEPTIONS class will be translated to a numeric.

TEST_ERROR_CODE_AS_STRING_WITHOUT_CONSTANT	TEST_ERROR_CODE_AS_STRING_WITHOUT_CONSTANT	If the constant does not exist in the RESURS_EXCEPTIONS class EComPHP will leave the error code as it was from its initial state. The purpose is to keep compatibility intact if Resurs Ecommerce ever adds more error codes in the future.
--	--	---

This exception however, supports to directly ask for the stringified code by something like this: `$e->getStringifiedCode()`. But to be absolutely sure that we're not breaking anything, the output will be pushed through the regular `getCode()`-method.

Testing (EComPHP)

- Configuration and credentials
- Merging 1.0.x with 1.1.0
 - Keeping compatibility
- EComPHP Test Suite

⚠ Production tests

Do you have plans running this suite in production mode? PLEASE, DO NOT!

The EComPHP testing suite is a bunch of function that is being runned every time Resurs Bank are releasing the library patches. In this way, we can be assured that the most common function in the library also works properly. To assure that things also works properly for our merchant developers (if they are utilizing with EComPHP), we also do release the test suite for external testings. As soon as we can we will completely leave 1.0 as the way it has been developed is obsolete.

Do EComPHP have an EOL?

Yes, it has. But it's not defined yet.

Configuration and credentials

To set up the test suite properly, you should take a look at test.json.sample and when you do have credentials to our test environment, edit the file as test.json, and change the values in the file to the data that fits you best.

Merging 1.0.x with 1.1.0

Currently, to get things in sync, we build from EComPHP 1.0 (that is actually deprecated). When functions works in 1.0 we transfer the same data to 1.1 and run tests.

Keeping compatibility

Classes in v1.0 must be updated to keep compatibility with 1.1.

The list below covers the classes that must be translated, when merging into v1.1 - In a close future, it will also cover the classes that must be translated from v1.1 to v1.0, as long as v1.0 do have maintenance support.

v1.0	v1.1	Notes
\ResursCallbackTypes::	\Resursbank\RBEcomPHP\ResursCallbackTypes::	
\ResursMethodTypes::	\Resursbank\RBEcomPHP\ResursMethodTypes::	
\ResursCallbackReachability::	\Resursbank\RBEcomPHP\ResursCallbackReachability::	
\ResursExceptions::	\ResursExceptions::	Unchanged

EComPHP Test Suite

As of the release of EComPHP v1.1, we will provide a test suite compatible with PHPUnit and PHPStorm, to make sure that at least the simplified flow works properly (Resurs Checkout will come soon). Currently, the suite is quite simple built to test most standard payments in sweden (and one in norway).

The test suite is delivered in the git branch in /tests, and contains a test.json.sample. This must be configured and set up as test.json before the suite will run. The json file contains a bunch of standard settings, required for the test suite and is not delivered with any merchant username/password. You must make sure you have this available. Most of the tests are built for sweden so if you don't have any norwegian merchant account, you can leave this empty and the tests will skip that part. You should also make sure that you have the four payment methods listed in this configuration example below (invoice legal, invoice natural, card and new card) as the tests will pass through them all.

You might also want to change the urls in the end of configuration to see what works and not works from there. In this case you should make sure that the urls exists and are reachable from where you run the tests.

test.json

```
{  
    "sweden": {
```

```
    "username": "",
    "password": ""
},
"norway": {
    "username": "",
    "password": ""
},
"alertReceivers": [],
"alertFrom": [],
"availableMethods": {
    "invoice_legal": "invoice_legal",
    "invoice_natural": "invoice_natural",
    "card": "card",
    "card_new": "card_new"
},
"availableMethodsNorway": {
    "invoice_natural": "invoice_natural",
    "card": "card",
    "card_new": "card_new"
},
"callbackUrl": "http://myurl.test.com/",
"signUrl": "https://test.resurs.com/signdummy/?success=true",
"failUrl": "https://test.resurs.com/signdummy/?success=false"
}
```

Version 2.x (EComPHP)

Source code is located at <https://bitbucket.org/resursbankplugins/ecom2>

ECom2 (or EComPHP version 2) is ecommerce library built for PHP starting with v8.1, with the intention to only support the Resurs REST-backend [MAPI Checkout Flow](#). None of the older API's (except for RCO) is included, so SOAP is no longer required. The library is also built to be an independent stand-alone library.

Table of contents

- [Table of contents](#)
- [Requirements](#)
- [Getting started](#)
 - [Config](#)
 - [Exceptions](#)
 - [Exceptions](#)
 - [Exceptions/Validation](#)
 - [Libraries](#)
 - [Api](#)
 - [Cache](#)
 - [Collection](#)
 - [DataStorage](#)
 - [Event](#)
 - [Locale](#)
 - [Log](#)
 - [Model](#)
 - [Network](#)
 - [Simplified](#)
 - [Utilities](#)
 - [Validation](#)
 - [Modules](#)
 - [Annuity](#)
 - [PaymentMethod](#)
 - [RCO](#)
 - [RCO Callback](#)
 - [Store](#)
 - [Callbacks](#)

Requirements

- PHP 8.1
- SSL-connectivity (preferably OpenSSL)
- CURL (ext-curl with necessary libraries) 7.61.0 or higher
- **Curl with CURLAUTH_BEARER-support**

If you run Ubuntu (bionic) the lowest available curl version will probably be 7.58.0 (focal is currently - aug22 - on 7.68) and in many systems bearers was introduced in 7.61.0 - however, you need to make sure it is really present in newer releases too.

Getting started

Config

The **Config::setup()** must always be called before performing any API call. This method creates a configured instance of the ECom library with all necessary information to execute API calls and perform related actions (such as logging, caching, data persistence etc.). **Config** acts as a singleton and the instance is stored in **Config::\$instance**.

This is a little unorthodox, but it allows for a single point of configuration for the library. This allows for simpler integrations. For example, if you utilize ECom from various places within your application you could call **Config::setup()** at a central point early in your application lifecycle, making sure that any calls to the library are configured properly always, regardless of where you may require the library. This can also have a beneficial impact on dependency management.

Exceptions

Exceptions

Custom Exception classes.

- [ApiException](#) | Base exception for all API exceptions.

- AuthException | Exception for authentication errors.
- CacheException | Exception for cache errors.
- CurlException | Exception for curl errors.
- EventException | Exception for event errors.
- EventSubscriberException | Exception for event subscriber errors.
- FilesystemError | Exception for filesystem errors.
- IOException | Exception for IO errors.
- TestException | Exception for test errors.
- ValidationException | Exception for validation errors, see below.

Exceptions/Validation

Custom Exception classes utilized for data validation.

NOTE: All of these exceptions are subclasses of ValidationException.

- EmptyValueException | Value was not expected to be empty.
- FormatException | Value was not in expected format.
- IllegalCharsetException | Value contained illegal characters.
- IllegalTypeException | Value was not of expected type.
- IllegalValueException | Value was not allowed.
- MissingKeyException | Required key was not found in array.

Libraries

Libraries are located under **src/Lib**. Libraries are allowed to communicate with each other. As such they are allowed to have dependencies on each other. These classes contain abstract business logic and are not meant to be called directly by the end user. Libraries are not allowed to communicate with modules, modules are however allowed to communicate with libraries.

Api

General API classes and functionality.

- Mapi | Contains centralized business logic for communication with Merchant API.

Cache

- AbstractCache | Abstract base class for all cache implementations.
- CacheInterface | Interface for all cache implementations.
- Filesystem | Filesystem cache implementation.
- None | No cache implementation. This is the default cache implementation.
- Redis | Redis cache implementation.

Collection

- Collection | Abstract base class for all collection implementations.

DataStorage

Work in progress. Intended to be a persistent data storage implementation.

Event

Work in progress. Intended to be an event system to communicate between Modules.

Locale

Work in progress. Intended to help with localization and country availability.

Log

Logging functionality.

- FileLogger | File logger implementation.
- LoggerInterface | Interface for all logger implementations.
- LogLevel | Log level enumeration.
- StdoutLogger | Stdout logger implementation.

Model

Data object implementations.

- Model | Abstract base class for all model implementations.

Network

Network communication functionality.

- ApiType | API type enumeration.
- AuthType | Authentication type enumeration.
- ContentType | Content type enumeration.
- Curl | Curl implementation.
- RequestMethod | Request method enumeration.
- Url | URL implementation.
- Curl/Header | Helper methods for Curl headers.
- Model/Auth/Basic | Basic authentication model.
- Model/Auth/Jwt | JWT authentication model.
- Model/Header | Header model.
- Model/JwtToken | JWT token model.
- Model/Response | Generic response model. This is the expected return type of all API calls.

Simplified

Data and logic specifically related to the Simplified API.

- Config | Configuration object for Simplified API.

Utilities

Generic functionality that does not belong to any specific library.

- Generic | Methods to extract Composer and Docblock information.
- DataConverter | Helps us convert anonymous arrays to known objects. Also lets us convert multidimensional arrays to collections.
- DataConverter/TestClasses | Test classes for DataConverter.

Validation

Functionality to help us validate various kinds of data. Classes are separated by the data type they validate.

- ArrayValidation | Validation for arrays.
- BoolValidation | Validation for booleans.
- FloatValidation | Validation for floats.
- IntValidation | Validation for integers.
- StringValidation | Validation for strings.

Modules

Modules are independent pieces of functionality that are meant to be used by the end user. Modules are not allowed to communicate with each other to allow for maximum flexibility.

Annuity

Integration of **annuity factors**, currently incomplete. At present this only reflect how we could leverage the **Event** library to implement logic that will fetch annuity factor information from the API when payment methods are being fetched (fetching factors for each method fetched from the API). Since modules are not allowed to have knowledge of each other, this is the best way to achieve this.

PaymentMethod

Integration of **payment methods**, currently incomplete. Work in progress.

RCO

Implementation of the Checkout API (iframe based checkout).

- Repository | Repository for RCO.
- Api/GetPayment::call() | Fetch payment information from the API.
- Api/InitPayment::call() | Initialize payment session (iframe) with the API.
- Api/UpdatePayment::call() | Update payment session in the API.
- Api/UpdatePaymentReference::call() | Update payment reference in the API.
- Model/* | Model classes for API requests and responses.

Please note that all API calls should be performed through **Repository**.

When using the RCO you first need to call **InitPayment** to initialize the payment session. You will be required to provide a reference for this session which should be your order number if you already have that on hand at this point. You will otherwise be able to update this value later using **UpdatePaymentReference**. **InitPayment** Will supply you with the iframe to allow checkout. **UpdatePayment** Allows you to update the payment session with new items etc. after the payment session has already been created, so you do not need to re-create the session every time the cart changes for example. Whenever the totals in your platform change you should call this endpoint to update the payment session which will reflect the new total within the iframe through JS sockets. You can call **GetPayment** to fetch the payment session information at any time. When the client completes their purchase the session will be converted to an actual payment.

RCO Callback

Documentation TBD.

Store

Implementation of **stores** in the Merchant API (MAPI).

- Repository | Repository for Store.
- Api/GetStores::call() | Fetch list of available stores from the API.
- Model/* | Model classes for API requests and responses.

Please note that all API calls should be performed through **Repository**.

Every API account has one or more stores. You can fetch the list of available stores through the **Repository** class. You will need your store(s) for subsequent API calls to fetch payment methods for example. The **Repository** class will return a **Collection** of **Store** objects read either from cache or directly from the API.

Callbacks

Incoming callbacks are not explicitly handled by the SDK. However, it can still handle the data models for the callbacks sent from Resurs.

Callbacks are handled by the repository located under **src/Lib/Module/Callback**.

In its simplest form (as the callback types are not auto discovered), you can use the following code to fetch the proper callback model for Authorization (where the repository itself also handles the data received via [php://input](#)).

```
use Resursbank\Ecom\Module\Callback\Repository;
$this->callbackModel = (new Repository(CallbackType::AUTHORIZATION))>getCallbackModel();
```

The models are based on the data sent from Resurs (which you can read about [here](#), and they are stored at **src/Lib/Model/Callback** as the two below:

- Authorization
- Management

API Types

This page is available if you need a closer look at each specific type. Under Simple Type you'll find the common types.

- [address](#)
- [basicPayment](#)
- [bookingResult](#)
- [bookPaymentResult](#)
- [bookPaymentStatus](#)
- [cardData](#)
- [countryCode](#)
- [customer](#)
- [customerCard](#)
- [customerIdentification](#)
- [customerIdentificationResponse](#)
- [customerType](#)
- [digestAlgorithm](#)
- [digestConfiguration](#)
- [ECommerceError](#)
- [formElement](#)
- [fraudControlStatus](#)
- [invoiceData](#)
- [invoiceDeliveryType](#)
- [limit](#)
- [limitApplicationFormAsCompiledForm](#)
- [limitApplicationFormAsObjectGraph](#)
- [limitDecision](#)
- [mapEntry](#)
- [option](#)
- [payment](#)
- [paymentData](#)
- [paymentDiff](#)
- [paymentDiffType](#)
- [paymentMethod](#)
- [paymentMethodType](#)
- [paymentSession](#)
- [paymentSpec](#)
- [paymentStatus](#)
- [pdf](#)
- [searchCriteria](#)
- [signing object](#)
- [Simple Types...](#)
- [sortAlternative](#)
- [sortOrder](#)
- [specLine](#)
- [webLink](#)
- [withMetaData](#)

address

The customer address details.

Contains elements as defined in the following table.

Component	Type	Occurs	Nillable?	Description
fullName	nonEmptyString	0..1	No	The full (possibly composite name) of the customer. Note: if you use firstName and lastName then fullName is not required
firstName	nonEmptyString	0..1	No	If available, the first name of the customer. Note: if you use fullName then firstName and lastName are not required
lastName	nonEmptyString	0..1	No	If available, the last name of the customer. Note: if you use fullName then firstName and lastName are not required
addressRow1	string	1..1	No	The first row of the customer address.
addressRow2	nonEmptyString	0..1	No	The second row of the customer address. In practice: Located as a second line on printouts and graphics, like invoices. Attn. If you are using this tag do not leave it empty!
postalArea	nonEmptyString	1..1	No	The postal area.
postalCode	nonEmptyString	1..1	No	The postal code.
country	countryCode	1..1	No	The country in which this address is located.



Do not do any logical operations on the strings, for example to determine if the customer is a NATURAL or LEGAL.

basicPayment

The basic payment information returned by a search. It does not contain all payment details, but enough to be presented in a list of links to the full details.

Contains elements as defined in the following table.

Component	Type	Occurs	Nillable?	Description
paymentId	id	1..1	No	The payment identity.
paymentMethodId	id	1..1	No	The payment method identity.
paymentMethodName	nonEmptyString	1..1	No	The payment method name.
governmentId	nonEmptyString	1..1	No	The customer government identity.
fullName	string	1..1	No	The full (possibly composite name) of the customer.
booked	dateTime	1..1	No	The timestamp of the payment booking.
modified	dateTime	0..1	No	The timestamp of latest payment modification.
finalized	dateTime	0..1	No	The timestamp of the latest payment finalization.
totalAmount	positiveDecimal	1..1	No	The total payment amount. Please be aware that this is the current total payment amount, i.e. taking into consideration the status of the various payment parts.
frozen	boolean	1..1	No	Whether or not the payment has been frozen by the fraud system for a more detailed control.
status	paymentStatus	0..100	No	The status of the payment as a list of status values.

bookingResult

The result of the payment booking attempt.

Contains elements as defined in the following table.

Component	Type	Occurs	Nillable?	Description
paymentId	id	1..1	No	The identity of the payment. Note: this is not the same as the payment session identity.
fraudControlStatus	fraudControlStatus	1..1	No	The result of the fraud control. This is only available if the fraud control was made synchronously

bookPaymentResult

The response type when using [Simplified Basic Show Flow Service](#)

Name	Type	Occurs	Nillable?	Description
paymentId	id	1..1	No	The ID for the payment, same as preferredId if that is specified, else it is an auto generated value.
bookPaymentStatus	bookPaymentStatus	1..1	No	The status of the payment.
signingUrl	String	0..1	Yes	If the bookPaymentStatus is SIGNING then this will contain the signing URL.
approvedAmount	positiveDecimal	1..1	No	The approved credit amount.
customer	customer	0..1	Yes	The address of the customer.

bookPaymentStatus

The different status that can be returned when using the [Simplified Flow API](#)

Status Code	Description
FINALIZED	The payment is finalized.
BOOKED	The payment is booked and you will have to finalize it on your own. To finalize booked payments automatically you could set a flag in the bookPayment.
FROZEN	The payment is currently frozen. This typically means that there is something that needs further investigation before the payment can be finalized.
SIGNING	The payment requires signing.
DENIED	The payment is denied.

cardData

The card payment specification data. Can be used for both card and account.

- i** Only set one of CardNumber and Amount, both should never be specified.

If agreed upon with Resurs Bank, the merchant can let the customer use an existing card without entering a card number, Resurs Bank will retrieve the card/account from the system and charge it. In this case both fields should be left empty, only the governmental ID is sent in and an e-signing will be raised.

Component	Type	Occurs	Nillable?	Description
cardNumber	string	0..1	Yes	If it is a purchase with an existing card specify the card/account number here.
amount	nonEmptyString	0..1	Yes	If customer applies for a new card/account, specify the credit amount that is applied for. If it is purchase with existing card omit this tag.

countryCode

The country code as defined by the ISO 3166-1 standard.

Value	Description
SE	Sweden
NO	Norway
DK	Denmark
FI	Finland

customer

Details about a (potential) customer, be it natural or legal.

Contains elements as defined in the following table.

Component	Type	Occurs	Nillable?	Description
governmentId	nonEmptyString	0..1	No	Identifies a customer uniquely within the country. <ul style="list-style-type: none">• SE: Personnummer/Organisationsnummer• DK: CPR-number• NO: Fødselsnummer• FI: Henkilötunnus
address	address	1..1	No	The customer address. It's only used for fraud control. billingAddress will always be the customers registered address.
cellPhone	string	0..1	No	The customer's cell phone number is specified here. Mandatory if Swish is to be used as payment method
email	nonEmptyString	0..1	No	The customer email address.
type	customerType	1..1	No	The customer type.

```
<customer>
    <governmentId>7312195873</governmentId>
    <address>
        <fullName>Asgeirsen Asgeir</fullName>
        <firstName>Asgeir</firstName>
        <lastName>Asgeirsen</lastName>
        <addressRow1/>
        <postalArea>Inøja</postalArea>
        <postalCode>1234</postalCode>
        <country>SE</country>
    </address>
    <phone/>
        <email>javatest@resurs.se</email>
        <type>NATURAL</type>
</customer>
```

extendedCustomer

Extended Customer is an extension of the Customer, it is used in the Simplified Shop Flow



Occurrences of fields below adhere to the xsd requirements.

Beware that some fields are required by the bank product the client and Resurs agreed upon.

Component	Type	Occurs	Nullable?	Description
cellPhone	string	0..1	false	The customer's cell phone number is specified here. Is mandatory if you are using Swish payment method via Resurs Bank
yourCustomerId	string	0..1	true	Your internal customer ID, if you want it on invoice etc.
deliveryAddress	address	0..1	true	If the delivery address should be changed, specify a new delivery address here else leave it empty.
contactGovernme ntId	string	0..1	true	If your customer is of LEGAL type (company), set the contact person for the payment. If a none LEGAL (i.e. NATURAL) leave it empty. Contact civic number. Company applicable only for Sweden.

Details about a (potential) customer, be it natural or legal.

Contains elements as defined in the following table.

Component	Type	Occurs	Nullable?	Description
governmentId	nonEmptyString	0..1	No	Identifies a customer uniquely within the country. <ul style="list-style-type: none">• SE: Personnummer/Organisationsnummer• DK: CPR-number• NO: Fødselsnummer• FI: Henkilötunnus
address	address	1..1	No	The customer address. It's only used for fraud control. billingAddress will always be the customers registered address.
cellPhone	string	0..1	No	The customer's cell phone number is specified here. Mandatory if Swish is to be used as payment method
email	nonEmptyString	0..1	No	The customer email address.
type	customerType	1..1	No	The customer type.

```
<customer>
  <governmentId>7312195873</governmentId>
  <address>
    <fullName>Asgeirsen Asgeir</fullName>
    <firstName>Asgeir</firstName>
    <lastName>Asgeirsen</lastName>
    <addressRow1/>
    <postalArea>Inøja</postalArea>
    <postalCode>1234</postalCode>
    <country>SE</country>
  </address>
  <phone/>
    <email>javatest@resurs.se</email>
    <type>NATURAL</type>
</customer>
```

customerCard

Component	Type	Occurs	Nullable?	Description
SEQUENCE		1..1		
governmentId	string	1..1	No	The government identity of the customer for which to retrieve bonus points.
customerType	customerType	0..1	No	The type of customer to retrieve.
cardNumber	string	1..1	No	A card number tied to the supplied government ID in Resurs Bank.

customerIdentification

Information used to identify the customer, in order to be able to retrieve its bonus.

Component	Type	Occurs	Nillable?	Description
CHOICE		1..1		
token	identificationToken	1..1	No	The identification token created by the identifyCustomer function. If the token isn't valid a fault will be returned.
customerAccount	customerCard	1..1	No	Information used to identify the customer, in order to be able to retrieve its bonus.

customerIdentificationResponse

Component	Type	Occurs	Nillable?	Description
token	identificationToken	1..1	No	A string used to identify a customer. Alphanumeric characters, 100 characters long.
expirationDate	dateTime	1..1	No	When this token expires. It cannot be used after that date. This date should at least be a couple of years in the future.

customerType

customerType

Description: The type of customer.

Value	Description
LEGAL	The customer is a legal customer, i.e. a company.
NATURAL	The customer is a natural customer, i.e. a person.

digestAlgorithm

Value	Description
MD5	
SHA1	

digestConfiguration

Contains elements as defined in the following table.

Component	Type	Occurs	Nillable?	Description
digestAlgorithm	digestAlgorithm	1..1	No	A digest will be hashed using the specified algorithm. The currently supported algorithms are MD5 and SHA1.
digestParameters	string	1..*	No	The digest will be created by hashing one or more parameters. These are the same as those available for specification in the URI.
digestSalt	string	0..1	No	To improve the strength of the hash, a secret hash salt must be provided. The salt will be added after all digest parameters

DigestAlgorithm

Value	Description
MD5	
SHA1	

ECommerceError

All errors/exception are returned as ECommerceError
Contains elements as defined in the following table.

Component	Type	Occurs	Nillable?	Description
errorTypeDescription	nonEmptyString	1..1	No	The textual description of the error type.
errorTypeid	int	1..1	No	Indicates which kind of error this is.
fixableByYou	boolean	1..1	No	"If this error has been avoided with some other input" = "It's a Resurs Bank problem"
userErrorMessage	nonEmptyString	1..1	No	An error message intended for the user of the web shop. This message must be shown!

formElement

The detailed specification of the form element. While new form elements might pop up, [we have a few which you can look out for](#) (if you happen to have the data the form element requests, or need the data from a particular form element).

Attributes

Name	Type	Required?
type	string	Yes
name	string	No
mandatory	boolean	No
length	int	No
Min	int	No
Max	int	No
uint	string	No
level	unsignedByte	No

Contains elements as defined in the following table.

Component	Type	Occurs	Nillable?	Description
label	string	0..1	No	The form element label. Typically, this is displayed as the field header.
description	string	0..1	No	A textual description of the form element. Typically, this is displayed when the user hovers the mouse pointer over the element.
format	string	0..1	No	The form element format as a regular expression. When submitted, the form element will be validated against this, and it may be a good idea to use this to validate the value of the element already on the client side.
format-message	string	0..1	No	The message to be presented when the format validation fails.
default-value	string	0..1	No	The default value.
option	option	0..*	No	If the form element has a predefined set of possible values, these are presented as a list.

fraudControlStatus

The status of the payment fraud control.

Value	Description
FROZEN	The payment is currently frozen. This typically means that there is something that needs further investigation before the payment can be finalized.
BOOKED	The payment is not frozen, and may be finalized at any time.
CONTROL_IN_PROGRESS	<i>The deprecated flow:</i> the automatic fraud control is still in progress. It will result in FROZEN or BOOKED. In the simplified flow you will get an intermediate status of FROZEN when the control is still running.

invoiceData

invoiceData is used when you specify the data when a paymentmethod is an Invoice.

-  Find out if the payment methode is an invoice by making a getPaymentMethods. If <specificType>INVOICE</specificType> in getPaymentMethodsResponse, payment methode is an invoice.

This should only be used when the finalizelfBooked in [paymentData](#) is set to true.

Component	Type	Occurs	Nillable?	Description
invoiceId	id	0..1	Yes	The invoice number. To be used if finalizelfBooked is set to <code>true</code> . This will be printed on the invoice. For payment methods other than INVOICE, setting this will generate an error.
invoiceDate	dateTime	0..1	Yes	The invoice date. This will be printed on the invoice. For payment methods other than INVOICE, setting this will generate an error.
invoiceDeliveryType	invoiceDeliveryType	0..1	Yes	This option will let you decide how the INVOICE should be delivered. NONE, EMAIL or by POST. Default: EMAIL

invoiceDeliveryType

The type of how an invoice should be delivered to a customer.

Value	Description
EMAIL	The invoice will be delivered e-mail.

limit

Detailed information about the limit.

Contains elements as defined in the following table.

Component	Type	Occurs	Nillable?	Description
approvedAmount	positiveDecimal	1..1	No	The amount that has been approved.
decision	limitDecision	1..1	No	The limit decision.
customer	customer	1..1	No	The customer details.
limitRequestId	id	1..1	No	Identifies this limit request uniquely, whether it's granted or not. It can be used to request more information, by phone, about the application from Resurs Bank in special cases.

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns3:submitLimitApplicationResponse xmlns:ns3="http://ecommerce.resurs.com/v4/msg/shopflow" xmlns:ns2="http://ecommerce.resurs.com/v4/msg/exception">
      <return>
        <approvedAmount>3030</approvedAmount>
        <decision>GRANTED</decision>
        <customer>
          <governmentId>7312195873</governmentId>
          <address>
            <fullName>Asgeirsen Asgeir</fullName>
            <firstName>Asgeir</firstName>
            <lastName>Asgeirsen</lastName>
            <addressRow1/>
            <postalArea>Inøja</postalArea>
            <postalCode>1234</postalCode>
            <country>SE</country>
          </address>
          <phone/>
          <email>javatest@resurs.se</email>
          <type>NATURAL</type>
        </customer>
      </return>
    </ns3:submitLimitApplicationResponse>
  </soap:Body>
</soap:Envelope>
```

limitApplicationFormAsCompiledForm

A complete limit application form in HTML format for embedding in the webshop page. Upon submission, the form itself will generate the application data to be submitted to Resurs Bank. The form contains HTML elements and JavaScript based validation will be performed when the form is submitted.

Contains elements as defined in the following table.

Component	Type	Occurs	Nillable?	Description
form	nonEmptyString	1..1	No	The main form content. It should be placed somewhere in the HTML BODY element of the page. Note: The form contents are URL encoded and must be decoded before insertion into the page.
formHeader	string	1..1	No	The form header data. It should be placed as a child of the HTML HEAD element of the page. Note: The form header is URL encoded and must be decoded before insertion into the page.
formOnLoad	string	1..1	No	A JavaScript function call that needs to be performed once the page has been fully rendered. This is done by placing it as an on-load event of the HTML BODY element of the page. Note: The form on-load event is URL encoded and must be decoded before insertion into the page.

limitApplicationFormAsObjectGraph

The limit application form as an object graph. All elements to be presented on the form are returned and need to be rendered accordingly.

Contains elements as defined in the following table.

Component	Type	Occurs	Nillable?	Description
formId	string	1..1	No	The identity of the form. If a form has been previously filled in by the customer and a new application attempt has to be made for some reason, without any change to the parameters, comparing this identity to the identity on the previous form will show if the new form needs to be presented, or if the same data can be re-submitted.
formElement	formElement	0..*	No	The list of form elements.

limitDecision

The possible decisions returned by a limit application.

Value	Description
DENIED	No limit at all is given.
GRANTED	The applied limit or more is given.
TRIAL	This would not normally be returned, but if it is, handle it as DENIED.

mapEntry

A representation of a simple map. WebService frameworks are not good at supporting maps natively. An instance of this object must be unique in regard to the key within the list.

Contains elements as defined in the following table.

Component	Type	Occurs	Nillable?	Description
key	nonEmptyString	1..1	No	The key of the pair.
value	string	0..1	No	The value of the pair.

option

An option in a list of predefined values.

Attributes

Name	Type	Required?
checked	boolean	No

Contains elements as defined in the following table.

Component	Type	Occurs	Nillable?	Description
label	string	0..1	No	The list option label. Typically, this is displayed as the option name.
value	string	0..1	No	The list option value. Typically, this is not shown to the customer.
description	string	0..1	No	A textual description of the list option. Typically, this is displayed when the user hovers the mouse pointer over the option.

payment

The detailed payment information. In addition to the overall information about the payment, it also contains full details about all part payments associated with the payment. The part payments are the complete history of the payment. (The current state of the payment, if needed, must be calculated client side.)

Contains elements as defined in the following table.

Component	Type	Occurs	Nillable?	Description
id	id	1..1	No	The payment identity.
totalAmount	positiveDecimal	1..1	No	The current total amount of the part payments making up this payment.
metaData	mapEntry	0..*	No	The meta data associated with the payment as key/value pairs.
limit	positiveDecimal	1..1	No	The limit (if any) associated with the payment. This could for instance be the credit amount applied for.
paymentDiffs	paymentDiff	0..*	No	The parts that make up this payment.
customer	customer	1..1	No	The payment customer information.
deliveryAddress	address	0..1	No	The delivery address, if one has been specified.
booked	dateTime	1..1	No	The timestamp of the payment booking.
finalized	dateTime	0..1	No	The timestamp of the latest payment finalization.
paymentMethodId	id	1..1	No	The identity of the payment method used for the payment.
paymentMethodName	String	0..1	Yes	The name of the payment method.
fraud	boolean	1..1	No	Whether or not the payment has been flagged as fraudulent.
frozen	boolean	1..1	No	Whether or not the payment has been frozen by the fraud system for a more detailed control.
status	paymentStatus	0..100	No	The status of the payment as a list of status values.
storeId	id	0..1	No	The identity of the actual store for the representative. This ID is defined by Resurs Bank. You can receive the list of stores tied to your user (representative) if you wish.
paymentMethodType	paymentMethodType	1..1	No	The type the payment, i.e invoice, new account etc.
totalBonusPoints	nonNegativeInteger	1..1	No	The current sum of bonus points of the payment's diffs making up this payment.

paymentData

The data for the payment during a Simplified Basic Show Flow.

Component	Type	Occurs	Nillable?	Description
preferredId	id	0..1	Yes	<p>The preferred payment ID for the payment, usually the order ID. This ID has to be unique per representative and payment. If nothing is specified, the system will generate an unique ID. This ID will in both cases be returned in the bookPaymentResult as paymentId.</p> <p>If you are to offer Swish (SE) or card payment via Nets through Resurs Bank, the maximum characters are 35 (Swish) and 32 (Nets) and <i>allowed characters are a-z A-Z 0-9 -</i></p>
paymentMethodId	id	1..1	No	The payment method to be used when doing a payment.
preferredTransactionId	id	0..1	Yes	An identifier which is reported back in economic reports. Can thus be used as a key to track transactions. It's optional. Only to be used when finalizelfBooked is set to <code>true</code> , else use this parameter in finalizePayment!
customerIpAddress	string	1...1	No	The IP address of the customer of the payment.
waitForFraudControl	boolean	0..1	Yes	If the system should wait to return a response until the fraud control has run. If you set waitForFraudControl to <code>false</code> you will have to register the callback AUTOMATIC_FRAUD_CONTROL to get notified when the control is finished, with result FROZEN or THAWED . Read more... Default: true .
annullIfFrozen	boolean	0..1	Yes	If the fraud freezes a payment, it will automatically annul the payment. Set it to <code>true</code> if you cannot wait for the order to be thawed before delivering the purchased items. Default: false .
finalizelfBooked	boolean	0..1	Yes	Will automatically finalize (debit) the payment if you will get an okay from the fraud control. If you set the waitForFraudControl to false you need to register the callback FINALIZATION if you want to get notified when a payment is finalized. Default: false .

paymentDiff

Detailed information about this part of the payment.

Contains elements as defined in the following table.

Component	Type	Occurs	Nillable?	Description
type	paymentDiffType	1..1	No	The type of payment part.
transactionId	id	0..1	No	This ID will be presented on the payment specifications sent from Resurs Bank. It can be left out, and in that case Resurs will use the payment ID as transaction ID.
created	dateTime	1..1	No	The timestamp of the payment part creation.
createdBy	string	0..1	No	Who created the payment part.
paymentSpec	paymentSpec	1..1	No	The full specification details of the payment part.
orderId	id	0..1	No	The order number as specified by the representative.
invoiceId	id	0..1	No	The invoice number as specified by the representative.
documentNames	nonEmptyString	0..*	No	The names of the documents associated with this payment part.

paymentDiffType

The type of payment part.

Value	Description
AUTHORIZE	The payment part is an authorization request.
DEBIT	The payment part is a debit request.
CREDIT	The payment part is a credit request.
ANNUL	The payment part is an annulment request.

Types of payment diffs

Today, the system supports the following types of payment diffs:

Type	Description	Banking system	Order value	When is it used?	Comments
AUTHORIZE	Reservation of part / whole of the allotted limit.	Authorization of amounts in accounts.	Increases	When a purchase is booked.	In earlier drafts DEBIT has had dual functions, authorization and billing has been done by
DEBIT	Making of a debit.	Transaction of the amount on account. New authorization on the remaining amount.	Unchanged	When goods are shipped to the customer.	See above
CREDIT	Effecting a crediting.	Crediting the amount of the account.	Reduced	When a return is received.	
ANNUL	Annulment of the reservation amount.	New authorization on the remaining amount.	Reduced	When a booked purchase is canceled.	

paymentMethod

Describes a way in which a purchase can be made. Typically, a representative will have two or more payment methods.

Contains elements as defined in the following table.

Component	Type	Occurs	Nillable?	Description
id	id	1..1	No	The identity of the payment method.
description	string	1..1	No	A textual description of the payment method. This is a help to the developer and should not be shown in the shop. That wording the merchant needs to figure out himself.
legalInfoLinks	webLink	0..*	No	A link to pages at Resurs Bank displaying miscellaneous legal information, such as annual-percentage-rate (APR) details etc. Note: the representative must display these links, or embed the output from the ShopFlowService.getCostOfPurchaseHtml(...) method into their shop. Irrespective of how the representative decides to implement it, the information must be available whenever a Resurs Bank payment method is presented.
minLimit	positiveDecimal	1..1	No	The minimum amount for which a limit application can be performed on this payment method. If the payment amount is less than this, the limit application will be performed on the minimum amount. Note: this information must not be presented to the customer.
maxLimit	positiveDecimal	1..1	No	The maximum amount for which a limit application can be performed on this payment method. If the payment amount is greater than this, the payment method may not be used. Note: this information must not be presented to the customer.
type	paymentMethodType	1..1	No	The type of the payment method. Can be used to group payment methods, and/or, apply some other logic to them. (Deprecated)
specificType	string	1..1	No	The type of the Resurs credit payment methods: INVOICE, CARD, REVOLVING_CREDIT, PART_PAYMENT PSP payment methods: DEBIT_CARD & CREDIT_CARD SWISH
customerType	customerType	1..*	No	The customer types the paymentMethod is valid for: NATURAL, LEGAL

paymentMethodType

Payment methods are divided into groups. This information can be used to retrieve orders based on which payment method type was used.

Value	Description
INVOICE	The customer wants to have an invoice sent to him, where he's able to pay the whole amount at once.
REVOLVING_CREDIT	The customer wants to start a new account, and finance the purchase with that account's credit limit. The purchase can be paid in a series of installments. In most cases there will be made a branded card sent to customer, depending on how your collaboration with Resurs Bank looks like. A credit application will need to be made, and the customer needs to sign a credit contract.
CARD	Every card issued by Resurs Bank falls into this group, while cards from other banks or credit institutions do NOT! This means that customers which have a branded Resurs card from another of Resurs representatives still can use it in your webshop.
BONUS	Not implemented.

paymentSession

The detailed information about the payment session.

Contains elements as defined in the following table.

Component	Type	Occurs	Nillable?	Description
id	id	1..1	No	The identity of the payment session. If one was specified by the representative, it will be used, otherwise it will be generated.
expirationTime	dateTime	1..1	No	When the payment session expires. Sessions will be automatically pruned after the expiration time, and if the payment is still valid, a new session must be created.
limitApplicationForm AsObjectGraph	limitApplicationForm AsObjectGraph	1..1	No	The limit application form as a graph of object. This is for use by representatives that want to generate the form themselves.
limitApplicationForm AsCompiledForm	limitApplicationForm AsCompiledForm	1..1	No	The limit application form as compiled HTML. This is for use by representatives that want to use the form created by Resurs Bank. Note: if no form action was supplied, this will be empty.

paymentSpec

The payment details. In it's simplest form it's just sum, i.e. totalAmount and totalVatAmount are set, but there are no specLines. If nothing else is said you shall send specLines .

Contains elements as defined in the following table.

Component	Type	Occurs	Nillable?	Description
specLines	specLine	0..*	No	The list of payment lines. In the case you're sending a simple payment, without lines, this parameter should be left empty. Sending payment lines may, or may not, be mandatory, depending on the contract with Resurs Bank.
totalAmount	positive Decimal	1..1	No	The total payment amount. The sum of all line amounts (if there are lines supplied) including VAT. If this payment is without lines this is the only value to be set on the payment spec.
totalVatAmount	decimal	0..1	Yes	The total VAT amount of the payment when there are specification lines supplied. If there are no lines this field must be empty (null).

Paymentspec - speclines



Observe that in order to have an invoice with specified order data, make sure to include the specLines in the web service call.

specLines are not mandatory for processing payments.

specLines can vary between start, finalize, credit and annul. It doesn't matter. Only the sum matter.

specLines make better invoices and help the merchant

The code below shows an example of one paymentSpec row when calling startPaymentSession method

paymentSpec example in bookPayment

Paymentspec - rounding

[see Rounding](#)

paymentStatus

Status Code	Description
DEBITABLE	Can be debited.
CREDITABLE	Can be credited.
IS_DEBITED	Is debited.
IS_CREDITED	Is credited.
IS_ANNULLLED	Is annulled

pdf

A PDF document

Contains elements as defined in the following table.

Component	Type	Occurs	Nillable?	Description
name	string	1..1	No	The name of the document.
pdfData	base64Binary	1..1	No	The document data.

searchCriteria

A set of search criteria.

Contains elements as defined in the following table.

Component	Type	Occurs	Nillable?	Description
anyId	id	0..1	No	Any identity associated with the payment, not just the payment identity. This includes the transaction identity, the invoice and credit note numbers, as well as the order number of any payment part belonging to the payment. If the exact payment identity is known, it is generally a better idea to use the <code>getPayment()</code> method as that is a lot quicker.
paymentMethodId	id	0..1	No	The identity of the payment method.
governmentId	nonEmptyString	0..1	No	The desired customer government identity.
customerName	string	0..1	No	The desired customer name. Please be aware that searches will be performed on the full (possibly composite name) of the customer.
bookedFrom	dateTime	0..1	No	The earliest desired payment booking timestamp.
bookedTo	dateTime	0..1	No	The latest desired payment booking timestamp.
modifiedFrom	dateTime	0..1	No	The earliest desired payment modification timestamp.
modifiedTo	dateTime	0..1	No	The latest desired payment modification timestamp.
finalizedFrom	dateTime	0..1	No	No The earliest desired payment finalization timestamp.
finalizedTo	dateTime	0..1	No	The latest desired payment finalization timestamp.
amountFrom	positiveDecimal	0..1	No	The minimum desired total payment amount. Please be aware that searches will be performed on the current total payment amount, i.e. taking into consideration the status of the various payment parts
amountTo	positiveDecimal	0..1	No	The maximum desired total payment amount. Please be aware that searches will be performed on the current total payment amount, i.e. taking into consideration the status of the various payment parts.
bonusFrom	nonNegativeInteger	0..1	Yes	The minimum desired total payment bonus amount. Please be aware that searches will be performed on the current total payment bonus amount, i.e. not taking into consideration the status of the various payment diffs.
bonusTo	nonNegativeInteger	0..1	Yes	The maximum desired total payment bonus amount. Please be aware that searches will be performed on the current total payment bonus amount, i.e. not taking into consideration the status of the various payment diffs.
frozen	boolean	0..1	No	The payment freeze status.
withMetaData	withMetaData	0..1	No	The desired meta data.
statusSet	paymentStatus	0..100	No	List of the statuses the payment must have.
statusNotSet	paymentStatus	0..100	No	List of the statuses the payment must not have

signing object

The signing URLs when a signing is required.



The URL's must not contain pipes ("|") since they will not work in redirects.

Component	Type	Occurs	Nillable?	Description
successUrl	string	1..1	No	The URL to return the customer to when a signing succeeds.
failUrl	string	1..1	No	The URL to return the customer to when a signing fails.

Simple Types...

Unknown macro: 'divbox'

sortAlternative

The sort columns available.

Value	Description
PAYMENT_ID	Sort the result on payment identity.
CUSTOMER_GOVERNMENT_ID	Sort the result on customer government identity.
CUSTOMER_NAME	Sort the result on customer name.
BOOKED_TIME	Sort the result on payment booking time.
MODIFIED_TIME	Sort the result on payment modification time.
FINALIZED_TIME	Sort the result on payment finalization time.
AMOUNT	Sort the result on total payment amount, taking into consideration the payment part status

sortOrder

How the search results should be ordered.

Contains elements as defined in the following table.

Component	Type	Occurs	Nillable?	Description
ascending	boolean	1..1	No	Whether or not the results are to be sorted in ascending order.
sortColumns	sortAlternative	1..*	No	On which columns, and in which order of importance, the result is to be sorted.

SortAlternativie

The sort columns available.

Value	Description
PAYMENT_ID	Sort the result on payment identity.
CUSTOMER_GOVERNMENT_ID	Sort the result on customer government identity.
CUSTOMER_NAME	Sort the result on customer name.
BOOKED_TIME	Sort the result on payment booking time.
MODIFIED_TIME	Sort the result on payment modification time.
FINALIZED_TIME	Sort the result on payment finalization time.
AMOUNT	Sort the result on total payment amount, taking into consideration the payment part status

specLine

The payment line (item) specification. These can be used to provide detailed information about the contents of the payment.

Contains elements as defined in the following table.

Component	Type	Occurs	Nillable?	Description
id	id	1..1	No	The line identity
artNo	string	1..1	No	Article number of the item. (Max 100 characters)
description	string	1..1	Yes	The item description.
quantity	decimal	1..1	No	The line quantity.
unitMeasure	string	1..1	No	
unitAmountWithoutVat	decimal	1..1	No	The unit amount without VAT.
vatPct	percent	1..1	No	The VAT percentage.
totalVatAmount	decimal	1..1	No	The total item VAT amount.
totalAmount	decimal	1..1	No	The total item amount, including VAT.
deliveryDate	date	0..1	No	Estimated delivery date

```
<paymentSpec>
    <specLines>
        <id>1</id>
        <artNo>NUT-001</artNo>
        <description>Nut (M8)</description>
        <quantity>10</quantity>
        <unitMeasure>st</unitMeasure>
        <unitAmountWithoutVat>0.80</unitAmountWithoutVat>
        <vatPct>25</vatPct>
        <totalVatAmount>2.00</totalVatAmount>
        <totalAmount>10.00</totalAmount>
            <deliveryDate>yyyyMMdd</deliveryDate>
    </specLines>
</paymentSpec>
```

webLink

Represents a link to be placed on a page. Usage (if appendPriceLast is `false`): `endUserDescription`. If appendPriceLast is `true`, the page that is linked is expected to show some information based on a particular amount, such as the price of a given product, and the URL is to be completed by appending the price of that product to the URL.
Contains elements as defined in the following table.

Component	Type	Occurs	Nillable?	Description
appendPriceLast	boolean	1..1	No	Whether or not the URL needs to be suffixed by an amount. Note: the amount is an integer! The web link URL " <code>http://site.com/cgi?param1=1&price=</code> " and an amount of SEK 999.90 would give the complete URL " <code>http://site.com/cgi?param1=1&price=1000</code> ".
endUserDescription	<code>nonEmptyString</code>	1..1	No	The link description. (<code>endUserDescription</code>)
url	<code>nonEmptyString</code>	1..1	No	The possibly incomplete URL to link to. See appendPriceLast for more details!

withMetaData

A set of meta data for searching.

Contains elements as defined in the following table.

Component	Type	Occurs	Nillable?	Description
withMetaDataKey	string	1..1	No	The desired meta data key.
withMetaDataValue	string	0..1	No	The desired meta data value.

Create part payment widget

Create a widget which displays the suggested part payment price on product pages.

You need to be able to configure the following:

1. **On/Off:**
Ability to turn feature on/off.
2. **Min.price:**
Only display the widget for products which price, incl. tax, is within the given range.
Where productPrice between Min.price and <maxLimit>50000.00</maxLimit>
Get maxLimit using api [getPaymentMethods](#)
3. **Payment method:**
You should be able to configure which payment method part payment data is based on.
Only show payment methods that are eligible for installment.
Where <specificType>PART_PAYMENT</specificType> or <specificType>REVOLVING_CREDIT</specificType>
Get specificType using api [getPaymentMethods](#)
4. **Months:**
Choose number of months for the selected payment method in step 3.
Get available annuity factors using api [getAnnuityFactors](#)

On product pages.

"Installment from 129 SEK - 12 month. [Read more...](#)"

- The payment method used for part payments include a min. max. amount on its own, which we also need to check before displaying the widget.
- Prices should be rounded to next integer.
- Read more.. can be a Pop-Up window populated using [getCostOfPurchaseHtml](#).

Customer data - Regular expressions

Note!

If you are using Resurs Checkout, note that the customer input is country specific for the test-/production account that you are using. If you are unsure of which country your account is setup for, please contact us.

GovernmentId

Legal

Country	Expression
Sweden	<code>^(16\d{2} 18\d{2} 19\d{2} 20\d{2} \d{2})(\d{2})(\d{2})(\d{4})\$</code>
Norway	<code>^([89]([-]?[0-9])\{8\})\$</code>
Finland	<code>^((\d{7})(\d)?\d)\$</code>
Denmark	

Natural

Country	Expression
Sweden	<code>^(18\d{2} 19\d{2} 20\d{2} \d{2})(0[1-9] 1[0-2])([0][1-9] 1[2][0-9] 3[0-1])(\d{4})\$</code>
Norway	<code>^([0][1-9] 1[2][0-9] 3[0-1])(0[1-9] 1[0-2])(\d{2})(\d)?\d\$</code>
Finland	<code>^((\d{6})(\d{3})([0-9]{3}))\$</code>
Denmark	<code>^((3[0-1]) ([1-2][0-9]) (0[1-9]))((1[0-2]) (0[1-9]))(\d{2})(\d)?\d\$</code>

Phone number

Country	Expression
Sweden	<code>^(0 \+46 0046)[-]?(200 20 70 73 76 74 [1-9][0-9]{0,2})([-]?)?{5,8}\$</code> <code>^(\+46 0046 0)[-]?{2}(200 20 70 73 76 74 4[0-5,7-9][0-9]? [1-3,5-9][0-9]{0,2})([-]?{0-9}){5,8}\$</code> - OBS. the double backslashes (\\\) are needed when using the reg.exp in JavaScript since JavaScript removes one.
Norway	<code>^(\+47 0047)?[-]?{2-9}([-]?{0-9}){7,7}\$</code>
Finland	<code>^((\+358 00358 0)[-]?([1-9][2-9] [1][0][1-9] 201 2021 [2][0][2][4-9] [2][0][3-8] 29 [3][0][1-9] 71 73 [7][5][0][0][3-9] 7[5][3][0][3-9] 7[5][3][2][3-9] 7[5][7][5][3-9] 7[5][9][8][3-9] 5[0][0-9]{0,2} 4[0-9]{1,3})([-]?)?{0-9}){3,10}\$</code>
Denmark	<code>^(\+45 0045)?[-]?{2-9}([-]?{0-9}){7,7}\$</code>

E-mail

Country	Expression	PHP preg_match example
All	<code>^[A-Za-z0-9!#%&'*+/=?^_`~-]+(\.[A-Za-z0-9!#%&'*+/=?^_`~-]+)*@([A-Za-z0-9]+)([.\.\-]?[a-zA-Z0-9]+)*\.([A-Za-z]\{2,\})\$</code>	<pre>preg_match('@^([A-Za-z0-9!#%&'*+/=?^_`~-]+)(\.[A-Za-z0-9!#%&'*+/=?^_`~-]+)*@([A-Za-z0-9]+)([.\.\-]?[a-zA-Z0-9]+)*\.([A-Za-z]\{2,\})\$@', \$emailAddress);</pre>

Card number

Country	Expression
All	<code>^([1-9][0-9]{3} []{0,1}[0-9]{4} []{0,1}[0-9]{4} []{0,1}[0-9]{4})\$</code>

Rounding

! Please, do not round numbers on your side. If calculations result in more than 2 decimals, please supply all decimals in the invocations of our service.

Later, the API will return five (5) decimals in the `getPayment-response` - no matter how many decimals where sent in earlier.

The rounding of purchase and VAT values is carried out by Resurs Bank's web services, based on a set of rules, and thus is not mandatory.

We always respect the price values (except VAT values) as total amount in each row and the payment's total amount, sent to us by the representative. However, there has to be a consistency between the received values and the ones we calculate ourselves taking into account of the rounding rules.

The service compensates for errors less than, or equal to, 0.05 currency units (i.e. no matter which currency).

This error threshold must be respected in both the "`specLines`" values and in the "`totalAmount`" and "`totalVatAmount`" values within the "`paymentSpec`" XML tag, [see paymentSpec](#).

To summarize:

1. Representatives should send in the price and VAT values as is, without rounding, and let us calculate them.
Precision is not an issue as our routines are capable of dealing with a great number of decimals.

2. If you perform rounding calculations before invoking the e-Commerce service, it may produce an error if the error margin is exceeded.
We will perform rounding calculations in any case as described below.

The web services rounding rules are:

- **Individual items***:

Given Item Net Price (without VAT) = Given row value / ((1 + VAT percent / 100) * Quantity)
Given Total VAT Amount = (VAT percent / 100) * Total !Calculated! Item's Net Price

- **Total Amount:**

Given Total Price Amount with VAT = (Given Total Individual Price Amount)
Given Total VAT Amount = (Given Total Individual VAT Amount)

* Refers to each "specLines" entry.

Example:

One of the most common sources of [rounding] errors is the representative adding the sales price with VAT and sending that sum to us.

A representative sells the following item: 1" x 4" building board, price: 15.71 SEK (includes VAT).

Upon receiving this value we recalculate it to a net price: $15,71 / 1,25 = 12,568$.

Now, say that the following is set within one "specLines" XML tag:

Quantity	1000
Net Price per Item	12,57 (!= 12,568)
VAT	25%
Total VAT	3142,00
Total Price including VAT	15710,00

Upon recalculation:

* $15710,00 / (1,25 * 1000) = 12,568$ (!= 12,57)

* $0,25 * 1000 * 12,568 = 3142,00 = 3142,00$

The difference between the received and calculated net price is: $12,57 - 12,568 = 0,002$.

This difference, being within the error margin, leads to 12,568 being used instead of 12,57.

Recognized metadata

What is metadata?

In short, it is key/value data piggybacked on the payment.

Read here: [Associated metadata](#)

Recognized keys and meaning

Generally we don't look at the metadata. Listed below are the exceptions to that rule.

Key name	Expected format	Description
invoiceExtrRef	String. 46 chars.	In the case that the payment generates invoices and credit notes this value will be printed as 'Your reference', for example the sales person responsible. Mostly for company invoices.
CustomerId	String. 20 chars.	In the case that the payment generates invoices and credit notes this value will be printed as 'Customer Id'

1 / 1

 **Resurs
Bank**

Faktura

Fakturaadress
Vincent Williamsson Alexandersson
Glassgatan 15
41655 Göteborg

Leveransadress
Vincent Williamsson Alexandersson
Glassgatan 15
41655 Göteborg

Fakturanummer 100171 **Kundnummer** Max 20 tgn **Fakturadatum** 2016-10-19 **Er ref / projektnummer** Max 46 tgn

Betalningsreferens/OCR 8538792891 **Betalningsvillkor** 41 dagar **Förfallodatum** 2016-11-30

Betalnummer 1001083 **Dräjämålränta** 20,00 %

Art. Nr.	Beskrivning	Antal	A-pris inkl. moms	Moms	Belopp
0	Order line:1	1	625,00 kr	25 %	625,00 kr
2	Order line:2	1	12,50 kr	25 %	12,50 kr
80	Discount	1	-6,25 kr	25 %	-6,25 kr
90	Shipping & Handling	1	5,00 kr	0 %	5,00 kr
			Totalt SEK exkl. moms		510,00 kr
			Momsbelopp		126,25 kr
			Totalt SEK inkl. moms		636,25 kr

från 100,00 kr/mån*

* Genom samarbete med Resurs Bank erbjuder vi möjligheten till förlängd kredit och räntefri delbetalning. [Läs mer här](#) eller på nästa sida.

Payment providers in simplified flow

Field validation with PAYMENT_PROVIDER

In normal cases, for Resurs "internal" payment methods, fields that is shown in the store during the checkout process should be validated against [our regular expression schemes](#). However, there is a condition for payment methods with the type PAYMENT_PROVIDER (PSP) that does not need such validation: government id/SSN. Since PSP redirects customers to external partners the government id does not need to forward those fields. As of EComPHP (example) v1.3.12 this behaviour is supported by either sending an empty string (or null) as government id - or a normal value. Both will be properly treated: If it is empty, the variable will be removed from the payload. If not, it will be passed through (even if it might not be handled by any partner).

Our WooCommerce plugin uses a (deprecated) method to load and show necessary customer fields that validates both "internal" payment methods and external PSP methods. However, it is only PSP that allows empty variables. This has been solved with a method called `getCanSkipGovernmentIdValidation`:

```
if ($fieldName == 'applicant-government-id' && empty($_REQUEST[$fieldName]) && $flow->getCanSkipGovernmentIdValidation()) {
    continue;
}
```

If the boolean of this method is true, the module knows it doesn't have to validate the content of the field, so it can be skipped. By doing this, the module does not need to be rewritten due to the structure changes (even if there are plans of making those parts independent).

Permissions and passwords - Platform access

 This page only talks about web service security, not [Payment administration GUI](#) security.

Authentication

The Resurs e-Commerce service is protected by the simple [http basic authentication mechanism](#). Behind a TLS/SSL protected channel this is an acceptable level of security.

We've chosen this solution in favour of the more complex [WS-Security](#) which still have some compatibility issues. With our solution the authorization is configured in a layer separate from the web service client.

Please, **enable preemptive authentication** to avoid unnecessary roundtrips (request, response). The standard mode in most HTTP client implementations is to try accessing the HTTP resource, and first when confronted with a 401, redo the request *with* credentials. This double roundtrip is a waste of precious milliseconds which risks make your webshop seem sluggish.

Flow Chart Library

Here are all the flow charts for the e-Commerce. Flow charts are a perfect way to get a better understanding on how the e-Commerce works, how to implement and consume its services.

Web Shop Flow

The flow over the Web Shop part from when a customer enters the shop to checkout

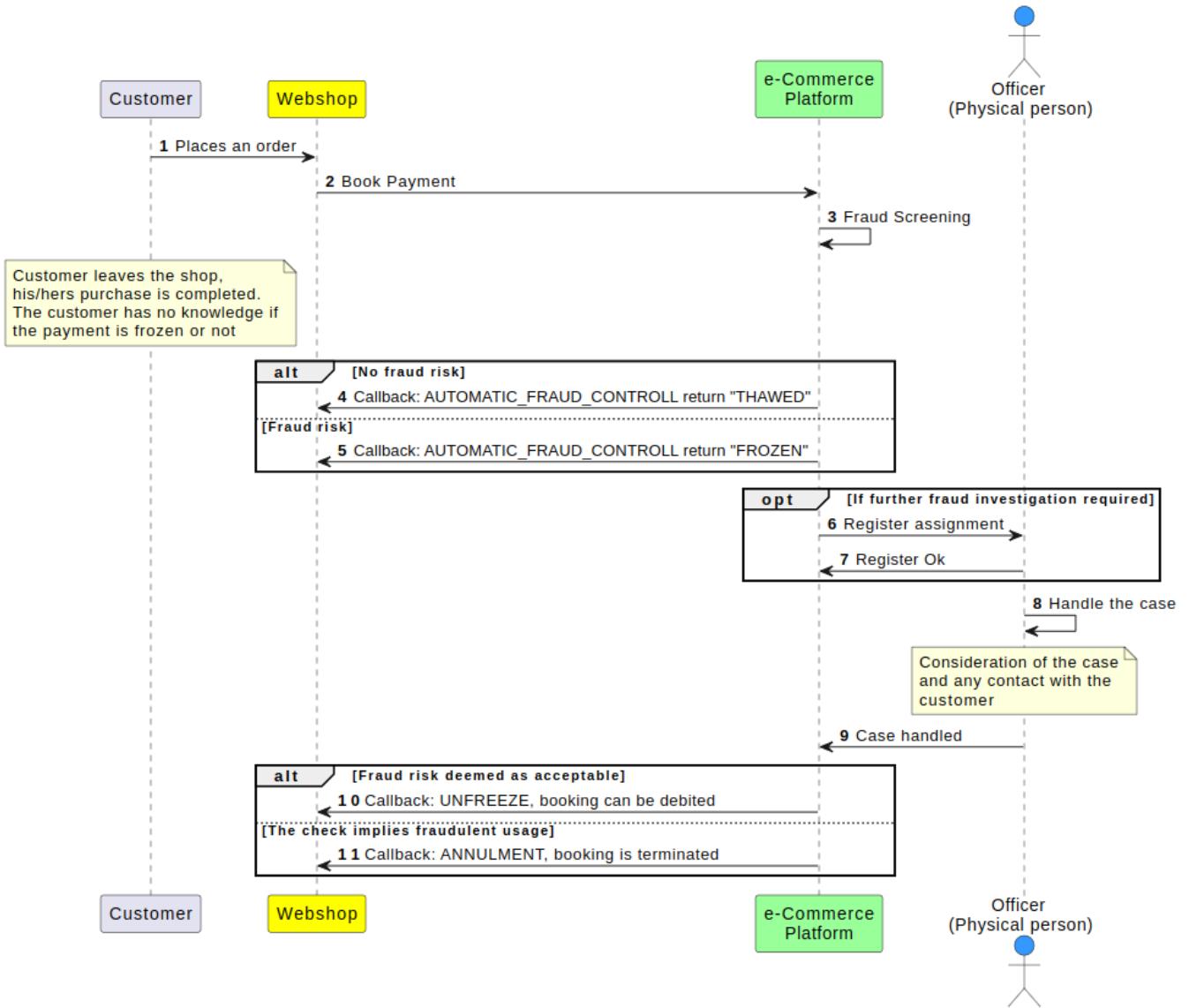
 Unknown macro: 'plantuml'

Fraud Control

The fraud control is made when the bookPayment is called from the shop flow.
Book Payment is called when the customer in the web shop completes the check out.

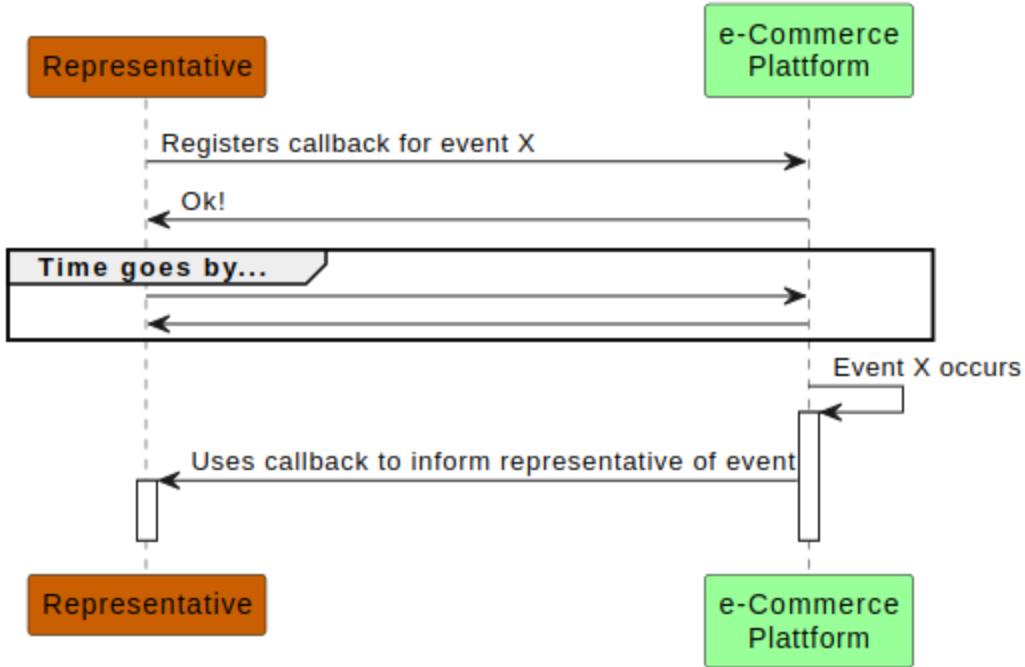


Agent represent both agent and customer in this flow



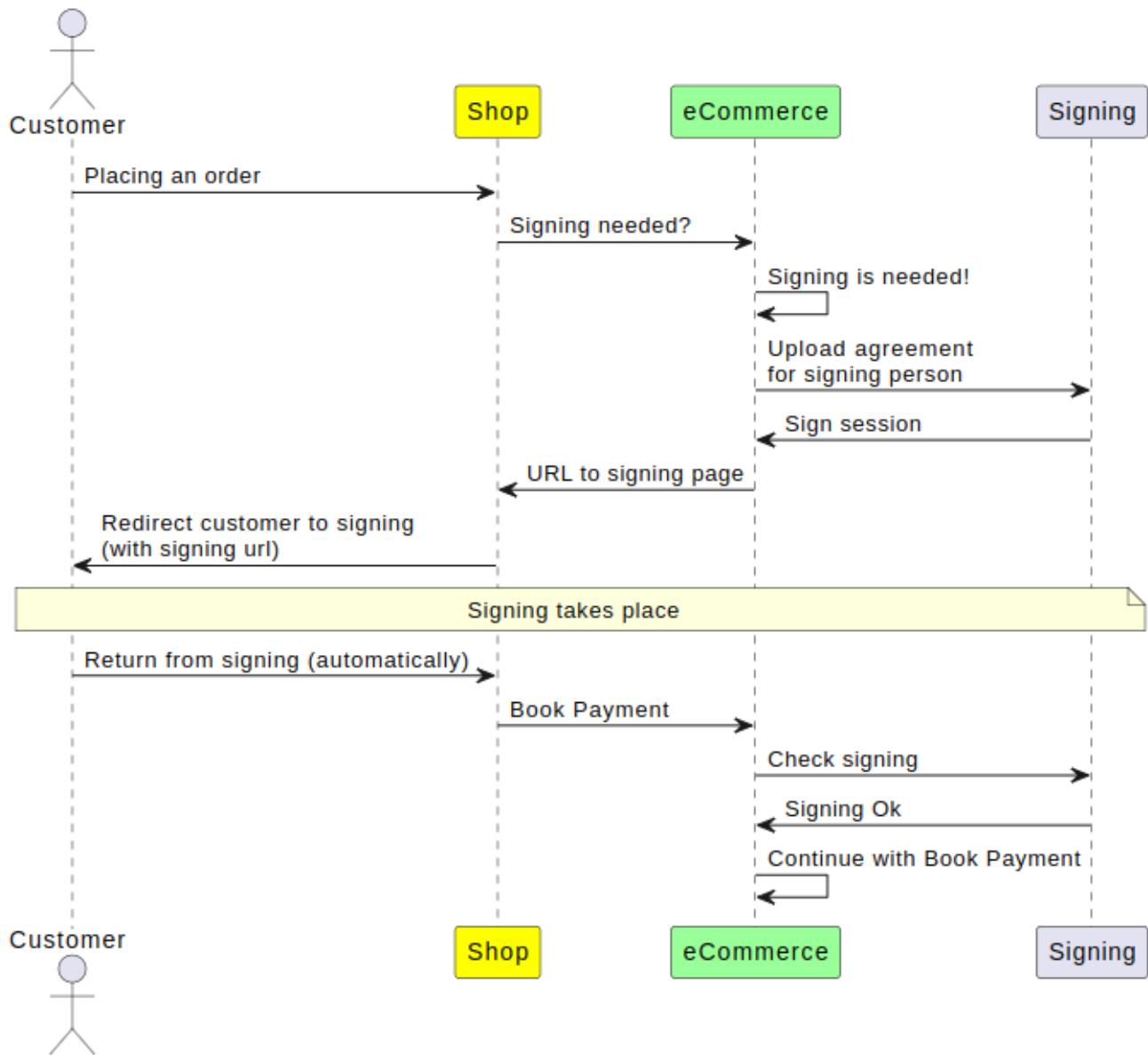
Callbacks

Registration on callbacks is made from the [Configuration Service](#), more information about our callbacks is [here](#)

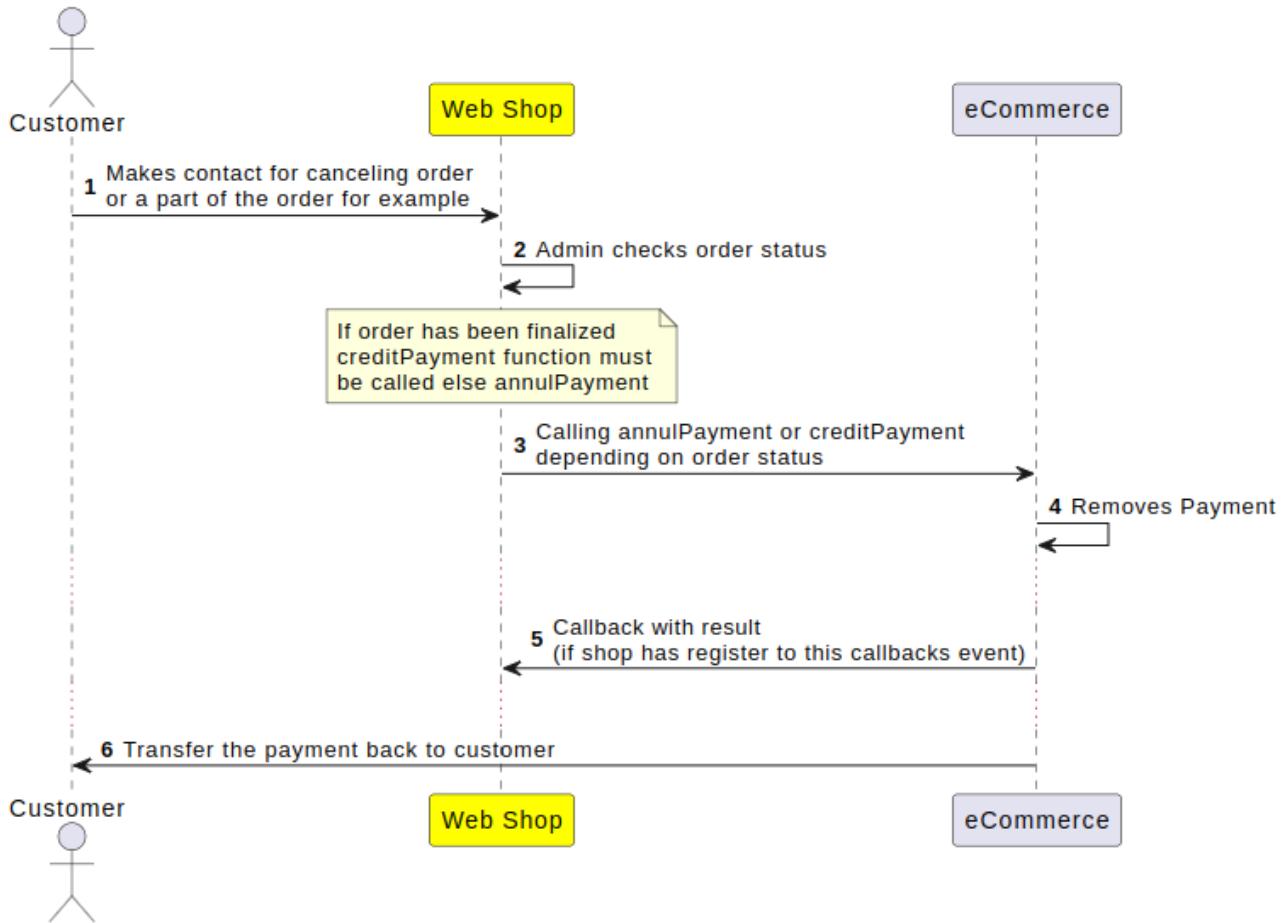


Signing

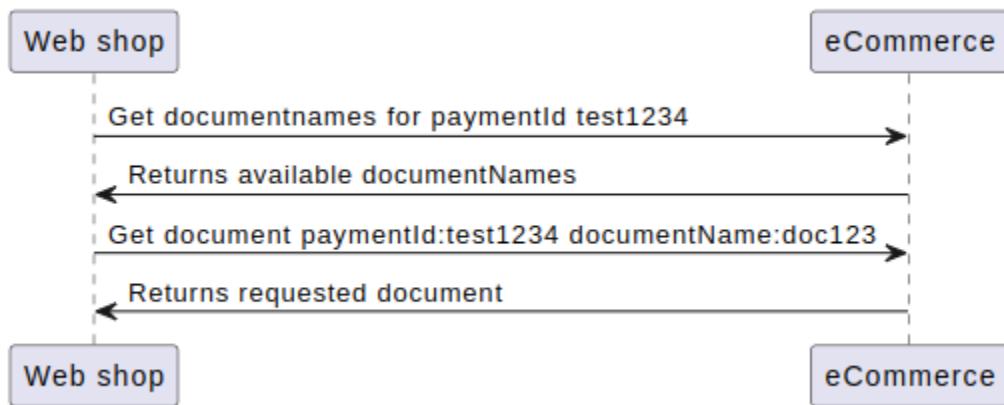
Signing of a payment



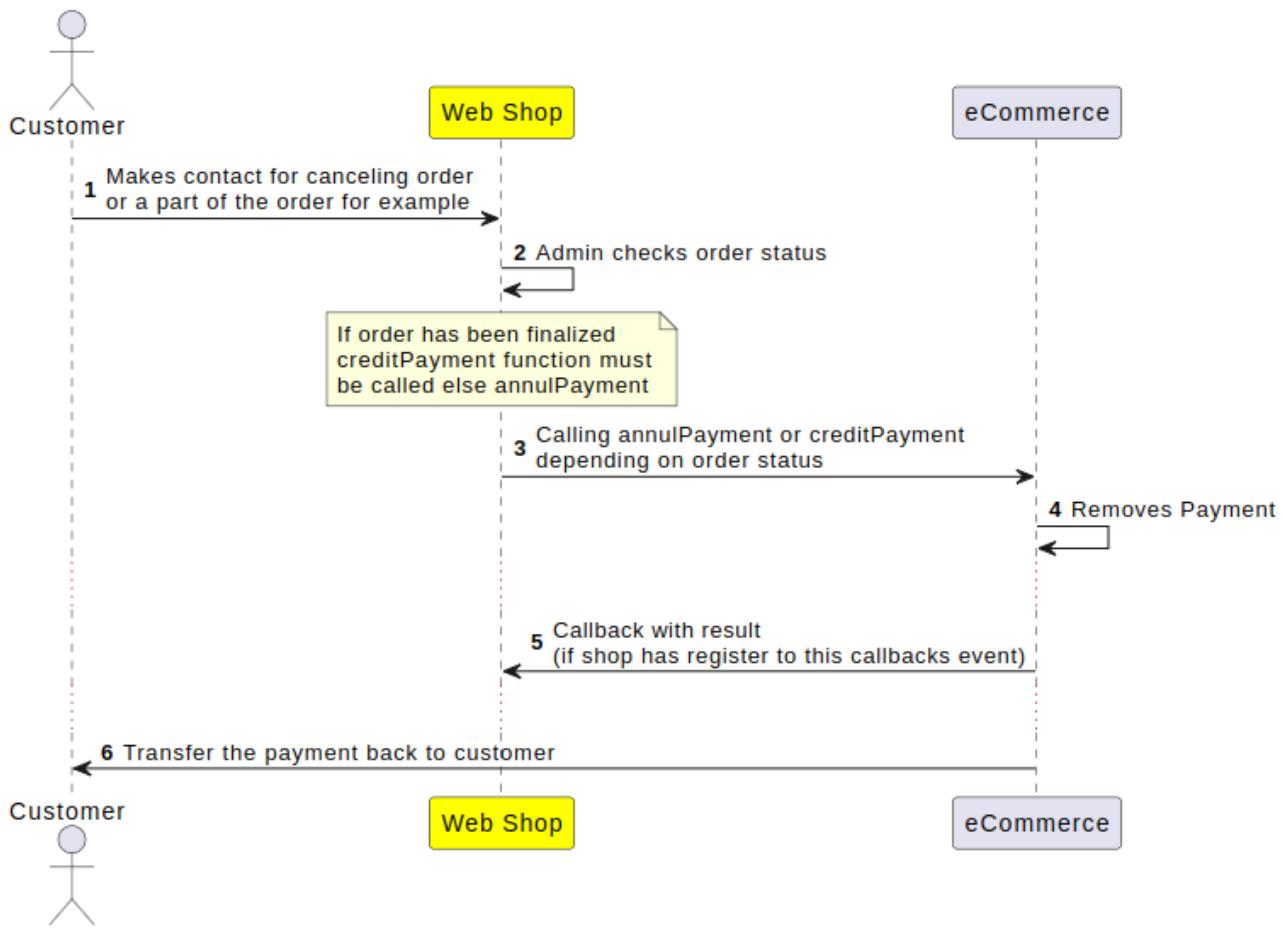
Annulment & Crediting



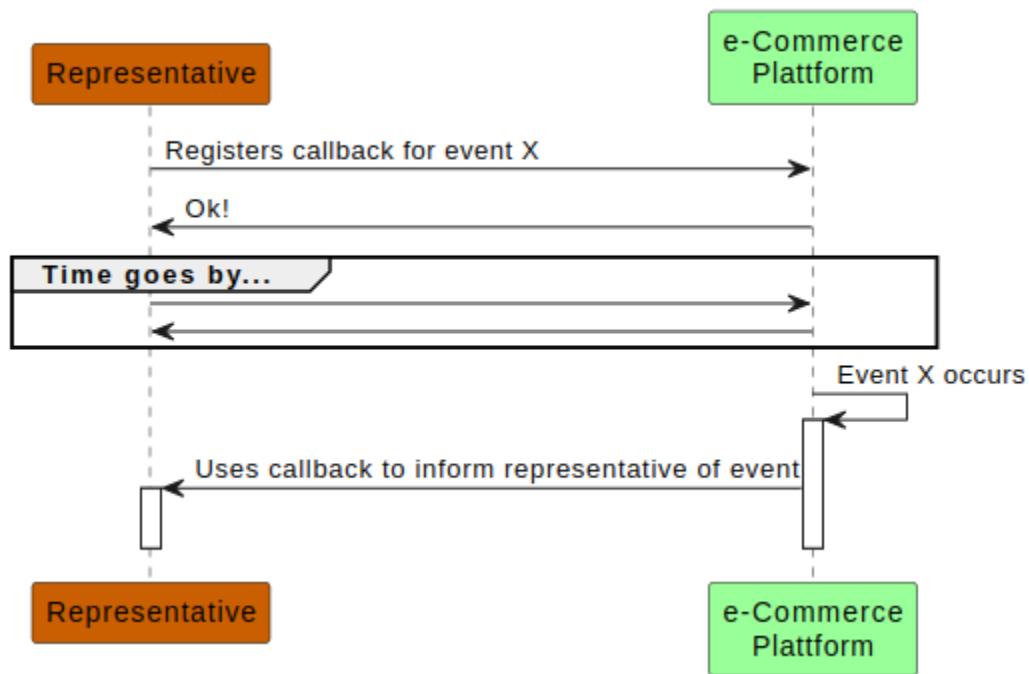
Get documentNames & document



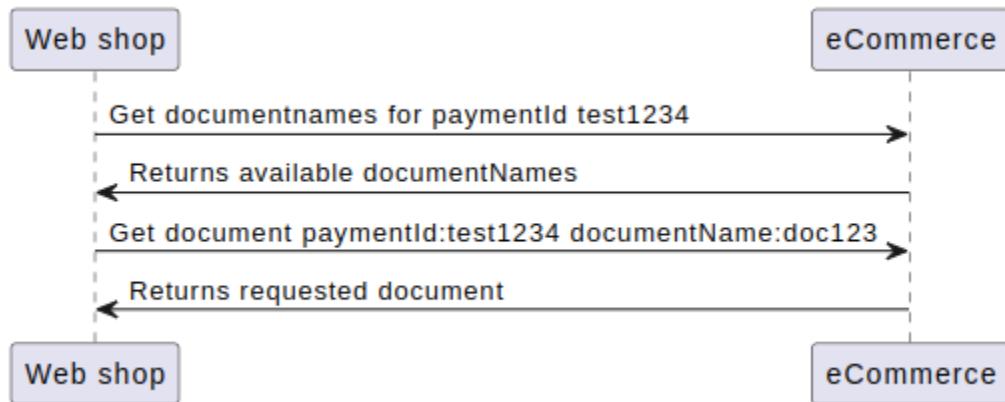
Annullment & Crediting Chart



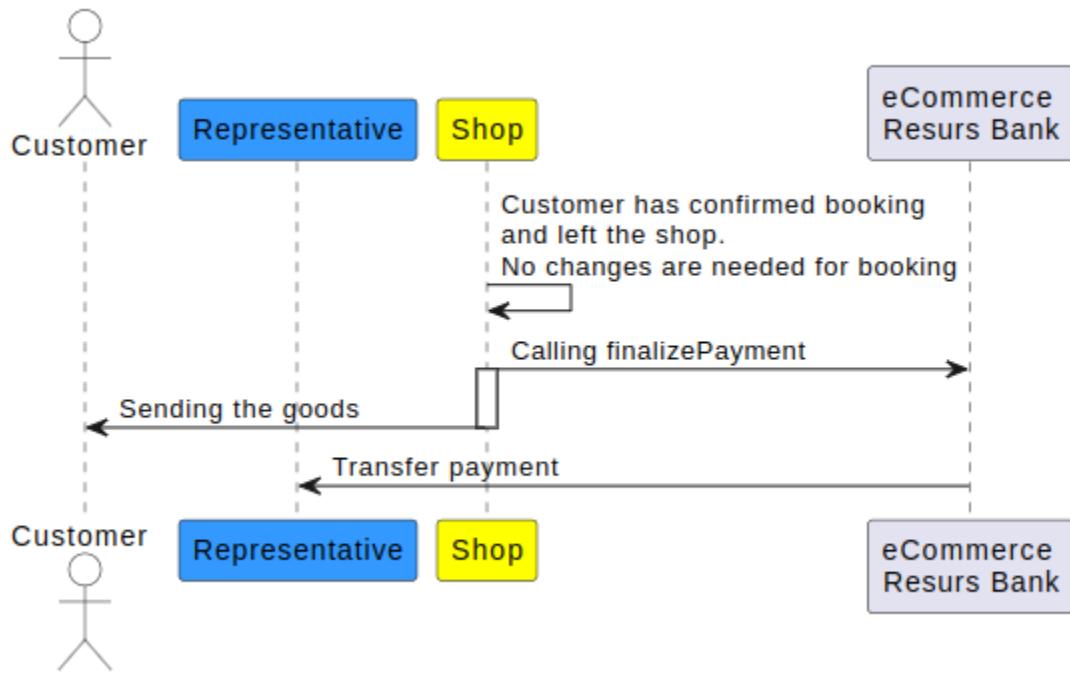
Callback Overall Flow Chart



DocumentNames and Document Flow chart

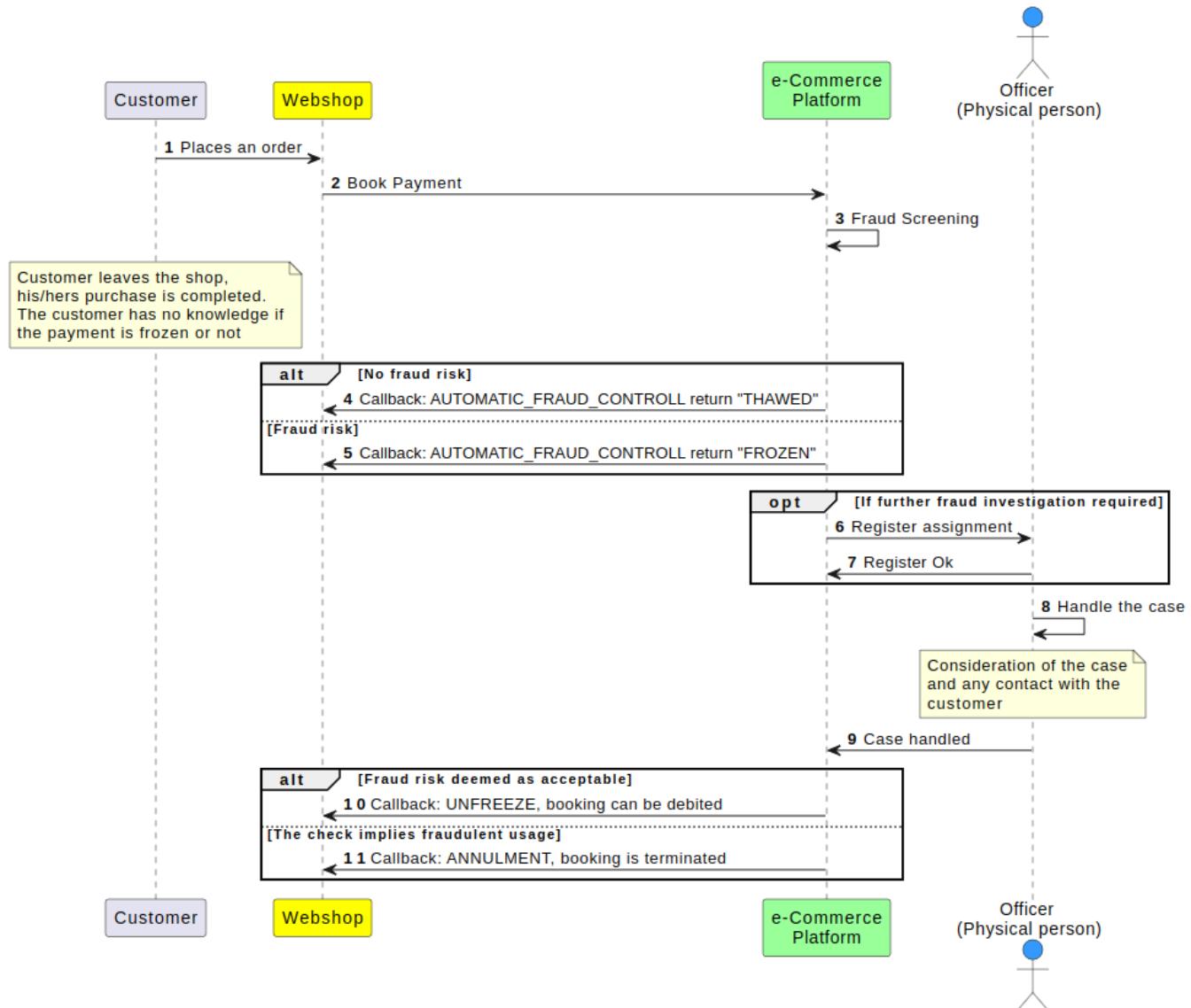


Finalize Payment Chart

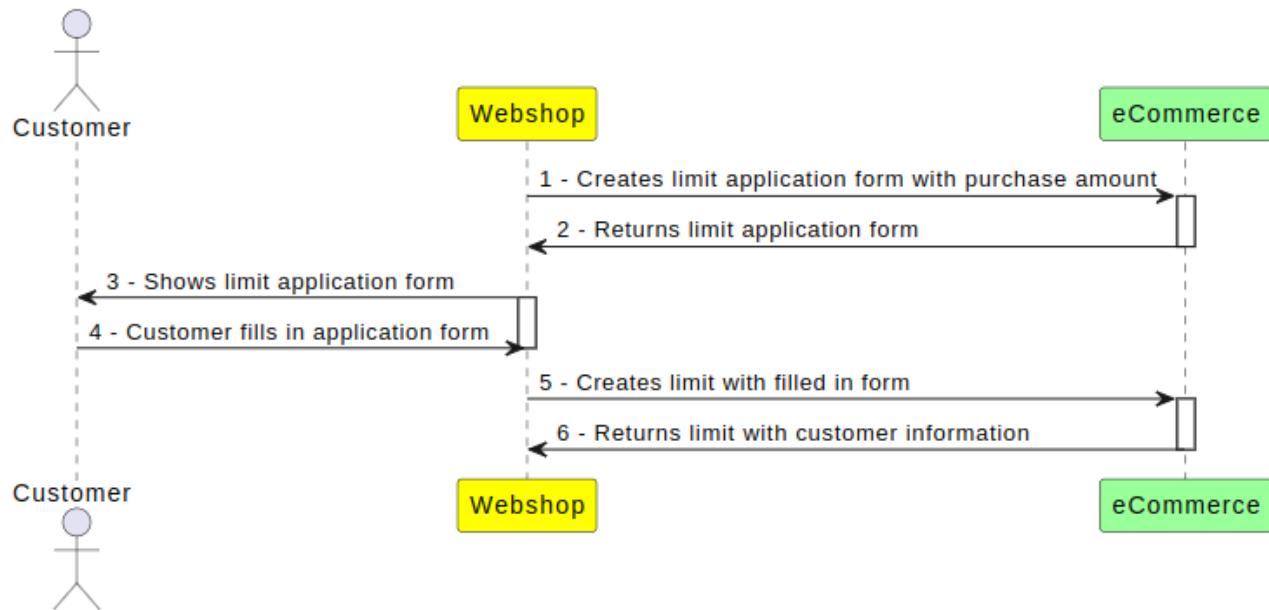


Fraud Screening Chart

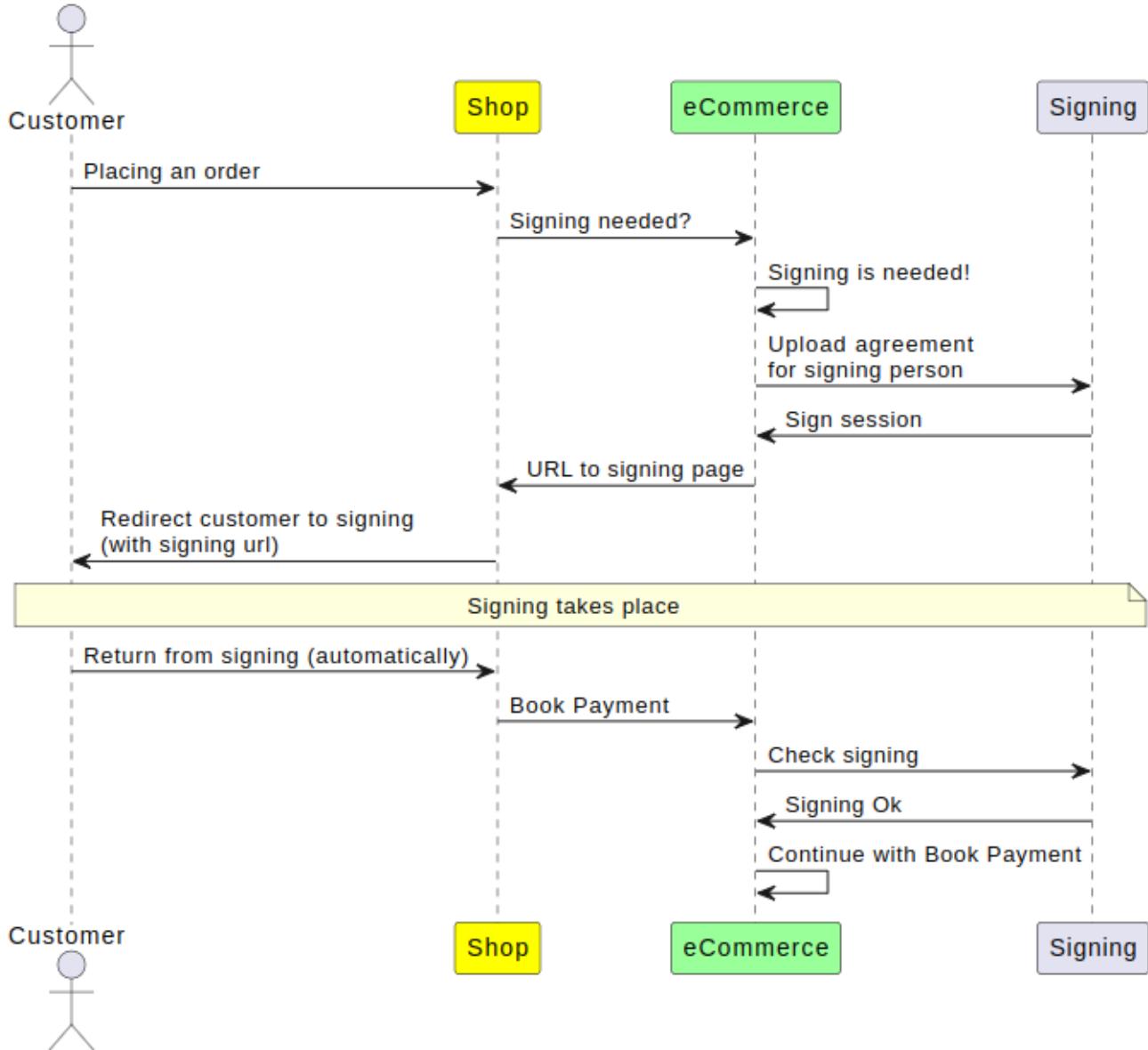
 Agent represent both agent and customer in this flow



Limit Application Form Chart

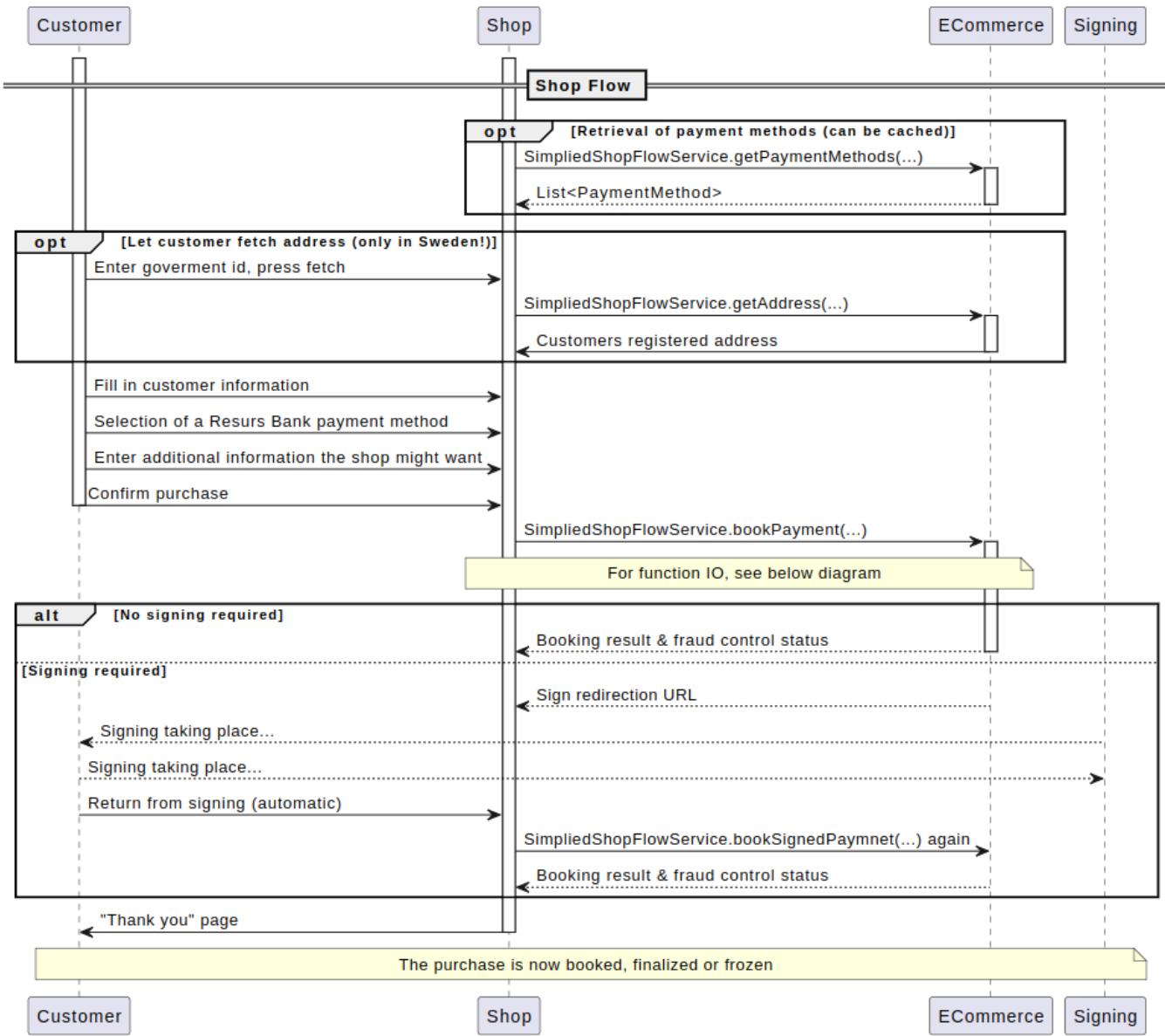


Signing Overall Flow Chart



Simplified Shop Flow Chart

For the IO and description, see the following page. [Simplified Flow API](#)



Gift Card Service

The Gift Card Service provides services for gift card related operations such as loading and canceling a gift card.

The examples below demonstrates how to interact with the Gift Card Service. For a full description of the API, please refer to the [Gift Card Service](#) manual.

Giftcard Test: <https://test.resurs.com/ws-22/giftcard/GiftCardService>

Giftcard Prod: <https://ws.butiksservice.resurs.com/ws-22/giftcard/GiftCardService>

Basic Auth:

Username: User#StoreNbr#SE

Pw: password set for user

1. Loading a card with a specified amount
2. Canceling a card, making it non-usuable
3. Getting a card's balance
4. Annul a load

1. Loading a card with a specified amount

This example shows how to load a card with a specified amount. To load a card you need to specify two parameters, *cardNumber* and *amount*.

The *cardNumber* parameters is the card's identity which is typically found on the card.

The *amount* parameters specifies the amount.

Load request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:giftcard="http://butiksservice.resurs.com/msg/giftcard">
  <soapenv:Header/>
  <soapenv:Body>
    <giftcard:load>
      <cardNumber>1234567890123456</cardNumber>
      <amount>200</amount>
    </giftcard:load>
  </soapenv:Body>
</soapenv:Envelope>
```

If the request was successful you should receive a response similar to the one below:

Load response

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns3:loadResponse xmlns:ns3="http://butiksservice.resurs.com/msg/giftcard" xmlns:ns2="http://butiksservice.resurs.com/msg/exception">
      <loadResult>
        <currentAmount>200.00</currentAmount>
        <authorizationId>135246</authorizationId>
        <expireDate>2015-10-04T00:00:00+02:00</expireDate>
      </loadResult>
    </ns3:loadResponse>
  </soap:Body>
</soap:Envelope>
```

The response contains the card's current amount, the card's expire date and an authorization id. The authorization id can be used annul the load.

It's a good idea to store the authorizationId to a persistent storage for later retrieval if needed.

2. Canceling a card, making it non-usuble

This example shows how to cancel a card. After a successful call to the *cancel* operation, the card is not usable anymore.

The *cancel* operation takes one parameter, *cardNumber*, which specifies the card that should be cancelled.

Cancel request

```

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:giftcard="http://butiksservice.resurs.com/msg/giftcard">
    <soapenv:Header/>
    <soapenv:Body>
        <giftcard:cancel>
            <cardNumber>1234567890123456</cardNumber>
        </giftcard:cancel>
    </soapenv:Body>
</soapenv:Envelope>

```

If the request was successful you should receive a response similar to the one below:

Cancel reposonse

```

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body>
        <ns3:cancelResponse xmlns:ns3="http://butiksservice.resurs.com/msg/giftcard" xmlns:ns2="http://butiksservice.resurs.com/msg/exception">
            <cancelResult>
                <authorizationId>224506000</authorizationId>
            </cancelResult>
        </ns3:cancelResponse>
    </soap:Body>
</soap:Envelope>

```

The response contains an authorization id for the cancellation.

3. Getting a card's balance

This example shows how to read out the card's current balance. The operation will also return the card's expire date.

The *getBalance* operation takes one parameter, *cardNumber*, which specifies the card.

Get Balance request

```

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:giftcard="http://butiksservice.resurs.com/msg/giftcard">
    <soapenv:Header/>
    <soapenv:Body>
        <giftcard:getBalance>
            <cardNumber>1234567890123456</cardNumber>
        </giftcard:getBalance>
    </soapenv:Body>
</soapenv:Envelope>

```

If the request was successful you should receive a response similar to the one below:

getBalance response

```

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body>
        <ns2:getBalanceResponse xmlns:ns3="http://butiksservice.resurs.com/msg/exception" xmlns:ns2="http://butiksservice.resurs.com/msg/giftcard">
            <balanceResult>
                <balance>400.00</balance>
                <expireDate>2015-08-26T00:00:00+02:00</expireDate>
            </balanceResult>
        </ns2:getBalanceResponse>
    </soap:Body>
</soap:Envelope>

```

The response contains the card's balance and expire date. The expire date is formatted according to ISO 8601.

4. Annul a load

This example shows how to annul a load. Two parameters is required to annul a load, *cardNumber* and *authorizationId*. *AuthorizationId* should be the authorization id you received from a *load* request.

Annul request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:giftcard="http://butiksservice.resurs.com/msg/giftcard">
  <soapenv:Header/>
  <soapenv:Body>
    <giftcard:annul>
      <cardNumber>1234567890123456</cardNumber>
      <authorizationId>135246</authorizationId>
    </giftcard:annul>
  </soapenv:Body>
</soapenv:Envelope>
```

If the request was successful you should receive a response similar to the one below:

Annul response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:giftcard="http://butiksservice.resurs.com/msg/giftcard">
  <soapenv:Header/>
  <soapenv:Body>
    <giftcard:annulResponse>
      <annulResult>
        <authorizationId>1234567</authorizationId>
      </annulResult>
    </giftcard:annulResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

The response contains an authorization id for the annul.

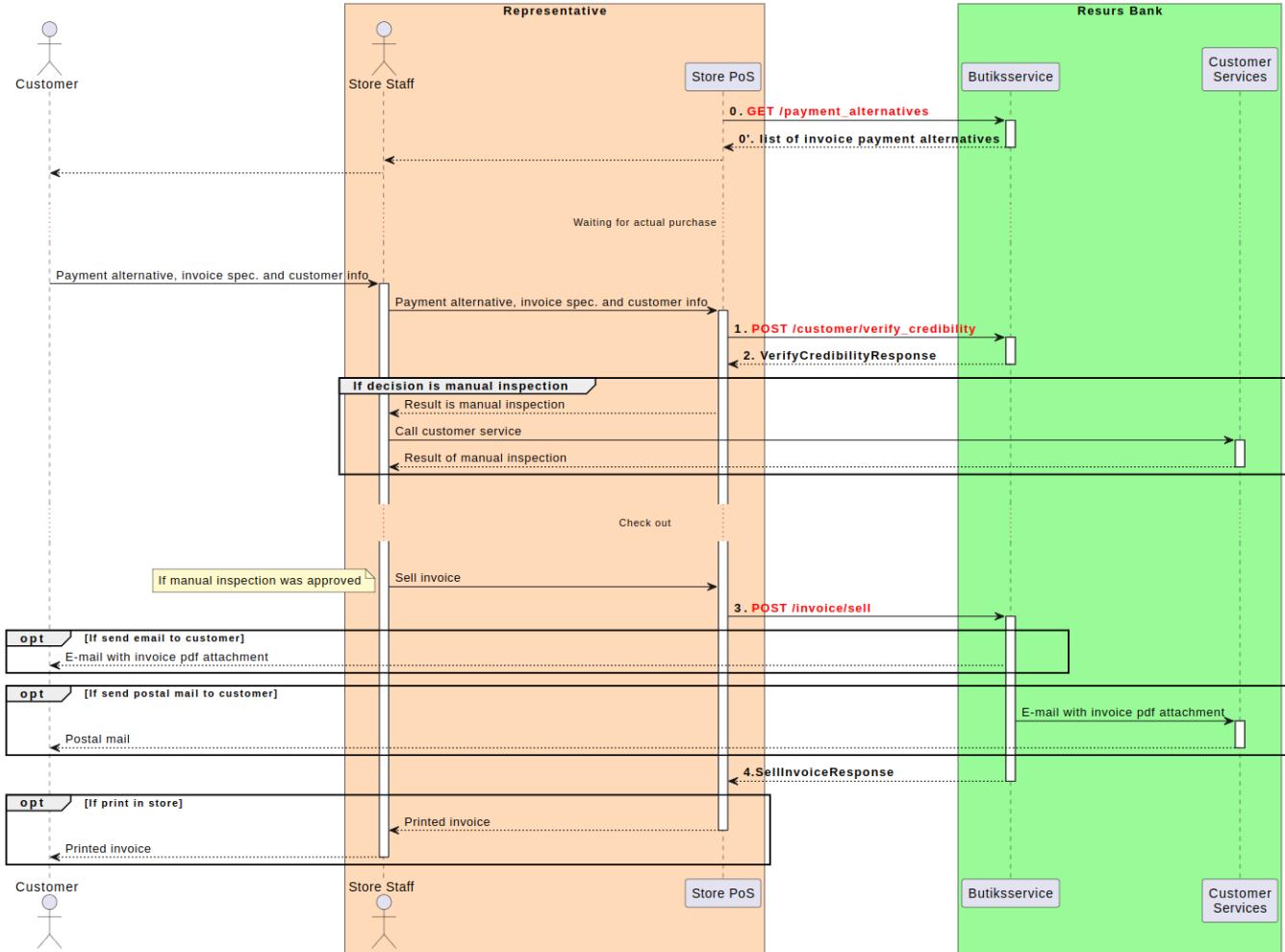
Invoice Service 2.0

The invoice web service offers an API to sell and credit invoice payments from Resurs Bank.

Authentication

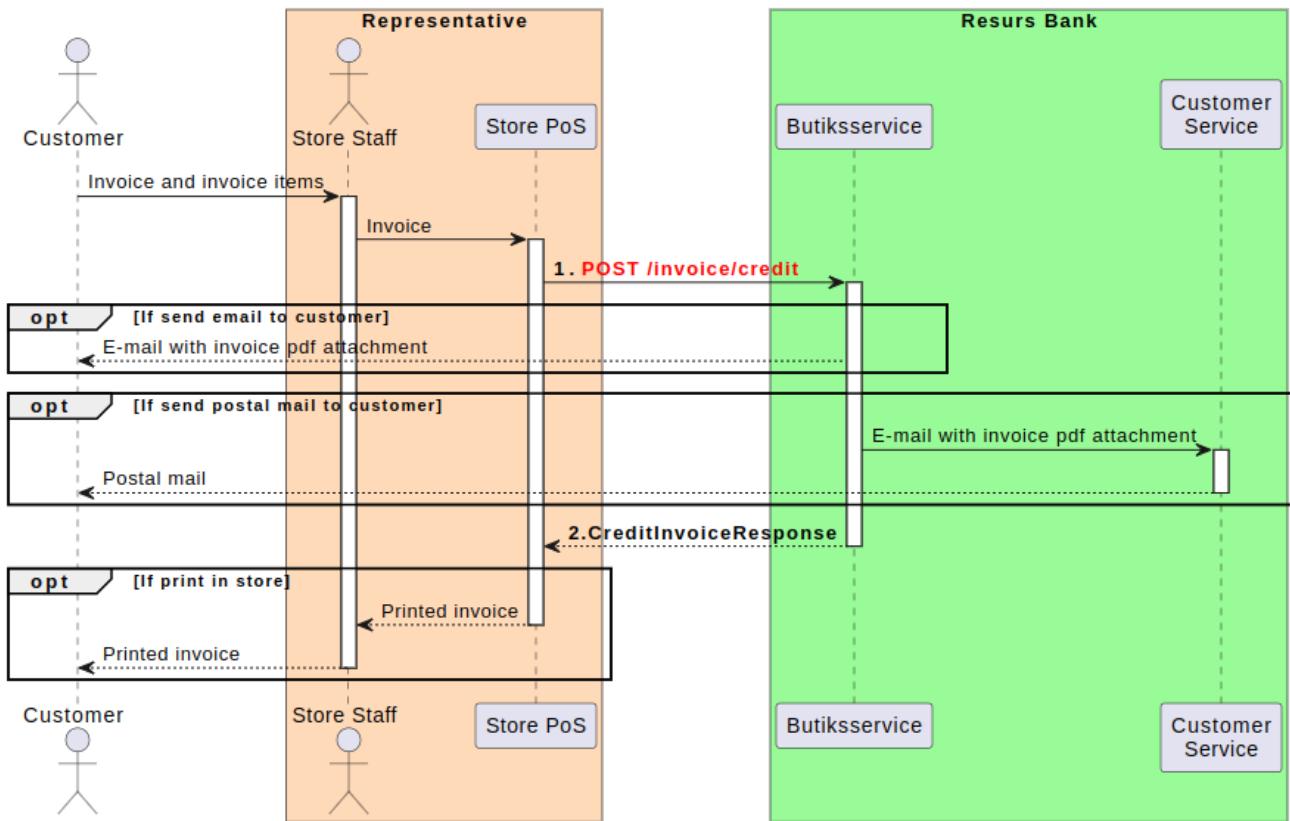
The Resurs shop service is protected by the simple [http basic authentication mechanism](#). Behind a TLS/SSL protected channel this can be considered secure.

Credit check (in infodisk) and purchase at checkout



Credit invoice

Credit invoice



API Reference

- [Methods](#)
 - [List payment alternatives](#)
 - [Verify credit](#)
 - [Sell invoice](#)
 - [Credit invoice](#)
 - [Configure invoice data](#)
 - [Retrieve invoice data configuration](#)
- [Common API-types](#)
 - [Payment alternative](#)
 - [Contact information](#)
 - [Invoice line](#)
 - [Pdf](#)
 - [Error message](#)

 The API is accessible in test and production via the following endpoints:

Test: <https://test.resurs.com/invoice-ws-v2>

Production: <https://ws.butiksservice.resurs.com/invoice-v2>

Methods

List payment alternatives

Use this to get a list of all available payment alternatives. This should be called before making a purchase.

 The payment alternatives can change (the id, description, etc.) and should not be cached.

Operation

```
GET /payment_alternatives?amount={amount}
```

Parameters

Name	Type	Mandatory	Constraints	Description
amount	decimal		Positive number	If specified, only payment alternatives matching the amount will be returned.

Response

Name	Type	Mandatory	Constraints	Description
payment_alternatives	array of payment_alternative			

Response sample

```
{  
    "payment_alternatives": [  
        {  
            "id": "AF682069",  
            "description": "SOVA FAKTURA TEST",  
            "customer_type": "NATURAL",  
            "min_amount": 1000,  
            "max_amount": 50000  
        },  
        {  
            "id": "AF682070",  
            "description": "SOVA FAKTURA TEST",  
            "customer_type": "NATURAL",  
            "min_amount": 1000,  
            "max_amount": 50000  
        }  
    ]  
}
```

```

        "id": "LG686069",
        "description": "TEST FAKTURA MED DELBET",
        "customer_type": "NATURAL",
        "min_amount": 10,
        "max_amount": 50000
    },
    {
        "id": "NZ690101",
        "description": "TEST FÖRETAGSFÄKTURA MT",
        "customer_type": "LEGAL",
        "min_amount": 1,
        "max_amount": 50000
    }
]
}

```

Verify credit

Use this to verify the credit worthiness of an applicant. The request should be made before submitting a sell request in order to verify that the customer is ok. If the returned decision is "APPROVED" the purchase flow can be continued and a subsequent sell request can be made. Making a sell request when verify credibility returns a decision of anything else than APPROVED will result in an error. If the returned decision is "MANUAL_INSPECTION" the PoS-system should inform the store staff to contact Resurs Bank customer service for inspection of the application. If the application is approved this will be reflected on further calls to verify credibility.

Operation

```
POST /customer/verify_credibility
```

Request

Name	Type	Mandatory	Constraints	Description
payment_alternative_id	string	x		The id of payment_alternative
requested_amount	decimal	x	Positive number	The requested amount of the credit application
government_id	string	x	A valid government id	The government id of the liable payer. Either an organisation number or a national identification number. Use www.personnummer.nu to generate numbers. Please note that a date in the second, or higher, quarter shall be used due to test filter characteristics.
contact_information	contact_information	x		The contact persons information

Request sample

```
{
    "payment_alternative_id": "AF682069",
    "government_id": "19910125-3905",
    "contact_information": {
        "email": "test@resurs.se",
        "phone": "0708-123456",
        "national_identification_number": "19910125-3905"
    },
    "requested_amount": 10000
}
```

Response

Name	Type	Mandatory	Constraints	Description
decision	string	x	"APPROVED", "MANUAL_INSPECTION", "REJECTED"	Decision of the evaluation of the credit worthiness of the applicant. An invoice can only be sold if the decision is "APPROVED".
approved_a	decimal	x	Positive number.	Will never be greater than requested amount, and always zero if decision is anything

mount				else than APPROVED
-------	--	--	--	--------------------

Response sample

```
{
  "decision": "APPROVED",
  "approved_amount": 10000
}
```

Sell invoice

Use this to make a purchase in the form of an invoice. In addition to returning the invoice-PDF in the response the invoice can also be delivered via e-mail (as attachment) or postal mail. The delivery option is configured by setting the invoice_delivery_option.

 Do not call sell invoice if verify credibility returned anything else than APPROVED.

 The amount fields on the invoice lines are not validated against total amount.

Operation

```
POST /invoice/sell
```

Request

Name	Type	Mandatory	Constraints	Description
external_referen ce	string	x	Unique per chain. a-z, A-Z, 0-9, "_" and "-" are allowed.	Payment reference. This identifier will be reported back in economic reports. Use an identifier that can easily be traced back to your system, such as order id.
payment_alternat ive_id	string	x		The id of payment_alternative
governm ent_id	string	x	A valid government id	The government id of the liable payer. Either an organisation number or a national identification number
contact_i nformati on	contact _inform ation	x		The contact persons information
invoice_l ines	array of invo ce_line	x	Must contain at least one element	
total_am ount_wit hout_vat	decimal	x		Total amount without VAT
total_vat _amount	decimal	x		Total VAT amount
total_am ount_wit h_vat	decimal	x		Total amount with VAT
invoice_n umber	long	x	Positive number	
invoice_d elivery_o ption	string	x	"POSTAL", "EMAIL", "NONE"	This determines how the invoice will be delivered to the customer. The "POSTAL"-option means that the invoice will be mailed by postal mail to the customer's address. The "EMAIL"-option means that the invoice will be delivered by e-mail to the supplied customer email. The "NONE"-option will not do any of the above. This option does not affect the response, which always contains the invoice PDF if the call was successful.
invoice_e xtra_da ta	string			Free-text field on the invoice

Request sample

```
{  
    "external_reference" : "79faa780-7b8e-11e4-82f8-0800200c9a66",  
    "payment_alternative_id" : "NZ690101",  
    "government_id": "19910125-3905",  
    "contact_information": {  
        "email": "test@resurs.se",  
        "phone": "0708-123456",  
        "national_identification_number": "19910125-3905"  
    },  
    "invoice_lines" : [  
        {  
            "article_id" : "sk-1",  
            "article_number" : "1",  
            "description" : "Lång smal skruf",  
            "quantity" : 4,  
            "unit_measure" : "paket",  
            "unit_amount_without_vat" : 250.00,  
            "vat_percentage" : 25,  
            "total_vat_amount" : 252,  
            "total_amount_with_vat" : 1252  
        },  
        {  
            "article_id" : "bit-2",  
            "article_number" : "2",  
            "description" : "Blandade bits för hemmasnickaren",  
            "quantity" : 1,  
            "unit_measure" : "st",  
            "unit_amount_without_vat" : 103.92,  
            "vat_percentage" : 25,  
            "total_vat_amount" : 25.98,  
            "total_amount_with_vat" : 129.9  
        }  
    ],  
    "total_amount_without_vat" : 1103.92,  
    "total_vat_amount" : 275.98,  
    "total_amount_with_vat" : 1379.9,  
    "invoice_number" : "103",  
    "invoice_delivery_option" : "NONE",  
    "invoice_extra_data" : "Data från kunden"  
}
```

Response

Name	Type	Mandatory	Constraints	Description
pdf	Pdf	x		Invoice PDF encoded in base 64

Sample response

```
{  
    "pdf": {  
        "pdfData": <base64 encoded pdf>  
    }  
}
```

Credit invoice

Use this to credit an invoice.

Operation

```
POST /invoice/credit
```

Request

Name	Type	Mandatory	Constraints	Description
external_reference	string	x	This can be the same as the external reference from the sell request, but it must be unique otherwise.	
sell_reference	string	x		External reference from the invoice sell request
invoice_lines	array of invoice_line	x	Must contain at least one element	
total_amount_without_vat	decimal	x	Total amount without VAT	
total_vat_amount	decimal	x	Total VAT amount	
total_amount_with_vat	decimal	x	Total amount with VAT	
invoice_number	long	x	Must be a positive number	
invoice_delivery_option	string	x	"POSTAL", "EMAIL", "NONE"	This determines how the invoice will be delivered to the customer. The "POSTAL"-option means that the invoice will be mailed by postal mail to the customer's address. The "EMAIL"-option means that the invoice will be delivered by e-mail to the supplied customer email in the sell request. The "NONE"-option will not do any of the above. This option does not affect the response, which always contains the invoice PDF if the call was successful.
invoice_extra_data	string			Free-text field on the invoice

Sample request

```
{
    "sell_reference" : "79faa780-7b8e-11e4-82f8-0800200c9a66",
    "external_reference" : "23aef167-4alc-70a1-82f8-6775126alc21",
    "invoice_lines" : [
        {
            "article_id" : "sk-1",
            "article_number" : "1",
            "description" : "Lång smal skruv",
            "quantity" : 4,
            "unit_measure" : "paket",
            "unit_amount_without_vat" : 250.00,
            "vat_percentage" : 25,
            "total_vat_amount" : 252,
            "total_amount_with_vat" : 1252
        },
        {
            "article_id" : "bit-2",
            "article_number" : "2",
            "description" : "Blandade bits för hemmasnickaren",
            "quantity" : 1,
            "unit_measure" : "st",
            "unit_amount_without_vat" : 103.92,
            "vat_percentage" : 25,
            "total_vat_amount" : 25.98,
            "total_amount_with_vat" : 129.9
        }
    ],
    "total_amount_without_vat" : 1103.92,
    "total_vat_amount" : 275.98,
    "total_amount_with_vat" : 1379.9,
    "invoice_number" : 104
        "invoice_delivery_option" : "EMAIL",
        "invoice_extra_data" : "Data från kunden"
}
```

Response

Name	Type	Mandatory	Constraints	Description
pdf	Pdf	x		Invoice PDF encoded in base 64

Sample response

```
{  
    "pdf": {  
        "pdfData": <base64 encoded pdf>  
    }  
}
```

Configure invoice data

Use this to configure the information that should be printed on the invoice. Each store must configure this information before making a sell request.

Operation

```
POST /configuration/store
```

Request

Name	Type	Mandatory	Constraints	Description
name	string	x		
street	string	x		
zipcode	string	x	Only numbers	
city	string	x		
country	string	x		
phone	string	x		
fax	string			
email	string	x		
homepage	string	x		
vatreg	string	x		
orgnr	string	x	Only numbers	
companystaxnote	boolean	x		
logotype	byte[]	x		base64 encoded image

Sample request

```
{  
    "name": "Resurs Bank",  
    "street": "Ekslingan 9",  
    "zipcode": "25467",  
    "city": "Helsingborg",  
    "country": "Sweden",  
    "phone": "0727556898",  
    "fax": "123456789",  
    "email": "mail@resurs.se",  
    "homepage": "www.resurs.se",  
    "vatreg": "24598789",  
    "orgnr": "25879945987",  
    "companystaxnote": true,  
    "logotype": "/9j/4AAQSkZJRgA.... base64 encoded image"  
}
```

Retrieve invoice data configuration

Use this to retrieve the stored invoice configuration of the store.

Operation

```
GET /configuration/store
```

Sample response

```
{  
    "name": "Resurs Bank",  
    "street": "Ekslingan 9",  
    "zipcode": "25467",  
    "city": "Helsingborg",  
    "country": "Sweden",  
    "phone": "0727556898",  
    "fax": "123456789",  
    "email": "mail@resurs.se",  
    "homepage": "www.resurs.se",  
    "vatreg": "24598789",  
    "orgnr": "25879945987",  
    "companystaxnote": true,  
    "logotype": <base64 encoded image>  
}
```

Common API-types

Payment alternative

Payment alternative

Property Name	Type	Mandatory	Constraints	Description
id	string	x		Payment alternative id. Use this id to specify payment alternative id in verify credibility and sell call.
description	string	x		Description of the payment alternative
customer_type	string	x	NATURAL, LEGAL	
min_amount	decimal	x		The minimum allowed amount for this payment alternative. Applying for an amount with a payment alternative less than min_amount will result in an error.
max_amount	decimal	x		The maximum allowed amount for this payment alternative. Applying for an amount with a payment alternative more than max_amount will result in an error.

Contact information

Holds information about the contact person.

If the liable payer is an organisation, the contact person may be different from the liable payer.

Property Name	Type	Mandatory	Constraints	Description
email	string	x	A valid email address.	The contact persons email address.
phone	string	x	A valid phone number.	The contact persons phone number.
national_identification_number	string			The contact persons national identification number.

Sample

```
{
    "email" : "customer@mail.com",
    "phone" : "0700-123456",
    "national_identification_number" : "85052312345"
}
```

Invoice line

Representation of an invoice line.

Property Name	Type	Mandatory	Constraints	Description
article_id	string	x	Min length: 1	The line identity
article_number	string	x	Min length: 1 Max length: 30	Article number of the item.
description	string	x	Min length: 1 Max length: 100	A description of the product.
quantity	double	x		The quantity of the product.
unit_measure	string	x	Min length: 1 Max length: 10	The unit the product quantity is measured in.
unit_amount_without_vat	double	x		The amount with VAT per unit.
vat_percentage	double	x	Min value: 0 Max value: 100	
total_vat_amount	double	x		The total item VAT amount.
total_amount_with_vat	double	x		The total item amount, including VAT.

Example

```
{
    "article_id" : "sc-001",
    "article_number" : "1",
    "description" : "Screws 40x8 200/package",
    "quantity" : 5,
    "unit_measure" : "package",
    "unit_amount_without_vat" : 200.0,
    "vat_percentage" : 25,
    "total_vat_amount" : 250
    "total_amount_with_vat" : 1250
}
```

Pdf

Property Name	Type	Mandatory	Constraints	Description
pdfData	string	x		base 64 encoded string

Example

```
{
    "pdfData" : <base64 encoded pdf>
}
```

Error message

Errors (HTTP-status 4xx or 5xx) are structured as a message with the following structure.

Property Name	Type	Mandatory	Constraints	Description
error_type	string	x	"SELL_INVOICE_ERROR", "PAYMENT_ALTERNATIVE_ERROR", "CREDIT_INVOICE_ERROR", "VERIFY_CREDIBILITY_ERROR", "ARGUMENT_ERROR", "UNKNOWN_ERROR"	
error_message	string	x		Message describing the error

Sample calls with CURL

Initial store configuration

Before an invoice can be generated the contact information and logotype must be configured for the store. The logotype must be encoded in base64.

POST configuration/store

```
$ curl -v <endpoint>/configuration/store -H 'Content-Type:application/json' -H 'Authorization:<credentials>' -d
'{  
    "name": "Resurs Bank",  
    "street": "Ekslingan 9",  
    "zipcode": "25467",  
    "city": "Helsingborg",  
    "country": "Sweden",  
    "phone": "042-382000",  
    "fax": "042-202972",  
    "email": "test@resurs.se",  
    "homepage": "www.resurs.se",  
    "vatreg": "24598789",  
    "orgnr": "25879945987",  
    "companytaxnote": true,  
    "logotype":  
        "iVBORw0KGgoAAAANSUhEUgAAATYAAAbkCAYAAAAMjRzhAAAAAXNSR0IArs4c6QAAAASi0deAAAAAAA+U07fwAAAAlwSF1zAAALewAACxMBAJq  
cGAAAAAd0SUFB90DGwsH6dt90UAAAAdaVRYdEnvbW1lnQAAAAAENyZWF0ZWQgd2l0aCBHSU1QZC51BwAAIABJREFUeNrtnXuQHMV9x7  
/ds7f3vtNJd3qip4ETeiDE0xJCEGSMCSHYiSmCuyRO  
/CA4DqbKsQsX2IUroQK4klSIy47tuFDsoJiE4IpjAg5YIFsgITBIEZYQenJIOp3udLrH3mt3Zzp/7Mzsrt3t67mZP7tzd/2rurp9z07OzKd/v  
/71r3/9ayaEEDBix1lRKSTc3AIjRowYw2bEiBEjxrAZMWLEiDFsRowYMWIMmxEjRowYw2bEiBFj2IwYMWLEGDYjRowYKaMkJuNJD  
/YN0617EACw70Qv0lnHf29pcylm1iVhcYbLFjuZwoaNkWhIKPaGre3sALYd0IMd73ZhT1sPDp7uR2p4GHDSAArgD+X++z5oEmAJAAywatA6tx4Xz  
q7DFUuasOGizly/vAXJhGVat2FjGE1hRiyOS6q2v3MGT+1+H8  
/ubUdbVw9gpwC7H3CGAwCIEA7gnTbzuLjP6WMAYJWAQVQytUCiDsnKJmy4uBkfu3wB7rh6IVoaqkzrN2wMoynGKDaG7eS5QXzv5aPYsuM42jrPAH  
YPk01zelxb1ZgvygLE8jPUAepwPjAFWHVDRBKtqLm5dMw93rV+Mjl+10GiEYWMYTRFGZTdsxzpTeOTzd7B1x3Gkh9uBTDeADLmrLHi3fubv+Hec9  
LjCPKA12SwFFaiEUj0QuvCxjb/luW4a  
/liWNzNqRg2htFkZ1Q2w9bZN4yvPr0PW3Ych51uB7JngzdPu0KBSCfvf5177OB3ob+hCMfZ9UCFS1YOn8xHtt86bT2Egwbw2iyMyqLYfvWi4fw4  
DNvo7f/FJDthPbu0h6d9jRqe8hrrb  
/X+216GsePAd11AMVs3HTZcvx7bsux4Vz6qeVvhg2htFUYFRSw3a4ox93fcv17D58BLDPAWJEAO1Lfjfj4b0KUw8PAUwBUEDCUY5x  
/xwAydlIi7Cnz62EvffesmUVxbDxjCaSoXkZtiefv19/MkPXkdq8BRgd2t6EhbyEr3zLKy7CXgjq3ofIWsxWoXnAWQVQMc3HrlpfjhZ6  
/GzLrKKakwh01hNNUYfd2w2Y6DL/14L/7h528CTi8gBo0BtCtYAwYlqGHKve8eEX4bG1RYYEP+pEWQonHwsQeqVXGjJuVg0dxn+48  
/X4epls6aMshg2htFUZVRUwzaUzit39mFn765H3C6ZKvOWO6Gqvfbc529aWjGgj2S6n+r7jRTZ4LU3gTyjacxBefkeyHqaidmoK5hGZ75wnrcG  
rupFcYw8YwmsqMimbYulMjuP3xV7Dj3cMuFLWhogn8JoORANHPHcmJ0ovhFFmdoTc+3i9jCM0EAkkqxHJmqV44tNX4ZPrFk9ahTfsDKOpzqgohi  
01nMGmx7Zj99EjgNNNegCwd4/pDwdKV+L1Rv7Lam/DtB60nEwt9DoBS14MQNDgp+Ji016I1QCJmXjy8zdPSgUybAyj6cBowteKprM2bn  
/8Few+dgwQ5wDGgzee0xtPGXG5F5I60YrvQz38nTc1znkXkTnTnsHS1CQ72H8Y113ngty  
/BBgd+NPFvA6ZtY18ZH8yaNwhg2htF0YTTtHtud392FrbeVsTZYO8BtafRPPduG1OhKe609zg8wzAYDJWCm89FmpMwMn3Sp57LehjabBaVNCu  
xcv33zBpgtaGjWE0XRhNqGF75NKd+OrTr+agMLVHUV1k7z+xj2W026Dwd040k5Ec+MtEQoKF11Wg5v+bI8LwHvue060EwTEGzb35jL83199OPYL  
tg2b+ItbFEPdtuPdtTzwyMuwcRJAlozzWRCS5f4z4ja7N1p9n0INxbU3odp3GahLoHVwIeLKEEPUUemvB6Jnqca  
/BZ+MiaVxJuSxtjqzCGzcbYGzXDaGIZWQ899NBDEzGds+mx7egd0Q0gtW40B7j7B5a74Zz8MQZwy33MyfufwKfzleMs9xiwf871dzDnb3m/750bz  
/Kx9Pe12hjYcofAGCIhzuHUJGoxsbW1ljorhk28WRjgBWP0YR4Bhd+dxe2vvY2wPrk3kB1naWehcsXrPQ8Fue4+9pW3LF2KzbPacSs2sqiVA3I2  
g46+ofw0run8PVn38Cxrr5gr+L30g5xpdi1jHAcQFpKVy  
/Cbh2+03frFyRCRh5GeJweWzDlJCnbuE1jdVAN8QRCo+pwWaiGSUsjs9euxybly7Dqn1NaKpJnpf0jGrtdGdsDGvsn04fQtu5ARzq7MX2w6ex  
/VA7+kYysWV03oTzx7udu05vngesLjkWwDjxcgkEx0mIu8xY3tILgHOon//ZTfhQ6/ySNrDO  
/iFc9shPckpvwL3hZPpa2Ao0As+xATTgxpxUr8Iuv3BCr4c1Es5GNHFKGDQg+B5Q8KxFcj6irFtGEHM8JpA9MFjYTzSiRsPD8PTdj08Wl0Zl01sa  
/vH4Ej774No6c7Y0do/Maitq0g489/ipoOpzoB11FcUORDzcby7i63XdfWfc13nb3nCx2  
/cX44sbSL5ytrazAQMbGy4c7iLvtKjGh4o4rAVqM4FhnGhfPmYnVc2eUXWGKxUYannBlCEOGOxc11+N7H78a3/+Dq  
/Hgh1Zi47LZOniZwqn+tHssk4cu9N5ydxgmDt4Ps1hkrMpBqM/u3Y5vrBxRcn03e1cVyychXs2tKI  
/bWNX291YMTqvsd2WHcex5+PqgA+TRq7GB0gcwIPELcBKAjAv+  
/Pe4xZgcfzBmiVla2xLztW755VwzynhxjI+VqWrMjeubMuVvр0PqSzdtmVplhscgpm5T9nWeQzCYBxrJgzA7v/4ib8  
/uoLOFBVgdpkAje3zswoz2/ChmWz3funfIf/W+5v+M8T7p/3050fTTEYbV67rCzXUGFx/P1Hr8I/bv4fK0b8fHqbh  
//7AMAG5ZgLDVxKAU0XkkUatK8kifyNYByr5jWWRbEd0zd1lI78+R6L1vc8v2F5gDj06z6dH736Xtk9gWxkRtnghihvNI99tuXYkz1MnBeyQTHw  
zevUgLgxNvTGTz6HK9MejbFYrRqXnk90bvXXYSHPrImNozGbd127mrDse4ugGfJFDXSXH3s3XnpsKZadzNa4Fn5WTx1k0HQnjOB7rx0hNzuR71EKA  
Co4TuAN4pvPHYTt0GvRzMVk41nC3  
//hmXNoed25YIzwd+hjd8i3hkjhqF33p0ctbEYzaiuKPs1Pfihfjg4pZYMBq3YXvk2XcAa1CJA7B8DMcfjibxGNWdli4y36ArLF7Cn1PgVN8Qnn  
yrDeu+vQ2nUh1FWa3gcEjqiVSX28bBjjP4ya9Plq2BFZONPwySPmtJzxncf1cLSXE5iR05aQn+ZAUnv6uc5yRmUyxGcdhw0Icf3vb2lgwGtda0d  
1Hz2J/+1mgwlZmz9SAL82VYUrjdPzFzvqaIPfScMuumjhUqqj9UKbf77Xla0R8cCHAZYpCeP13t1r2CkztPx4afhe04bpDUBjCEH  
/zyWnqvhedDU05AJRqErmKEa+d7MFvLdUvk3ntRE/u08PYcA0bP9gsJjWbojKKqjNCqepB9CdpMVRZHVVJC/Prq7CooQpXzG  
/Ejctm4YMLo22evH5JM65e2I1zd7rLymhchulHr76XC3hKa90U2SvftaYzJGQ624sR+DNgQ14eMiYlnX+KLhkJMXLqezS1wVMSx8k1ICA3Re2QUi  
/+zXc/z91rcdzv8df1zfzFCbzpwuncIcxurS6o0BbOhyZ5QYZseGyjJmN6NVpb7uArywLZDeOmPmlCZkBVuJ0vggW2HgoYtChsv  
/4my8s51krA5b0bqgSgJGJlui6gxTlocS/Uk7QN0R6MtckGoggzfa/+DMwTPAS4fQOrMG/3TbKtywd0xbetWIDdp/rKqj8F+6  
/pri2tu9qAxIgM0spEtptMY6aprimUJSNcWfAbARIpyXRWl5qoQx+QbGsQA+vHcMh1SC4/12cC/UBvIjAks/kAtu5sK6nCFMaG5Xt6QT0Geg
```

/IUMIL7EtZ6UyZ1cqc9t/NkPzZseR3PH+7CQDqLwYyN/z1yFhue21ldJ
 /vde677C2HjpT6oMT2vc60TEFTf1M2798a6ZFs11I0nTY2q7JEtDqP7wAKOD3U04+v/fwK73e8b8meuWNpddfwo2bC/8pgPdQynkM8s9Q6IoC
 /WoAkMbJWFj/xwvzLD5v2spS2z8U8Gqiqx+sTr1YEAQjAPyZ0zqTR493IVkZ/OTN0sZyOrMh94VTD8EbFgrZ0+0a5TqScrD8ce4xb7SncMu
 /7UXdo9tR++jLuPnf9uCN06mQVEewoA3VFnDul3CCizqvFjE50Rk9/zhyg07kY30eYs+o8zyJ0ReFB
 /Asu28q+lHeBrLx0a82cWNFSXXX8KHoo+u7cd4GkSW0G1zWTFSLmPHSevSNI+E5pqBmNBkoakRkEsmC82SHjULntalcsh3+gd7Qx07X1tKu1lDuK6
 zA8By3WxvMTAXQMLBzsNn0Z0aKdkmFgWx0WEC6SzX0fbFSLBaqvNFhhqOMAbJqrhAH+Py7D1VmSYJBk14t17KxS88+RMidHFk00kR1d2LhACsJ1
 806XXMX8YXoh7opQxisxI+Oy3HT+Ht00gOcoE34yqCjIKK4
 /+FOyx7TjUBfCM29uz4JBTwleI4PBSSmtaGHKD2e5EoZ05niBTvj1FPRBPjo8Mvvycj5wF1Qa+acU8XG1vBmUH3Gm0+gm0HzpRMaaKxUYwaxX+BoT
 yX42CcxuRAYnKwsuCzaR5TpELBnhIor80C5w4SuvCYcmVoHVM2YzLsxYuhbIziT6KwYJ22yDE2Tzj0Rgpzx3GcDo1MurPpNjZhv
 /p9acgw9Y7mMb+k325mQuqADQwsYcHjC04iFedo1f+R031EKQRMF08jZGhprpwPmVB6ZBQ1coIKBFUw
 /YJp0HISdnUfOlkRh1rMBlwsUgturKR+v+Fbf8g5Vxxcca+fU4eeefWIOBr2xE31eh59uXo1LWuqUyQoWNQgBEj3QD1kFDxq0mLIZm5Fm
 /wgmqZDr8e06t7FsCl6QvVHikpqwlBpGRRDY+XoOXNtvCQasExKLG94jmqaCi670QvbJaGVH1AqvDag6Wk1ellftJjB3qhsUyy7lim
 /7g0Axeyes+4+k3yJ3g71ZHCbkjQ+InGJx5Esqe9/teK62AyeQfn38XEmUJjobMtjwvE
 /ilp5HADUTJ41ckPQM50M+K5pr8Ku71q12mZ+tu+2iZmxcoANXbxKTh7oHPFEmlCwcWdlpaID8cSdc8WPZkxG0HxsNMQ6md44cdVIRY1Li2Dbp
 /scMCKPJFBkd9mMaJRWjW42dp
 /scXk4QR31zqHIjAry2Paf6g0YziPQeEyc6aUw2I+XB4SFRQvoj4WC3qGk1JCvtbQhuUFpGnPQRdOU1nocuAaEJlrjGXQqKzUYwa9WbVa4U6c0
 ZmsL3ORVNas58ENiyWj5k1jvQLfuG5xyPlxpY1oPgjFTXJEJLk1hmwiMZI6WSZ7ZqqB4UyuqxY1OZfpnAimCUooIYP80PG+axaN+TP
 /eeCmpMzmSg23EjAcqyGM73JECmBPBWxPybI4HhltAoiI
 /K8KUxhnmcUUVOjkgkhDx7JNV7V4ZgTBOr4OpnyYe8PCxBehdvHwsvyVQ4gMXQnRopSZA6MhvGZYPc80aKMQYBTzjam0BAyKy3Mpy9YVH4Wt8blzS
 RTUeoEru9uB9TsvLf77jv0477uw4JnJPPCRFLNmZEnK7pUpPh+o+L6KJJ0b22GiYgfo26rZ5Xudi+R7dVXPrcM8Vc0b9iV0nerHrlfjS06Z20W6
 +qjd5UDxGBRm2g6f73cxgj04R6Hofbw0gNBu3aaCanY91Y5phDzksFyEOh01vkI1LPf8cj044010yhjtFTWvGc8+lu3Qc6xro0jKE40NzXKns12
 5Bi44BwQDA40QAvRK7w426rZus2vCYzAtNRVBBfM6EEbKSXsFKRnPDTu9FQhQhy+QZwkRPZjMplCKDwYjvB1BLIH5a3fGzY1Jnm0Bie6kUCmxY
 14Me3XYzEKO1GdvBF18gCdhS+MYzo14SPC8qo4IMW/dAmuxwa
 /2GEr41lZQZzeRaXFACmec1ykyUqyhVjC1Qh1IC1MnwWtuJMpodzfKJk35hRpE5bbteac6odqfSRVeaSGx0//1hpncznxiENFvHAVhKvG10sUzp
 /DyQGAzzyHoxNOptev7oW74q6Qw0C+MEZuxGVmyoWE80DOPpof4XluigKEoDw
 /leAaBM1Q1OjqrK7C4oRj59Ti1gubRvXCPbnvxaPyfXowZ3QdJ++xeUsWvVg1Lz6jggxb71Am70pqS0zz
 /d6FTFWKR+HKV5ENPMNvnrr9uBtY1nHqnkrjucNn8bXtr3FmIENChqS3ZxwMvwvkrn50kkR4Q6NPJhCuDlfNmDnrql3KFP8Gbd1bBgZrjM5uZ05
 HhtTkpM5xzut1VpnHeaxeT06o0Mhn19KxWei1UhgnmedjzYjm2IDAk1b01a4u4YUCosOIhpv48rqixFwfEbj3xaP4zt404mWyyM54w00
 /XMSJk1AcRoUZtsFMMND1hfaArOcL1MGnQUsn5wloA2XF1lQtNwNHqhC9dvgDXL27Czd9
 /HcO0MoPjXoNQJupYn93Jn6vX63jxH+56eo77mAuKrrLFv5pIbM1wXm1Xowm0iYfVjDhbKJF2t1c45kwEZwtVDfl9b0Bj08MySbmbeB14Uawcf
 xaSsRDRDEusqSgQCHlg7CD+9Lkj2NWeckAcNeHktENY4DtGgR09rYo1qOC+uDUSDY4ta32JH7SLAvOVkqTA7SBFphrOFHSOqsGv3dJi6Zx5IlptThs
 NPJgoqSau+utPgaF08au2RmITUbh1tyN1AtbNQyoqGKZZN6kmdodMODf1RMDozGg82ERmxNWinUq2AG05ybeKZ04
 /twplkZpgeatjAJue2p8zaoFQE9Pkq1EbYRWVET
 //xsjkKgTU42FqdQhlyEmHiNsdhxLLksZqbeP1m12mHxL4123JpWbo+suyNLoQnkwmKhMh55qIyvHNagZwlBWGSgJpd56R
 /XjiCsbDSNOzzoteQ01DY0Ix40
 /cVLHzCqbx7Z2Ti203305ttzyAVxQ10QgF1xtELWL7YvDqCDDV1eZILEATEPT5seo+w+yfCCRzsIIItamhEs17fcOhKha65RxX01WgBnzzjyle
 /IYX1y2geCwsvkG8YaEZk1VOYXmzqnV2aWlc1zxGOLBzmxGyoQ81Aoqaq1tv2wtk7NxBpIWxx+vmo13P3Mz7r18rmy01LTCuuWPRWJUmGGrSsi5
 YmyDaHUkwtU2yBDCe0Weelk0Pdg3jmYjc+2AqgnimkSzt1XWem8XK8XkgAM2uTxVeaKgxGvQ5a5cFrGQIxswxkJ12zDhg0k7EpmlAjhiRUN
 N3PNhsQ8ubSwzFJdYeEfNi3Fv9zyATfMqVw5
 /oOt0IMCpo8aKmvmB6RhmKwSqqBZVvMdl0hPqczNhzbRbEegPZXG80e78bVfHsNQNry+er8gs4KmpBnQAno0IOp7F24w1CmN8kRjo2x9xhXDZ9H1s
 VbzvQht0A0W2XTZzeqUz1FJ7qTgsWATmRFT1rJrdmqXS
 /O+InJi39wZzQvjdHVJC01VF15uqsa1LTX47WVN2HBBfS5dZxT5o1wZMBx8pkxjshZEGBkpWvm5i0Wh1FBhm3RzBrgPQ7ADhmCImjk1DWHWq9B5
 HPdog7nfeRxtfsryhC2jj0k6EJ1ComTNy+x+pRCeg+WfJAdsgaUttUVxmkhsmBLbUUMGdAlVr1ya0q4EWUH1d1mEygBZbNiMixEg1xfksJpkpMTWC
 Za019A9nEX3cBzHekbw3LEePLr7FJbPrMkj1y/G7144c9TP37N2lV7n6Dn87Ei3UiomsjkkXJRBQ1FF82qyU3DjhZfk6pqKK6zT110NdjwNnkm8
 /wmUFK6ah00T14UBFYLNubOp5rx3H0rttJeysoDeyGBwbflTwje0LlhsCgfSpuTMvhjwaYgRhhlBp5i8sqll3A8+k73MD76k4P4u9dPjXns39+4h
 NxqulSR60d1E8yoIMO2Yn5DDozagwYahvQL3blmswqu2UsxomET6t2bOE001Xhev7eXjdMnoM1+UduedddA6t74kOwhFYqPuzk2v0csA92bfP
 Fnqubg5QzbqKEphKw0nRmwK0x
 /FgKslhNthaiK7IAHgK9vfw5sdqVGpu7CpGh9e0gTt8r4SMCrElcsaQicSzcPcVOM4ul6Thawr5eNLntQ1KbkwukZqBq9oqkWQ7
 /Wn29Xs1Ezf6wSmnnv9WYHwfUlaXCR2CCs6oniRajGLDb0mlq
 /DeHEiM2BTHiy1Iyeg20IO0ErK0eZ4xaAN95q2PM4267sk1s+10Qywud14CZ1VxyImTdxIFFndIRxqK1Mc3uR4XEdk18sYAB1RhC1aCqf95ifbv
 Yng6SrhyXhu2cX1s0kaW2Q2gcR3uxjSGwrbsNsUpqC8WAbigmcekPV0quB8q031CSi8vvT920aFr5teXTX8Kvis3rzwdZc2MGen3iS11gZm
 lN3JAIC+z8Jgl1+fnqybOW7orRmnQ7yu8wzetrNCTERSblMouwwLzEYNsquzpSwanZ4Y+6apChxDntez0dUsYcvZyr+Uqj019vrNrfWVzdfgg3
 b9a0tgB0125kurSDBuFkv+d1oAjxcwNr1KguryyWjY8tJhhxOGqrhdByfqE1i9cEbJGlthbEIUS7c1Ksr+LIpmJE6+oFaobe8ydoJCEmzTVV
 W2RgVfHduXTOp9Di1eAhagdESNg7pT5PR7Dcc1JB1X9B5jjJUHcrmeSyryQbDyWjQ4s9mfIzobk81zjZahLmgfuzcsrQtysaoYMO2aFyt11l
 tBr185jGYMO9EMwtz3ivwNa52/wjuWr+4pFc4edmuoKMrM5upxuiWpwn7ut3D2bIxGpc
 /+6kNS4A0n6SnfdzvffgbjsDcBC+LVzB52ZsIfxnZTBVGNQmOl/jrQsN1/9nhsjea1939+JUXIjk15z8j5dKI6aM
 /fcP45LpFJc2Rmmz12YqMew4J8
 /8gEsaawa89jX2vvLxmhc5xZV41PfnARMMzDb7iYRsqqiHv2Apf81gfK8tuGtxZTHZGq5qr8cLmfFjds6LnvD59sLtsjMztEu+
 /dTmsocT4Pjx1FUsAPY0445qFuHbOfdnOwrCJL5vJwIgDaEhaWfifxKZfdfjs1fpwy9cib2fWhNp7wMA2HmyH/u6BsvGKDHd7bOa8Dhr7oAT+1
 /D6gW+ptOSzbr1iX6i51ZyDGT0Cpo7McD964r6zkyNvFlUzRGUczGlo1t37V9
 /5f2yMjqvQMDv3MjR4AGLrDMcpdKGxyV3Co9zzqZwxzUlSwJBY91PpZxssk74F2VsZ9qzKQyj24mPym3ZdwYvvtdbVkbzndhWL5yBL374ImCQKT2
 J7rHs24wiY0HklhuiEmr1CSCdRv3
 /MB7bfGksGlex2ESR91G2S2sfyEx7NsVg1FPCTw1Gkzc7Urj3F8fkzui8p4a+8dGVWJCsBtJebTKhd6mjKI573Lkx81
 /ODWcROzk3gAdiuuWQlmmPci0rFyBNFRuutx9WTT0E2E3o7cBsl/LG6dTuPk/DqA
 /455d0XkbtrqqCvztJ9YAQ0K2xNdtBk3CopVQ8zJw0HfFOSHqFPzCac6rq8Cxb2mNldIug00UeXjXcfRpPiie4Sz+eucJw6YIjP794NmyXUPWEXj
 81+3YsPvtdA11Y8FoQpJ57rhmET591Vkg35HjARIAES2WIwR+fKBr1N976p2uGDRJyKG0qjrgCctd19T1tyUrGJ1kd6Rrbh62/w/NFzGMzY6E
 /b+Onhbqz7ch+09Y4YNkVg9P29HdjeVlpvud2VxrffOo1Ln9iLL247jhFbxIYRE2Ji5o6H011c+Y0Xst81ANQnyMygPL
 /pLueaxenqYthcEuAvPrEKGxcB2H5n+
 /u92Ptuftj1ClOpQwIBoL0bP9y8GnetXxJbxZ1INrmHMZhnaRsJpJrkN8bs0cbf7eJxJXN1WisTIX7F66s15C2HYzYAgm0jY7BDDoGMjjje04I9Z
 wbwZscA9pwZgBNTRhNm2ABg/81erH94G3qrBFbbQchVchVt9hSNrnXpabCwgPrFuDOS1owv64Cp1IZbD3Qib
 /aeXLuzVdKpjgAcLoXn1rRjCc+c3XsFwci2cToSElyNobRxDoaUMMGANv2d+Cwv

```

/sV0jMTQFVCrkrAlaKSXm20uHsGgVsKgI5e3DSvFj+7bwOSCWTsKI5hYxhNF0YTHni4ccUcPPHq2B1ZYARWjNfExwNBEHHF9MpeWytsw9XNybxX
/de06kUx7AxjKYLo0QxvvST6xZjKGpjm0+9letAqipyF8RUOErmrEAwk7rcWe+qC32mD6urOX523wZUJxOTTnEMG8NoOjCa8KEoladea8MffX830
k1JoC6pd6kD241pX0pyudXqrenowfUt1five69FY00Sk1kMG8NoKjMqqmHzYga3P
/4KUrUW0FC1D4JqA6FlhkNvS9YGuvrwsaUz80Td10xqb8CwMYymA6QiGzYAONjeh9/71qvYnxoGZtVoNlyhgVCUF4560
/qGYJ3qxtdvX4Gv374SU00MG8NoKjIqiWEDcnk6n
//hm9iy52Su56mu0LjUEeEUC5B6Kzr7MDedxpN3X4MbV8yZsopj2BhGU41RyQwbjRvct3UPTjMAS2rJXpZlhKPPzDCTwh0rZ+Mf71yLloYqTAcb
AyjqcKo5IYNAFLDGtz4zNv41ouHYc+qAWZUK0mHChyJywQCUi99JAocG0BrJc0377p8WngCho1hNBuZlcWwebL/ZC8e
/tkBbP31SaCxCmiqDWZSS1vmSaceckWscCDaedeAbbf23Lsfnb1g2KXOgDBvDyDAsDLWYAAABXk1EQVSKgWHz5GB7Hx559h386NX3YM+oBhqq
gcoKBQ4KAzSWOA6QGgZ6BtBan8SXb2nFXesXG6UxbAyjkCaofobNk86+YWx55Ti27mzDntP9QG01UJ0Eap05HeSZZv
/PAJdRQA2ngYERYHAEjbaNW9fMw6evWzqthzWGjWE0FRnFyrBR0daZwrN727H9YCe2H+xE54idWztXWQEkrNx
/D0SFBFVG8N8b3JGsD6Swks29PpLFFQsbcDPKObh+eQtuvGS28QAMG8NoijKKrWHTxRP2tPVg34len04dxr4T+dpTx7oG0D2QxuoLGpG0ckHTuY1
VuHBOhvBmb8DqCxqxYn7DlM1IN2yMGEZTxLAZMWLESFTh5hYYMWLEGDYjRowYMybNiBEjRoхM2LEiBFj2IwYMWLEGDYjRowYw2bEiBEjxrAZMWL
EiDFsRowYMWIMmxEjRowYw2bEiJHpi/8PQHO/M6LtlZ4AAAASUVORK5CYII="
}'

```

You can also verify the information with a GET request to the same resource:

GET configuration/store

```
$ curl -v -X GET <endpoint>/configuration/store -H "Content-Type:application/json" -H"Authorization:<credentials>"
```

Continue to the next step if everything went ok.

Sell invoice

Start by listing the payment alternatives:

GET payment_alternatives

```
$ curl -v -X GET <endpoint>/payment_alternatives -H "Content-Type:application/json" -H"Authorization:<credentials>"
```

The response contains all available payment methods:

GET payment_alternatives Response

```
{"payment_alternatives": [{"id": "M3310115", "description": "TEST FTG BUTIKSSERVICE", "customer_type": "LEGAL", "min_amount": 1000.00, "max_amount": 100000.00}, {"id": "AF682069", "description": "FAKTURA TEST", "customer_type": "NATURAL", "min_amount": 1000.00, "max_amount": 50000.00}, {"id": "LG686069", "description": "TEST FAKTURA MED DELBET", "customer_type": "NATURAL", "min_amount": 10.00, "max_amount": 50000.00}, {"id": "NZ690101", "description": "TEST FÖRETAGSFAKTURA MT", "customer_type": "LEGAL", "min_amount": 1.00, "max_amount": 50000.00}]}]
```

We choose payment alternative "AF682069" and next we call verify_credibility:

POST verify_credibility

```
$ curl -v <endpoint>/customer/verify_credibility -H 'Content-Type:application/json' -H 'Authorization:<credentials>' -d
'{
    "payment_alternative_id" : "AF682069",
    "requested_amount" : 1488,
    "government_id" : "<government-id>",
    "contact_information" : {
        "email" : "test@resurs.se",
        "phone" : "0708-123456"
    }
}'
```

```
}
```

We get the following response:

POST verify_credibility Response

```
{"decision": "APPROVED", "approved_amount": 1488}
```

Because we got decision "APPROVED" we continue with sell:

POST invoice/sell

```
$ curl -v <endpoint>/invoice/sell -H 'Content-Type:application/json' -H 'Authorization:<credentials>' -d
'{
    "payment_alternative_id" : "AF682069",
    "government_id" : "<government-id>",
    "contact_information" : {
        "email" : "test@resurs.se",
        "phone" : "0708-123456"
    },
    "external_reference" : "TEST0000001",
    "invoice_number" : 100000001,
    "invoice_lines" : [
        {
            "article_id" : "BOLT-012",
            "description" : "Bolt (M8x125mm)",
            "unit_measure" : "st",
            "quantity" : 300,
            "unit_amount_without_vat" : 3.98,
            "unit_vat_amount" : 1.00
        },
        {
            "article_id" : "NUT-001",
            "description" : "Nut (M8)",
            "unit_measure" : "st",
            "quantity" : 300,
            "unit_amount_without_vat" : 0.98,
            "unit_vat_amount" : 0.25
        }
    ],
    "total_amount_without_vat" : 1488,
    "total_vat_amount" : 372,
    "total_amount_with_vat" : 1860,
    "invoice_delivery_option" : "NONE",
    "invoice_extra_data" : "Data från kunden"
}'
```

The response contains the [invoice PDF](#) encoded in base 64.

POST invoice/sell Response

```
{"pdf": {"pdfData": <base64 encoded PDF>}}
```

Credit invoice

To credit the invoice a sell reference must be supplied. This is the external_reference from the sell payload. To credit an invoice make the following request:

POST invoice/credit

```
$ curl -v <endpoint>/invoice/credit -H 'Content-Type:application/json' -H 'Authorization:<credentials>' -d
'{
    "sell_reference" : "TEST0000001",
    "external_reference" : "TEST0000002",
    "invoice_number" : 100000002,
    "invoice_lines" : [
        {
            "article_id" : "BOLT-012",
            "description" : "Bolt (M8x125mm)",
            "unit_measure" : "st",
            "quantity" : 300,
            "unit_amount_without_vat" : 3.98,
            "unit_vat_amount" : 1.00
        },
    ],
    "total_amount_without_vat" : 1194,
    "total_vat_amount" : 298.50,
    "total_amount_with_vat" : 1492.50,
    "invoice_delivery_option" : "NONE",
    "invoice_extra_data" : "Data från kunden"
}'
```

And the response contains the [invoice PDF](#) encoded in base 64.

POST invoice/credit Response

```
{"pdf":{ "pdfData":<base64 encoded PDF>}}
```

Invoice Service 1.0 (deprecated)

The Invoice Service provides services for invoice operations.

For detailed reference, please download the [Invoice Service manual](#)

Consumer Loan API

Authentication

Authentication is done by using **http basic authentication mechanism** (Authenticate pre-emptively).

Below test accounts are for generic testing only. For integrating anything specific for you, please use your designated test account.

User: testse
Pwd: Resurs12345

User: testno
Pwd: Resurs12345

User: testdk
Pwd: Resurs12345

User: testfi
Pwd: Resurs12345

WSDL library

Test: <https://test.resurs.com/cl-webservice/ApplicationService?wsdl>

Production: <https://consumerloan.resurs.com/cl-webservice/ApplicationService?wsdl>

Webservice URL

Test: <https://test.resurs.com/cl-webservice/ApplicationService>

Production: <https://consumerloan.resurs.com/cl-webservice/ApplicationService>

Rest base URL

Test: <https://test.resurs.com/cl-webservice/>

Production: <https://consumerloan.resurs.com/cl-webservice/>

IP-addresses to whitelist

Production:

91.198.202.100
192.121.110.100

CTE Integration:

194.68.237.250
192.121.110.100

Decisions - social security numbers

Sweden

MANUAL INSPECTION: 970415-2371
APPROVED: 850614-8264
REJECTED: 650329-5799
TECHNICAL ERROR: 560625-0396

Finland

MANUAL INSPECTION: 130100A-517V
APPROVED: 040190-811S
REJECTED: 260274-290U
TECHNICAL ERROR: 030654-428H

DENMARK

MANUAL INSPECTION: 2111969591
APPROVED: 1506899591
REJECTED: 1708749591
TECHNICAL ERROR: 0112589591

Norway

MANUAL INSPECTION: 05129911125

APPROVED: 10108819559
REJECTED: 01097414277
TECHNICAL ERROR: 09085905462

submitApplicationExt DK

DK

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:app="http://consumerloan.resurs.com/v1/msg/application">
  <soapenv:Header/>
  <soapenv:Body>
    <app:submitApplicationExt>
      <xml><![CDATA[<resurs-response>
        <applicant-credit-status-consent>true</applicant-credit-status-consent>
          <amount>10000</amount>
        <time>60</time>
        <purpose-of-loan>OTHER</purpose-of-loan>
        <purpose-of-loan-other>HOUSING_MORTGAGE_LOAN</purpose-of-loan-other>

        <applicant-government-id>2111929591</applicant-government-id>
        <applicant-mobile-number>666 77 888</applicant-mobile-number>
        <applicant-email-address>test@resurs.se</applicant-email-address>

        <applicant-nationality>OTHER</applicant-nationality>
        <applicant-in-denmark-since-year>0101</applicant-in-denmark-since-year>
        <applicant-resident-permit>F</applicant-resident-permit>
        <applicant-resident-permit-valid-date>0123</applicant-resident-permit-valid-date>
          <applicant-ancestral-homeland>DK</applicant-ancestral-homeland>
        <applicant-marital-status>SEPERATED</applicant-marital-status>
        <applicant-number-of-children-in-age-group1>2</applicant-number-of-children-in-age-group1>
        <applicant-number-of-children-in-age-group3>1</applicant-number-of-children-in-age-group3>
        <applicant-habitation-form>WITH_PARENTS</applicant-habitation-form>
          <applicant-share-of-expenses>50</applicant-share-of-expenses>

        <applicant-employment-type>TRAINEE</applicant-employment-type>
        <applicant-employment-year-from>0101</applicant-employment-year-from>

        <applicant-monthly-income-gross>10000</applicant-monthly-income-gross>
        <applicant-monthly-income-net>8000</applicant-monthly-income-net>
        <applicant-monthly-rental-cost>5000</applicant-monthly-rental-cost>
        <household-living-expenses>1000</household-living-expenses>
        <household-other-loan-expenses>1000</household-other-loan-expenses>

        <applicant-dankort></applicant-dankort>
        <debit-cards-master-card>true</debit-cards-master-card>
        <credit-cards-master-card>true</credit-cards-master-card>
        <credit-cards-visa>true</credit-cards-visa>
        <credit-cards-diners-club></credit-cards-diners-club>
        <credit-cards-petrol>true</credit-cards-petrol>
        <finance-other-loans-balance-of-loans>100000</finance-other-loans-balance-of-loans>
        <finance-other-loans-monthly-cost-of-loans>3000</finance-other-loans-monthly-cost-of-loans>
        <payout-clearing-number>1234</payout-clearing-number>
          <payout-account-number>1234567891</payout-account-number>
            <approve-conditions>true</approve-conditions>
            <agreement-sign-type>E_SIGN</agreement-sign-type>
        </resurs-response>]]></xml>
        <consumerIp>1.2.3.4</consumerIp>
        <!--Optional:<br/>
        <externalReference>1234</externalReference>
      </app:submitApplicationExt>
    </soapenv:Body>
  </soapenv:Envelope>
```

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns2:submitApplicationExtResponse xmlns:ns2="http://consumerloan.resurs.com/v1/msg/application" xmlns:ns3="http://consumerloan.resurs.com/v1/msg/exception">
      <submitApplicationExtResult>
        <applicationReference>lacca31b-c4c8-466a-9f7e-51c8c06591dd</applicationReference>
        <decision>APPROVED</decision>
        <approvedAmount>100000</approvedAmount>
```

```
<interest>22.30</interest>
<tenor>72</tenor>
<effectiveInterest>25.31</effectiveInterest>
<monthlyCost>2598.00</monthlyCost>
<consolidationDemand>10000.0</consolidationDemand>
<adminFee>19.00</adminFee>
<arrangementFee>399.00</arrangementFee>
<documentTypes>ID</documentTypes>
<documentTypes>PAYMENT_SLIP</documentTypes>
<totalRepaymentAmount>187007.00</totalRepaymentAmount>
<monthlyAccountFee>0</monthlyAccountFee>
<totalFeesAndInterest>87007.00</totalFeesAndInterest>
</submitApplicationExtResult>
</ns2:submitApplicationExtResponse>
</soap:Body>
</soap:Envelope>
```

submitApplicationExt FI

FI

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:app="http://consumerloan.resurs.com/v1/msg/application">
  <soapenv:Header/>
  <soapenv:Body>
    <app:submitApplicationExt>
      <xml><![CDATA[<resurs-response>
<amount>10000</amount>
<time>120</time>
<applicant-government-id>020189-074E</applicant-government-id>
<applicant-mobile-number>+3585005555127</applicant-mobile-number>
<applicant-email-address>test@resurs.se</applicant-email-address>
<purpose-of-loan>HOUSING_MORTGAGE_LOAN</purpose-of-loan>
<applicant-marital-status>MARRIED</applicant-marital-status>
<applicant-habitation-form>DETACHED_HOUSE</applicant-habitation-form>
<applicant-habitation-year>2010</applicant-habitation-year>
<applicant-employment-type>FAST_ANSTALLD</applicant-employment-type>
<applicant-monthly-income-gross>3200</applicant-monthly-income-gross>
<household-monthly-income>6400</household-monthly-income>
<household-living-expenses>300</household-living-expenses>
<household-mortgage-debt>10000</household-mortgage-debt>
<household-other-loan-expenses>200</household-other-loan-expenses>
<other-expenses>400</other-expenses>
<household-mortgage-total-amount>10000</household-mortgage-total-amount>
<household-mortgage-monthly-cost>200</household-mortgage-monthly-cost>
<insurance>false</insurance>
<confirm-legal-documents>true</confirm-legal-documents>
<approve-conditions>true</approve-conditions>
<agreement-sign-type>MAIL</agreement-sign-type>
</resurs-response>]]></xml>
      <consumerIp>1.2.3.4</consumerIp>
      <!--Optional:-->
      <externalReference>onboarding1</externalReference>
    </app:submitApplicationExt>
  </soapenv:Body>
</soapenv:Envelope>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<resurs-form>

  <!-- ***** Lånebelopp -->
  <element java-property="amount">
    <label>Lainasumma</label>
    <description>Täytä toivomasi luottoraja (€)</description>
    <name>amount</name>
    <is-mandatory>true</is-mandatory>
    <type>integer</type>
    <min-value>100</min-value>
    <max-value>40000</max-value>
    <format-message>Kenttään on täytettävä haettava summa!</format-message>
    <unit>money</unit>
  </element>

  <element java-property="paymentTerms">
    <label>Takaisinmaksuaika</label>
    <description>Toivottu takaisinmaksuaika.</description>
    <name>time</name>
    <type>radio</type>
    <is-mandatory>true</is-mandatory>
    <unit>time</unit>
    <items>
      <item>
        <label>1</label>
        <value>12</value>
      </item>
    </items>
  </element>
```

```

<item>
    <label>2</label>
    <value>24</value>
</item>
<item>
    <label>3</label>
    <value>36</value>
</item>
<item>
    <label>4</label>
    <value>48</value>
</item>
<item>
    <label>5</label>
    <value>60</value>
</item>
<item>
    <label>6</label>
    <value>72</value>
</item>
<item>
    <label>7</label>
    <value>84</value>
</item>
<item>
    <label>8</label>
    <value>96</value>
</item>
<item>
    <label>9</label>
    <value>108</value>
</item>
<item>
    <label>10</label>
    <value>120</value>
</item>
<item>
    <label>11</label>
    <value>132</value>
</item>
<item>
    <label>12</label>
    <value>144</value>
</item>
</items>
<format>^(12|24|36|48|60|72|84|96|108|120|132|144)$</format>
</element>

<!-- ***** Huvudsökande - Personalia --&gt;
&lt;element java-property="applicant.govtId"&gt;
    &lt;label&gt;Henkilötunnus&lt;/label&gt;
    &lt;description&gt;Täytä henkilötunnus muodossa PPKKVVYXXXX&lt;/description&gt;
    &lt;name&gt;applicant-government-id&lt;/name&gt;
    &lt;is-mandatory&gt;true&lt;/is-mandatory&gt;
    &lt;type&gt;string&lt;/type&gt;
    &lt;length&gt;11&lt;/length&gt;
    &lt;format&gt;^((\\d){6})[\\ \\-A]((\\d){3})([0123456789ABCDEFHJKLMNPRSTUVWXY])$&lt;/format&gt;
    &lt;format-message&gt;Kenttäään on täytettävä henkilötunnus muodossa PPKKVV-XXXX tai PPKKVVXXXX!&lt;/format-
message&gt;
&lt;/element&gt;

&lt;element java-property="applicant.mobilePhoneNumber"&gt;
    &lt;label&gt;Matkapuhelin&lt;/label&gt;
    &lt;description&gt;Varmista, että matkapuhelinnumeroasi on oikein.&lt;/description&gt;
    &lt;name&gt;applicant-mobile-number&lt;/name&gt;
    &lt;is-mandatory&gt;true&lt;/is-mandatory&gt;
    &lt;type&gt;string&lt;/type&gt;
    &lt;length&gt;16&lt;/length&gt;
    &lt;format&gt;^((\\+358|00358|0)[- ]?(1[1-9]|2[0-9]|1[0-9]|201|2021|[2][0][2][4-9]|[2][0][3-8]|29|[3][0]
[1-9]|71|73|[7][5][0][0][3-9]|7[5][3][0][3-9]|7[5][3][2][3-9]|7[5][7][5][3-9]|7[5][9][8][3-9]|5[0][0-
9]{0,2}|4[0-9]{1,3})([- ]?[0-9])\{3,10}\)$&lt;/format&gt;
</pre>

```

```

<format-message>Tarkista puhelinnumerosi.</format-message>
</element>

<element java-property="applicant.emailAddress">
    <label>Sähköpostiosoite</label>
    <description>Varmista, että sähköpostiosoitteesi on oikein.</description>
    <name>applicant-email-address</name>
    <is-mandatory>true</is-mandatory>
    <type>string</type>
    <length>50</length>
    <format>^([A-Za-z0-9!#%&'*+/=?^`~-]+)(\\.([A-Za-z0-9!#%&'*+/=?^`~-]+)*)@([A-Za-z0-9]+)(([\\\\.\\/-]?[a-zA-Z0-9]+)*)([A-Za-z]{2,})$</format>
        <format-message>Tarkista sähköpostiosoitteesi.</format-message>
    </element>

<element java-property="loanPurpose">
    <label>Lainan käyttötarkoitus</label>
    <description>Lainan käyttötarkoitus</description>
    <name>purpose-of-loan</name>
    <type>radio</type>
    <is-mandatory>true</is-mandatory>
    <items>
        <item>
            <label>Auto / Moottoripyörä</label>
            <value>MOTOR</value>
        </item>
        <item>
            <label>Hääät</label>
            <value>WEDDING</value>
        </item>
        <item>
            <label>Vene</label>
            <value>BOAT</value>
        </item>
        <item>
            <label>Asunto-omaisuuden hankkiminen</label>
            <value>HOUSING_MORTGAGE_LOAN</value>
        </item>
        <item>
            <label>Sijoittaminen (esim. osakkeet)</label>
            <value>INVESTMENT</value>
        </item>
        <item>
            <label>Ajokortti</label>
            <value>DRIVERS_LICENSE</value>
        </item>
        <item>
            <label>Sisustus (esim. huonekalut, kodin elektroniikka)</label>
            <value>INTERIOR</value>
        </item>
        <item>
            <label>Remontointi, kunnostaminen</label>
            <value>RENOVATION</value>
        </item>
        <item>
            <label>Matkailu, vapaa-aika</label>
            <value>TRAVEL</value>
        </item>
        <item>
            <label>Lainojen ja/tai luottojen yhdistäminen</label>
            <value>CONSOLIDATE_LOANS</value>
        </item>
        <item>
            <label>Terveyden- tai hampaidenhoito</label>
            <value>DENTAL_OR_OTHER_CARE</value>
        </item>
        <item>
            <label>Opiskelu</label>
            <value>STUDIES</value>
        </item>
        <item>

```

```

        <label>Muu kulutus</label>
        <value>OTHER</value>
    </item>
</items>
<format>^
(MOTOR|WEDDING|BOAT|HOUSING_MORTGAGE_LOAN|INVESTMENT|DRIVERS_LICENSE|INTERIOR|RENOVATION|TRAVEL|CONSOLIDATE_LOAN
S|DENTAL_OR_OTHER_CARE|STUDIES|OTHER)$</format>
</element>

<element java-property="applicant.maritalStatus">
    <label>Siviilisääty</label>
    <description>Valitse siviilisääty valikon vaihtoehdista</description>
    <name>applicant-marital-status</name>
    <type>radio</type>
    <default></default>
    <is-mandatory>true</is-mandatory>
    <items>
        <item>
            <label>Avioliitto / Rekisteröity parisuhde</label>
            <value>MARRIED</value>
        </item>
        <item>
            <label>Naimaton</label>
            <value>SINGLE</value>
        </item>
        <item>
            <label>Avoliiitto</label>
            <value>DOMESTIC_PARTNER</value>
        </item>
        <item>
            <label>Eronnut</label>
            <value>DIVORCED</value>
        </item>
        <item>
            <label>Leski</label>
            <value>WIDOW</value>
        </item>
    </items>
    <format>^(MARRIED|DOMESTIC_PARTNER|SINGLE|DIVORCED|WIDOW)$</format>
</element>

<element java-property="applicant.habitationType">
    <label>Asumismuoto</label>
    <description>Valitse asumismuoto valikon vaihtoehdista</description>
    <name>applicant-habitation-form</name>
    <is-mandatory>true</is-mandatory>
    <type>radio</type>
    <default></default>
    <items>
        <item>
            <label>Omistusasunto</label>
            <value>DETACHED_HOUSE</value>
        </item>
        <item>
            <label>Vuokra-asunto</label>
            <value>APARTMENT</value>
        </item>
        <item>
            <label>Asumisoikeusasunto</label>
            <value>CONDOMINIUM</value>
        </item>
        <item>
            <label>Osaomistusasunto</label>
            <value>CONDOMINIUM_PARTLY_OWNED</value>
        </item>
        <item>
            <label>Alivuokralainen</label>
            <value>BY_PARENTS</value>
        </item>
        <item>
            <label>Työsuhdeasunto</label>

```

```

        <value>OFFICIAL_RESIDENCE</value>
    </item>
    <item>
        <label>Muu</label>
        <value>OTHER</value>
    </item>
</items>
<format>^
(DETACHED_HOUSE|APARTMENT|CONDOMINIUM|CONDOMINIUM_PARTLY OWNED|BY_PARENTS|OFFICIAL_RESIDENCE|OTHER)$</format>
</element>

<element java-property="applicant.habitationDate">
    <label>Asunut nykyisessä osoitteessa alkaen</label>
    <description>Valitse muuttovuosi- ja kuukausi.</description>
    <name>applicant-habitation-year</name>
    <is-mandatory>true</is-mandatory>
    <type>string</type>
    <length>4</length>
    <format>^([0-9]{2})(0[1-9]|1[012])$</format>
    <format-message>Pakollinen kenttä</format-message>
</element>

<element>
    <name>coapplicant-toggle</name>
    <type>checkbox</type>
    <label>Haen lainaa toisen henkilön kanssa</label>
    <is-mandatory>false</is-mandatory>
    <items>
        <item>
            <label></label>
            <value>true</value>
        </item>
    </items>
    <format>^(true| )$</format>
</element>

<element java-property="coApplicant.governmentId">
    <label>Henkilötunnus</label>
    <description>Täytä henkilötunnus muodossa PPKKVVYXXXX tai PPKKVVAXXXX.</description>
    <name>coapplicant-government-id</name>
    <is-mandatory>false</is-mandatory>
    <type>string</type>
    <length>11</length>
    <format>^(\d{6})\d{2}(\d{3})([0123456789ABCDEFHJKLMNPRSTUVWXY])$</format>
    <format-message>Kenttäään on täytettävä henkilötunnus muodossa PPKKVV-XXXX tai PPKKVVAXXXX!</format-
message>
</element>

<element java-property="coApplicant.mobilePhoneNumber">
    <label>Matkapuhelin</label>
    <description>Varmista, että matkapuhelinnumerosi on oikein.</description>
    <name>coapplicant-mobile-number</name>
    <is-mandatory>false</is-mandatory>
    <depends-on>coapplicant-government-id</depends-on>
    <type>string</type>
    <length>16</length>
    <format>^((\\+358|00358|0)[-| ]?(1[1-9]| [2-9]| [1][0]| [1-9]| 201| 2021| [2][0]| [2][4-9]| [2][0]| [3-8]| 29| [3][0]
[1-9]| 71| 73| [7][5]| [0][0]| [3-9]| [7][5]| [3][0]| [3-9]| [7][5]| [3][2]| [3-9]| [7][5]| [7][5]| [3-9]| [7][5]| [9][8]| [3-9]| [5][0]| [0-
9]| [0,2]| [4][0-9]| {1,3}| [-| ]?[0-9]| {3,10})$</format>
    <format-message>Tarkista puhelinnumerosi.</format-message>
</element>

<element java-property="coApplicant.emailAddress">
    <label>Sähköposti</label>
    <description>Varmista, että sähköpostiosoitteesi on oikein.</description>
    <name>coapplicant-email-address</name>
    <is-mandatory>false</is-mandatory>
    <depends-on>coapplicant-government-id</depends-on>
    <type>string</type>
    <length>50</length>
    <format>^([A-Za-z0-9!#%&'*+/=?^_`~-]+|([\\.[A-Za-z0-9!#%&'*+/=?^_`~-]+)+)@([A-Za-z0-9]+)(([\\.\\"-]?[a-zA-Z0-9]*|[\\.\\"-]?[0-9]*|[\\.\\"-]?
[a-zA-Z0-9]*@[a-zA-Z0-9]*\\.)+[a-zA-Z0-9]*@[a-zA-Z0-9]*\\.)$</format>
    <format-message>Oikea sähköpostiosoite</format-message>
</element>

```

```

z0-9]+*)\\(.([A-Za-z]{2,}))?$/</format>
    <format-message>Tarkista sähköpostiosoitteesi.</format-message>
</element>

<element java-property="coApplicant.lastMonthIncomeNet">
    <label>kk:n nettotulo</label>
    <description>Edeltävän kk:n nettotulo</description>
    <name>coapplicant-last-month-income-net</name>
    <is-mandatory>false</is-mandatory>
    <type>integer</type>
    <min-value>0</min-value>
    <max-value>99999</max-value>
    <unit>money</unit>
    <format-message>Tarkista kuukausitulosi (0-99 999 €).</format-message>
</element>

<element java-property="coApplicant.sixMonthsAverageIncomeNet">
    <label>6 kk:n nettotulo</label>
    <description>Edeltävän 6 kk:n keskimääriinen nettotulo</description>
    <name>coapplicant-six-months-average-income-net</name>
    <is-mandatory>false</is-mandatory>
    <type>integer</type>
    <min-value>0</min-value>
    <max-value>99999</max-value>
    <unit>money</unit>
    <format-message>Tarkista kuukausitulosi (0-99 999 €).</format-message>
</element>

<!-- ***** Huvudsökande - Arbete -->
<element java-property="applicant.employmentType">
    <label>Hakijan työsuhteen muoto</label>
    <description>Valitse työmuoto valikon vaihtoehtoista</description>
    <name>applicant-employment-type</name>
    <is-mandatory>true</is-mandatory>
    <type>radio</type>
    <default></default>
    <items>
        <item>
            <label>Vakituinen</label>
            <value>FAST_ANSTALLD</value>
        </item>
        <item>
            <label>Määrääikainen</label>
            <value>VIKARIAT</value>
        </item>
        <item>
            <label>Osa-aikainen</label>
            <value>PART_TIME</value>
        </item>
        <item>
            <label>Eläkkeellä</label>
            <value>PENSIONAR</value>
        </item>
        <item>
            <label>Työkyvyttömyys- tai varhaiseläkkeellä</label>
            <value>SJUKPENSIONAR</value>
        </item>
        <item>
            <label>Yrittäjä</label>
            <value>EGEN_FORETAGARE</value>
        </item>
        <item>
            <label>Opiskelija</label>
            <value>STUDENT</value>
        </item>
        <item>
            <label>Työtön</label>
            <value>ARBETSLOS</value>
        </item>
        <item>
            <label>Maanviljelijä</label>

```

```

        <value>FARMER</value>
    </item>
</items>
<format>
(FAST_ANSTALLD|VIKARIAT|PART_TIME|PENSIONAR|SJUKPENSIONAR|EGEN_FORETAGARE|STUDENT|ARBETSLOS|FARMER) $</format>
</element>

<element java-property="applicant.employedSince">
    <label>Nykyinen työsuhde alkoi</label>
    <description>Valitse aloitusvuosi ja -kuukausi.</description>
    <name>applicant-employment-year-from</name>
    <is-mandatory>false</is-mandatory>
    <type>string</type>
    <length>4</length>
    <format>^([0-9]{2})(0[1-9]|1[012])$</format>
    <format-message>Pakollinen kenttä</format-message>
</element>

<element java-property="applicant.employedTo">
    <label>Työsuhde päättyy</label>
    <description>Valitse päätymisvuosi ja -kuukausi.</description>
    <name>applicant-employment-year-until</name>
    <is-mandatory>false</is-mandatory>
    <type>string</type>
    <length>4</length>
    <format>^([0-9]{2})(0[1-9]|1[012])$</format>
    <format-message>Pakollinen kenttä</format-message>
</element>

<element java-property="applicant.employerName">
    <label>Työnantaja</label>
    <description>Ilmoita työnantajasi nimi</description>
    <name>applicant-employer</name>
    <is-mandatory>false</is-mandatory>
    <type>string</type>
    <length>100</length>
    <format>^[\s\S]{0,100}$</format>
    <format-message>Täydennä työntantajasi nimi</format-message>
</element>

<element java-property="applicant.profession">
    <label>Ammatti</label>
    <description>Ilmoita ammattisi</description>
    <name>applicant-profession</name>
    <is-mandatory>false</is-mandatory>
    <type>string</type>
    <length>100</length>
    <format>^[\s\S]{0,100}$</format>
    <format-message>Ilmoita ammattisi</format-message>
</element>

<element java-property="applicant.grossMonthlyIncome">
    <label>Omat kuukausitulot brutto</label>
    <description>Omat kuukausitulot ennen veroja.</description>
    <name>applicant-monthly-income-gross</name>
    <is-mandatory>true</is-mandatory>
    <type>integer</type>
    <min-value>0</min-value>
    <max-value>99999</max-value>
    <unit>money</unit>
    <format-message>Tarkista kuukausitulosi (0-99 999 €).</format-message>
</element>

<element java-property="applicant.lastMonthIncomeNet">
    <label>Hakijan kk:n nettotulo</label>
    <description>Edeltävän kk:n nettotulo</description>
    <name>applicant-last-month-income-net</name>
    <is-mandatory>false</is-mandatory>
    <type>integer</type>
    <min-value>0</min-value>
    <max-value>99999</max-value>
    <unit>money</unit>

```

```

<format-message>Tarkista kuukausitulosi (0-99 999 €).</format-message>
</element>

<element java-property="applicant.sixMonthsAverageIncomeNet">
    <label>Hakijan 6 kk:n nettotulo</label>
    <description>Edeltävän 6 kk:n keskimääräinen nettotulo</description>
    <name>applicant-six-months-average-income-net</name>
    <is-mandatory>false</is-mandatory>
    <type>integer</type>
    <min-value>0</min-value>
    <max-value>99999</max-value>
    <unit>money</unit>
    <format-message>Tarkista kuukausitulosi (0-99 999 €).</format-message>
</element>

<element java-property="householdIncome">
    <label>Talouden kuukausitulot netto</label>
    <description>Talouden yhteenlasketut kuukausitulot verojen jälkeen.</description>
    <name>household-monthly-income</name>
    <is-mandatory>true</is-mandatory>
    <type>integer</type>
    <min-value>0</min-value>
    <max-value>99999</max-value>
    <unit>money</unit>
    <format-message>Tarkista talouden kuukausitulot (0-99 999 €).</format-message>
</element>

<element java-property="householdLivingExpenses">
    <label>Omat asumiskustannukset kuukaudessa (sis. asuntolainan)</label>
    <description>Asumiskustannuksesi sisältäen osuutesi asuntolainan lyhennyksestä, vuokrasta/vastikkeesta, sähköstä ja lämmityksestä kuukaudessa.</description>
    <name>household-living-expenses</name>
    <is-mandatory>true</is-mandatory>
    <type>integer</type>
    <min-value>0</min-value>
    <max-value>9999</max-value>
    <unit>money</unit>
    <format-message>Tarkista omat asumiskustannuksesi (0-9 999 €).</format-message>
</element>

<element java-property="creditCardMasterCard">
    <label>MasterCard</label>
    <description>MasterCard</description>
    <name>credit-cards-master-card</name>
    <type>checkbox</type>
    <is-mandatory>false</is-mandatory>
    <items>
        <item>
            <label>Kyllä</label>
            <value>true</value>
        </item>
    </items>
    <format>>^(true|)${</format>
</element>

<element java-property="creditCardVisa">
    <label>VISA</label>
    <description>VISA</description>
    <name>credit-cards-visa</name>
    <type>checkbox</type>
    <is-mandatory>false</is-mandatory>
    <items>
        <item>
            <label>Kyllä</label>
            <value>true</value>
        </item>
    </items>
    <format>>^(true|)${</format>
</element>

<element java-property="creditCardAmericanExpress">

```

```

<label>American Express</label>
<description>American Express</description>
<name>credit-cards-american-express</name>
<type>checkbox</type>
<is-mandatory>false</is-mandatory>
<items>
    <item>
        <label>Kyllä</label>
        <value>true</value>
    </item>
</items>
<format>^(true|)${</format>
</element>

<element java-property="creditCardDinersClub">
    <label>Diners Club</label>
    <description>Diners Club</description>
    <name>credit-cards-diners-club</name>
    <type>checkbox</type>
    <is-mandatory>false</is-mandatory>
    <items>
        <item>
            <label>Kyllä</label>
            <value>true</value>
        </item>
    </items>
    <format>^(true|)${</format>
</element>

<element java-property="creditCardOthers">
    <label>Muut</label>
    <description>Muut</description>
    <name>credit-cards-others</name>
    <type>checkbox</type>
    <is-mandatory>false</is-mandatory>
    <items>
        <item>
            <label>Kyllä</label>
            <value>true</value>
        </item>
    </items>
    <format>^(true|)${</format>
</element>

<element java-property="mortgageDebt">
    <label>Oma osuus asuntolainasta</label>
    <description>Jäljellä oleva oma osuutesi koko asuntolainasta.</description>
    <name>household-mortgage-debt</name>
    <is-mandatory>true</is-mandatory>
    <type>integer</type>
    <min-value>0</min-value>
    <max-value>999999</max-value>
    <unit>money</unit>
    <format-message>Tarkista summa (0-999 999 €).</format-message>
</element>

<element java-property="householdLoanExpenses">
    <label>Omat muut lainat ja luottokorttilainat yhteensä</label>
    <description>Luottokorttilainat ja muut lainat kuin asuntolaina.</description>
    <name>household-other-loan-expenses</name>
    <is-mandatory>true</is-mandatory>
    <type>integer</type>
    <min-value>0</min-value>
    <max-value>999999</max-value>
    <unit>money</unit>
    <format-message>Tarkista summa (0-999 999 €).</format-message>
</element>

<element java-property="otherExpenses">
    <label>Omat lainojen lyhennykset kuukaudessa (ei asuntolaina)</label>
    <description>Muiden lainojen kuin asuntolainan lyhennykset kuukaudessa yhteensä.</description>

```

```

<name>other-expenses</name>
<is-mandatory>true</is-mandatory>
<type>integer</type>
<min-value>0</min-value>
<max-value>9999</max-value>
<unit>money</unit>
<format-message>Tarkista summa (0-9 999 €).</format-message>
</element>

<element java-property="mortgageTotalAmount">
    <label>Talouden kaikki lainat yhteensä</label>
    <description>Kanssasi samassa taloudessa asuvien kaikki lainat yhteensä.</description>
    <name>household-mortgage-total-amount</name>
    <is-mandatory>true</is-mandatory>
    <type>integer</type>
    <min-value>0</min-value>
    <max-value>999999</max-value>
    <unit>money</unit>
    <format-message>Tarkista summa (0-999 999 €).</format-message>
</element>

<element java-property="monthlyMortgageCost">
    <label>Talouden kaikkien lainojen lyhennykset kuukaudessa</label>
    <description>Kanssasi samassa taloudessa asuvien kaikkien lainojen lyhennykset kuukaudessa yhteensä.</description>
    <name>household-mortgage-monthly-cost</name>
    <is-mandatory>true</is-mandatory>
    <type>integer</type>
    <min-value>0</min-value>
    <max-value>9999</max-value>
    <unit>money</unit>
    <format-message>Tarkista summa (0-9 999 €).</format-message>
</element>

<element java-property="isFinancingOtherLoan">
    <label>Haluan yhdistää nykyisiä lainojaani</label>
    <name>finance-other-loans</name>
    <type>checkbox</type>
    <is-mandatory>false</is-mandatory>
    <items>
        <item>
            <label></label>
            <value>true</value>
        </item>
    </items>
    <format>^(true|)$</format>
</element>

<element java-property="nbrOfLoansToConsolidate">
    <label>Yhdistettävien lainojen lukumäärä</label>
    <description>Anna lunastettavien lainojen ja luottokorttien lukumäärä</description>
    <name>finance-other-loans-nbr-of-loans</name>
    <is-mandatory>false</is-mandatory>
    <type>integer</type>
    <min-value>1</min-value>
    <max-value>20</max-value>
    <unit>kpl</unit>
    <format-message>Voit yhdistää enintään 20 lainaa</format-message>
</element>

<element java-property="balanceOfConsolidatedLoans">
    <label>Yhdistettävien lainojen yhteissumma</label>
    <description>Anna yhdistettävien lainojen ja luottokorttilainojen yhteissumma</description>
    <name>finance-other-loans-balance-of-loans</name>
    <is-mandatory>false</is-mandatory>
    <type>integer</type>
    <min-value>0</min-value>
    <max-value>99999</max-value>
    <unit>money</unit>
    <format-message>Voit yhdistää lainoja (0 - 99 999 €).</format-message>
</element>
```

```

<element java-property="monthlyCostOfConsolidatedLoans">
    <label>Yhdistettävien lainojen lyhennykset kuukaudessa</label>
    <description>Anna lunastettavien lainojen ja luottokorttien yhteenlaskettu kuukausimaksu</description>
    <name>finance-other-loans-monthly-cost-of-loans</name>
    <is-mandatory>false</is-mandatory>
    <type>integer</type>
    <min-value>0</min-value>
    <max-value>9999</max-value>
    <unit>money</unit>
    <format-message>Yhdistettävien lainojen lyhennykset kuukaudessa voivat olla 0 - 9 999 €.</format-
message>
</element>

<element java-property="financeOtherLoansConsentToConsolidate">
    <label>Hyväksyn ja annan toimeksiannon yhdistettävien lainojeni poismaksuun</label>
    <description>Hyväksyn ja annan toimeksiannon yhdistettävien lainojeni poismaksuun</description>
    <name>finance-other-loans-consolidation-consent</name>
    <type>radio</type>
    <is-mandatory>false</is-mandatory>
    <items>
        <item>
            <label>Kyllä</label>
            <value>true</value>
        </item>
        <item>
            <label>Ei</label>
            <value>false</value>
        </item>
    </items>
    <format>^(true|false)$</format>
</element>

<element java-property="payoutAccountNo">
    <label>Tilinumero IBAN-muodossa</label>
    <name>payout-account-number</name>
    <is-mandatory>false</is-mandatory>
    <type>string</type>
    <length>18</length>
    <format>^[A-Z0-9]{18}$</format>
    <format-message>Tarkista tilinumerosi, sen pitäisi olla IBAN-muodossa: FI01234567890000000.</format-
message>
</element>

<element java-property="isInsured">
    <label>Haluan liittää lainaani Maksuvakuutuksen.</label>
    <description></description>
    <name>insurance</name>
    <type>radio</type>
    <is-mandatory>true</is-mandatory>
    <items>
        <item>
            <label>Kyllä</label>
            <value>true</value>
        </item>
        <item>
            <label>Ei</label>
            <value>false</value>
        </item>
    </items>
    <format>^(true|false)$</format>
</element>

<element>
    <type>checkbox</type>
    <name>campaign-code-toggle</name>
    <label>Lisää kampanjakoodi</label>
    <is-mandatory>false</is-mandatory>
    <items>
        <item>
            <label></label>

```

```

        <value>true</value>
    </item>
</items>
<format>^(true| )$</format>
</element>

<element java-property="campaignCode">
    <label>Kampanjakoodi</label>
    <description>Jos sinulla on kampanjakoodi, täytä se tähän.</description>
    <name>campaign-code</name>
    <is-mandatory>false</is-mandatory>
    <type>string</type>
    <length>10</length>
    <format>^([A-ZÄÖÜ\\-\\.\\\$\\\\d a-zAÖÜ]{2,10})?</format>
    <format-message>Ystäväällisesti tarkistakaa että kamppanjakoodi on oikein.</format-message>
</element>

<element java-property="confirmRegisterConsent">
    <name>confirm-register-concent</name>
    <label>Hyväksyn tietojeni käsittelyn</label>
    <description>Annan/Annamme toimeksiannon Resurs Bankille kerätä ja käsitellä muulta luotonantajilta saatavia tietoja luotoistani luottohakemuksen käsitellyä varten. Annan suostumukseni siihen, että luotonantajat luovuttavat minua koskevia tietoja luotoistani. Näitä tietoja Resurs Bank kysyy Suomen Asiakastieto Oy:n ylläpitämän kyselyjärjestelmän avulla järjestelmään osallistuvilta yrityksiltä.</description>
    <type>checkbox</type>
    <is-mandatory>false</is-mandatory>
    <items>
        <item>
            <label>Annan/Annamme toimeksiannon Resurs Bankille kerätä ja käsitellä muulta luotonantajilta saatavia tietoja luotoistani luottohakemuksen käsitellyä varten. Annan suostumukseni siihen, että luotonantajat luovuttavat minua koskevia tietoja luotoistani. Näitä tietoja Resurs Bank kysyy Suomen Asiakastieto Oy:n ylläpitämän kyselyjärjestelmän avulla järjestelmään osallistuvilta yrityksiltä.</label>
            <value>true</value>
        </item>
    </items>
    <format>^(true| )$</format>
</element>

<element java-property="confirmLegalDocuments">
    <name>confirm-legal-documents</name>
    <label>Vahvistan tietoni ja hyväksyn ehdot</label>
    <description>Vakuutan, että antamani tiedot ovat oikeat. Olen tutustunut Vakiomuotoiset eurooppalaiset kuluttajaluottotiedot -lomakkeeseen (VEK) sekä Resurs Bankin yksityisiä tili- ja korttiluottoja sekä yksityislainoja koskeviin yleisiin ehtoihin, ja hyväksyn ne.</description>
    <type>checkbox</type>
    <is-mandatory>true</is-mandatory>
    <items>
        <item>
            <label>Vakuutan, että antamani tiedot ovat oikeat. Olen tutustunut Vakiomuotoiset eurooppalaiset kuluttajaluottotiedot -lomakkeeseen (VEK) sekä Resurs Bankin yksityisiä tili- ja korttiluottoja sekä yksityislainoja koskeviin yleisiin ehtoihin, ja hyväksyn ne.</label>
            <value>true</value>
        </item>
    </items>
    <format-message>Sinun on hyväksyttävä ehdot edetä.</format-message>
    <format>^(true)$</format>
</element>

<element java-property="termsAgreedTo">
    <label>Annan suostumukseni henkilötietojeni käsittelyn</label>
    <description>
        Tehdessäsi sopimuksen Resurs Bankin kanssa sinut rekisteröidään Resurskonsernin asiakasrekisteriin. Pääasialliset henkilötietojen käyttötarkoituksemme ovat sopimuksen tekemisen edellyttämien tietojen kerääminen, tarkastaminen ja rekisteröinti sekä tekemäsi sopimuksen dokumentointi, hallinnointi ja toimeenpano. Henkilötietoja voidaan käsitellä myös markkinoinnissa. Kun henkilötietoja käsitellään markkinoittitarkoitukseissa, voidaan suorittaa myös profilointia sinulle soveltuviin tuote-ehdotusten tekemiseksi, jolloin sinulle voidaan tarjota vaihtoehtoisia tuotteitaamme tai voit saada yhteydenoton Resurs
    </description>
</element>

```

Bankin yhteistyökumppaneilta, mihin annan suostumukseni. Kattavat tiedot henkilötietojemme käsittelystä ja sinun oikeuksistasi saat henkilötietojen käsittelyä koskevista sopimusehdotamme. Ymmärrän, että vain kielää suoramarkkinoinnin ilmoittamalla asiasta Resurs Bankille.

Vakuutan, että annetut tiedot ovat oikein ja täydellisiä ja annan suostumukseni hakemukseni luottoharkintaan varten. Luotonantaja pitää itsellään vapaan harkintaoikeuden. Vakuutan lisäksi, että olen saanut tiedon ja tutustunut asiakirjoihin Vakiomuotoiset eurooppalaiset kuluttajaluottotiedot voimassaolevine erikoismaksuehtoineen sekä Resurs Bankin yksityisiä tili- ja korttiluottoja sekä yksityislainoja koskeviin yleisiin ehtoihin, mitkä sopimusehdot täten hyväksyn. Nämä asiakirjat ja asiakaskohtainen sopimus tallennetaan ja ovat saatavilla Omat Sivut - palvelussa www.resursbank.fi

```
</description>
<name>approve-conditions</name>
<is-mandatory>true</is-mandatory>
<type>confirmation</type>
<items>
    <item>
        <label>Allekirjoittaessasi sopimuksen Resurs Bankin kanssa tallennetaan henkilötietosi asiakasrekisteriin. Samalla annat suostumuksesi siihen, että luottohakemusta käsittellessä saadaan käyttää henkilötietojasi. Henkilötietojasi voidaan käyttää ja luovuttaa suoramarkkinointitarkoituksiin. Sinulla on oikeus kielää henkilötietojesi käsittely suoramarkkinointitarkoituksiin ilmoittamalla siitä Resurs Bankin asiakaspalveluun. Siinä tapauksessa, että hakemuksesi hylätään, Resurs Bankin yhteistyökumppani voi tarjota sinulle vastaavanlaista tuotetta.</label>
        <value>true</value>
    </item>
</items>
<format-message>Sinun on hyväksyttävä ehdot edetä.</format-message>
<format>^(true)$</format>
</element>

<element java-property="consentEmailMarketing">
    <name>consent-email-marketing</name>
    <label></label>
    <description>Annan suostumukseni siihen, että Resurs Bank voi lähettää minulle tarjouksia sähköpostitse.</description>
    <type>checkbox</type>
    <is-mandatory>false</is-mandatory>
    <items>
        <item>
            <label>Annan suostumukseni siihen, että Resurs Bank voi lähettää minulle tarjouksia sähköpostitse.</label>
            <value>true</value>
        </item>
    </items>
    <format>^(true| )$</format>
</element>

<element java-property="agreementSignType">
    <label>Valitse allekirjoitustapa</label>
    <description>Valitse allekirjoitustapa</description>
    <name>agreement-sign-type</name>
    <type>radio</type>
    <is-mandatory>true</is-mandatory>
    <items>
        <item>
            <label>Pankkitunnukilla</label>
            <value>E_SIGN</value>
        </item>
        <item>
            <label>Postitse</label>
            <value>MAIL</value>
        </item>
    </items>
    <format>^(MAIL|E_SIGN)$</format>
</element>

<!-- ***** mediaCode -->
<element java-property="mediaCode">
    <type>hidden</type>
    <name>mediacode</name>
    <is-mandatory>false</is-mandatory>
</element>
```

```
</resurs-form>
```

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns2:submitApplicationExtResponse xmlns:ns2="http://consumerloan.resurs.com/v1/msg/application" xmlns:ns3="http://consumerloan.resurs.com/v1/msg/exception">
      <submitApplicationExtResult>
        <applicationReference>lacca31b-c4c8-466a-9f7e-51c8c06591dd</applicationReference>
        <decision>APPROVED</decision>
        <approvedAmount>10000</approvedAmount>
        <interest>22.30</interest>
        <tenor>72</tenor>
        <effectiveInterest>25.31</effectiveInterest>
        <monthlyCost>259.00</monthlyCost>
        <consolidationDemand>1000.0</consolidationDemand>
        <adminFee>2.00</adminFee>
        <arrangementFee>39.00</arrangementFee>
        <documentTypes>ID</documentTypes>
        <documentTypes>PAYMENT_SLIP</documentTypes>
        <totalRepaymentAmount>18700.00</totalRepaymentAmount>
        <monthlyAccountFee>0</monthlyAccountFee>
        <totalFeesAndInterest>8700.00</totalFeesAndInterest>
          <topUpAmount>9000.0</topUpAmount>
          <monthlyCostTopUp>169.0</monthlyCostTopUp>
      </submitApplicationExtResult>
    </ns2:submitApplicationExtResponse>
  </soap:Body>
</soap:Envelope>
```

submitApplicationExt NO

NO

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:app="http://consumerloan.resurs.com/v1/msg/application">
  <soapenv:Header/>
  <soapenv:Body>
    <app:submitApplicationExt>
      <xml><![CDATA[<resurs-response>
<amount>100000</amount>
<time>120</time>
<applicant-government-id>05128727908</applicant-government-id>
<applicant-mobile-number>22563733</applicant-mobile-number>
<applicant-email-address>test@resurs.se</applicant-email-address>
<purpose-of-loan>HOUSING_MORTGAGE_LOAN</purpose-of-loan>
<applicant-marital-status>MARRIED</applicant-marital-status>
<number-children-not-of-age>1</number-children-not-of-age>
<applicant-habitation-form>DETACHED_HOUSE</applicant-habitation-form>
<applicant-employment-type>FAST_ANSTALLD</applicant-employment-type>
<applicant-nationality>NORWEGIAN</applicant-nationality>
<applicant-monthly-income-gross>32000</applicant-monthly-income-gross>
<applicant-monthly-income-net>27000</applicant-monthly-income-net>
<applicant-mortgage-debt>1200</applicant-mortgage-debt>
<applicant-car-loan-sum>10000</applicant-car-loan-sum>
<applicant-student-loan-sum>10000</applicant-student-loan-sum>
<applicant-credit-card-sum>3200</applicant-credit-card-sum>
<applicant-monthly-debt-repayment>5000</applicant-monthly-debt-repayment>
<finance-other-loans>true</finance-other-loans>
<approve-conditions>true</approve-conditions>
<agreement-sign-type>MAIL</agreement-sign-type>
</resurs-response>]]></xml>
      <consumerIp>1.2.3.4</consumerIp>
      <!--Optional:<br/>
      <externalReference>onboarding1</externalReference>
    </app:submitApplicationExt>
  </soapenv:Body>
</soapenv:Envelope>
```

```
<resurs-form>

  <!-- ***** Lånebelopp -->
  <element java-property="amount">
    <label>Lånebeløp</label>
    <description></description>
    <name>amount</name>
    <is-mandatory>true</is-mandatory>
    <type>integer</type>
    <min-value>1000</min-value>
    <max-value>500000</max-value>
    <unit>money</unit>
  </element>

  <element java-property="paymentTerms">
    <label>Nedbetalingstid</label>
    <description></description>
    <name>time</name>
    <type>list</type>
    <is-mandatory>true</is-mandatory>
    <unit>time</unit>
    <items>
      <item>
        <label>1</label>
        <value>12</value>
      </item>
      <item>
        <label>2</label>
        <value>24</value>
      </item>
    </items>
  </element>
```

```

        </item>
        <item>
            <label>3</label>
            <value>36</value>
        </item>
        <item>
            <label>4</label>
            <value>48</value>
        </item>
        <item>
            <label>5</label>
            <value>60</value>
        </item>
        <item>
            <label>6</label>
            <value>72</value>
        </item>
        <item>
            <label>7</label>
            <value>84</value>
        </item>
        <item>
            <label>8</label>
            <value>96</value>
        </item>
        <item>
            <label>10</label>
            <value>120</value>
        </item>
        <item>
            <label>12</label>
            <value>144</value>
        </item>
    </items>
    <format>^(12|24|36|48|60|72|84|96|120|144)$</format>
</element>

<!-- ***** Huvudsökande - Personalia --&gt;
&lt;element java-property="applicant.govtmentId"&gt;
    &lt;label&gt;Fødselsnummer&lt;/label&gt;
    &lt;description&gt;Oppgi ditt fødselsnummer (format DDMMÅÅXXXXXX).&lt;/description&gt;
    &lt;name&gt;applicant-government-id&lt;/name&gt;
    &lt;is-mandatory&gt;true&lt;/is-mandatory&gt;
    &lt;type&gt;string&lt;/type&gt;
    &lt;length&gt;11&lt;/length&gt;
    &lt;format&gt;^([0][1-9]| [1-2][0-9]| [3][0-1])(0[1-9]| 1[0-2]) ([0-9]{2}) ([0-9]{5})$&lt;/format&gt;
    &lt;format-message&gt;Ugyldig fødselsnummer. Oppgi ditt fødselsnummer i formatet DDMMÅÅXXXXXX.&lt;/format-message&gt;
&lt;/element&gt;

&lt;element java-property="applicant.mobilePhoneNumber"&gt;
    &lt;label&gt;Mobilnummer&lt;/label&gt;
    &lt;description&gt;Oppgi ditt mobil telefonnummer.&lt;/description&gt;
    &lt;name&gt;applicant-mobile-number&lt;/name&gt;
    &lt;is-mandatory&gt;true&lt;/is-mandatory&gt;
    &lt;type&gt;string&lt;/type&gt;
    &lt;length&gt;16&lt;/length&gt;
    &lt;format&gt;^(\ \ +47|0047|)?[-]?[2-9]( [-]?[0-9]) {7,7} )? $&lt;/format&gt;
    &lt;format-message&gt;Oppgi ditt mobil telefonnummer.&lt;/format-message&gt;
&lt;/element&gt;

&lt;element java-property="applicant emailAddress"&gt;
    &lt;label&gt;E-postadresse&lt;/label&gt;
    &lt;description&gt;Oppgi din e-postadresse.&lt;/description&gt;
    &lt;name&gt;applicant-email-address&lt;/name&gt;
    &lt;is-mandatory&gt;true&lt;/is-mandatory&gt;
    &lt;type&gt;string&lt;/type&gt;
    &lt;length&gt;50&lt;/length&gt;
    &lt;format&gt;^[_A-Za-z0-9!#%&amp;'*+/=?^`~-]+(\ .[_A-Za-z0-9!#%&amp;'*+/=?^`~-]+)*@([A-Za-z0-9]+)([[\ \ .\ \ -]?[a-zA-Z0-9]+)*\ .([A-Za-z]{2,})$&lt;/format&gt;
    &lt;format-message&gt;Oppgi din e-postadresse.&lt;/format-message&gt;
&lt;/element&gt;</pre>

```

```

<!-- ***** Medsökande - Personalia -->

<element java-property="coApplicant.governmentId">
    <label>Fødselsnummer</label>
    <description>Oppgi ditt fødselsnummer (format DDMMÅXXXXX).</description>
    <name>coapplicant-government-id</name>
    <is-mandatory>false</is-mandatory>
    <type>string</type>
    <length>11</length>
    <format>^([0][1-9][1-2][0-9][3[0-1])(0[1-9]|1[0-2])([0-9]{2})([0-9]{5})$</format>
    <format-message>Ugyldig fødselsnummer. Oppgi ditt fødselsnummer i formatet DDMMÅXXXXX.</format-message>
</element>

<element java-property="coApplicant.mobilePhoneNumber">
    <label>Mobilnummer</label>
    <description>Oppgi ditt mobil telefonnummer.</description>
    <name>coapplicant-mobile-number</name>
    <is-mandatory>false</is-mandatory>
    <depends-on>coapplicant-government-id</depends-on>
    <type>string</type>
    <length>16</length>
    <format>^((\+47|0047|)?[-]?[2-9]( [-]?[0-9])?{7,7})?</format>
    <format-message>Oppgi ditt mobil telefonnummer.</format-message>
</element>

<element java-property="coApplicant.emailAddress">
    <label>E-postadresse</label>
    <description>Oppgi din e-postadresse.</description>
    <name>coapplicant-email-address</name>
    <is-mandatory>false</is-mandatory>
    <depends-on>coapplicant-government-id</depends-on>
    <type>string</type>
    <length>50</length>
    <format>^([A-Za-z0-9!#%&'*+/=?^`~-]+(\.\.[A-Za-z0-9!#%&'*+/=?^`~-]+)*@[([A-Za-z0-9]+)([\\.\\\-]?[a-zA-Z0-9]+)*\\.\.([A-Za-z]{2,})$</format>
    <format-message>Oppgi din e-postadresse.</format-message>
</element>

<element java-property="coApplicant.consentEmailMarketing">
    <name>coapplicant-consent-email-marketing</name>
    <label>Jeg ønsker fremtidige tilbud fra Resurs Bank på e-post.</label>
    <type>checkbox</type>
    <is-mandatory>false</is-mandatory>
    <items>
        <item>
            <label></label>
            <value>true</value>
        </item>
    </items>
    <format>^(true|)$</format>
</element>

<element java-property="coApplicant.employmentType">
    <label>Arbeidsforhold</label>
    <description></description>
    <name>coapplicant-employment-type</name>
    <is-mandatory>false</is-mandatory>
    <depends-on>coapplicant-government-id</depends-on>
    <type>radio</type>
    <default></default>
    <items>
        <item>
            <label>Fast ansatt</label>
            <value>FAST_ANSTALLD</value>
        </item>
        <item>
            <label>Midlertidig ansatt/vikar</label>
            <value>VIKARIAT</value>
        </item>
        <item>

```

```

        <label>Selvstendig næringsdrivende</label>
        <value>EGEN_FORETAGARE</value>
    </item>
    <item>
        <label>Pensjonist</label>
        <value>PENSIONAR</value>
    </item>
    <item>
        <label>Student</label>
        <value>STUDENT</value>
    </item>
    <item>
        <label>Uføretrygd</label>
        <value>SJUKPENSIONAR</value>
    </item>
    <item>
        <label>Arbeidsledig</label>
        <value>ARBETSLOS</value>
    </item>
    <item>
        <label>Arbeidsavklaringspenger</label>
        <value>REHABILITATION</value>
    </item>
    <item>
        <label>Annet</label>
        <value>LONG_TERM_SICKNESS</value>
    </item>
</items>
<format>^
(FAST_ANSTALLD|VIKARIAT|EGEN_FORETAGARE|PENSIONAR|STUDENT|SJUKPENSIONAR|ARBETSLOS|REHABILITATION|LONG_TERM_SICKNESS)$</format>
<format-message>Velg et av alternativene.</format-message>
</element>

<element java-property="coApplicant.grossMonthlyIncome">
    <label>Månedsinntekt før skatt </label>
    <description>Oppgi din månedsinntekt før skatt.</description>
    <name>coapplicant-monthly-income-gross</name>
    <is-mandatory>false</is-mandatory>
    <depends-on>coapplicant-government-id</depends-on>
    <type>integer</type>
    <min-value>0</min-value>
    <max-value>99999999</max-value>
    <unit>money</unit>
    <format-message>Oppgi gyldig verdi (0 - 9999999 kr).</format-message>
</element>

<element java-property="coApplicant.netMonthlyIncome">
    <label>Månedsinntekt etter skatt</label>
    <description>Oppgi din månedsinntekt etter skatt.</description>
    <name>coapplicant-monthly-income-net</name>
    <is-mandatory>false</is-mandatory>
    <depends-on>coapplicant-government-id</depends-on>
    <type>integer</type>
    <min-value>0</min-value>
    <max-value>99999999</max-value>
    <unit>money</unit>
    <format-message>Oppgi gyldig verdi (0 - 9999999 kr).</format-message>
</element>

<element java-property="coApplicant.mortgageDebt">
    <label>Bolig/hytte lån - lånesaldo (medsøkers andel)</label>
    <description>Fyll ut hvor mye medsøker har i boliglån (kun medsøkers andel dersom medsøker deler med noen andre)..</description>
    <name>coapplicant-mortgage-debt</name>
    <is-mandatory>false</is-mandatory>
    <default>0</default>
    <type>integer</type>
    <min-value>0</min-value>
    <max-value>99999999</max-value>
    <unit>money</unit>

```

```

<length>8</length>
<format-message>Oppgi gyldig verdi (0 - 99999999 kr).</format-message>
</element>

<element java-property="coApplicant.mortgageMonthlyCost">
    <label>Bolig/hytte lån - månedskostnad (medsøkers andel)</label>
    <description>Fyll ut hvor mye medsøker betaler på boliglån hver måned (kun medsøkers andel dersom medsøker deler med noen andre).</description>
    <name>coapplicant-mortgage-monthly-cost</name>
    <is-mandatory>false</is-mandatory>
    <default>0</default>
    <type>integer</type>
    <min-value>0</min-value>
    <max-value>99999999</max-value>
    <unit>money</unit>
    <length>8</length>
    <format-message>Oppgi gyldig verdi (0 - 99999999 kr).</format-message>
</element>

<element java-property="coApplicant.carLoanSum">
    <label>Bil/MC/Båt - lånesaldo (medsøkers andel)</label>
    <description>Fyll ut hvor mye medsøker har i lån med pant I bil eller andre kjøretøy (kun medsøkers andel dersom medsøker deler med noen andre).</description>
    <name>coapplicant-car-loan-sum</name>
    <is-mandatory>false</is-mandatory>
    <default>0</default>
    <type>integer</type>
    <min-value>0</min-value>
    <max-value>9999999</max-value>
    <unit>money</unit>
    <length>7</length>
    <format-message>Oppgi gyldig verdi (0 - 9999999 kr).</format-message>
</element>

<element java-property="coApplicant.carLoanMonthlyCost">
    <label>Bil/MC/Båt - månedskostnad (medsøkers andel)</label>
    <description>Fyll ut hvor mye medsøker betaler på lån med pant I bil eller andre kjøretøy hver måned (kun medsøkers andel dersom medsøker deler med noen andre).</description>
    <name>coapplicant-car-loan-monthly-cost</name>
    <is-mandatory>false</is-mandatory>
    <default>0</default>
    <type>integer</type>
    <min-value>0</min-value>
    <max-value>9999999</max-value>
    <unit>money</unit>
    <length>7</length>
    <format-message>Oppgi gyldig verdi (0 - 9999999 kr).</format-message>
</element>

<element java-property="coApplicant.studentLoanSum">
    <label>Studielån - lånesaldo</label>
    <description>Fyll ut hvor mye medsøker har i studielån</description>
    <name>coapplicant-student-loan-sum</name>
    <is-mandatory>false</is-mandatory>
    <default>0</default>
    <type>integer</type>
    <min-value>0</min-value>
    <max-value>9999999</max-value>
    <unit>money</unit>
    <length>7</length>
    <format-message>Oppgi gyldig verdi (0 - 9999999 kr).</format-message>
</element>

<element java-property="coApplicant.studentLoanMonthlyCost ">
    <label>Studielån - månedskostnad</label>
    <description>Fyll ut hvor mye medsøker betaler på studielånet hver måned.</description>
    <name>coapplicant-student-loan-monthly-cost</name>
    <is-mandatory>false</is-mandatory>
    <default>0</default>
    <type>integer</type>
    <min-value>0</min-value>

```

```

<max-value>9999999</max-value>
<unit>money</unit>
<length>7</length>
<format-message>Oppgi gyldig verdi (0 - 9999999 kr).</format-message>
</element>

<element java-property="coApplicant.creditCardSum">
    <label>Kredittkortgjeld/forbrukslån - lånesaldo</label>
    <description>Fyll ut medsøkers samlede kredittkortgjeld/forbrukslån (inkluder ubenyttede kreditter dersom medsøker har dette)</description>
    <name>coapplicant-credit-card-sum</name>
    <is-mandatory>false</is-mandatory>
    <default>0</default>
    <type>integer</type>
    <min-value>0</min-value>
    <max-value>9999999</max-value>
    <unit>money</unit>
    <length>7</length>
    <format-message>Oppgi gyldig verdi (0 - 9999999 kr).</format-message>
</element>

<element java-property="coApplicant.creditCardMonthlyCost">
    <label>Kredittkortgjeld/forbrukslån - månedskostnad</label>
    <description>Fyll ut hvor mye medsøker betaler på hver måned på medsøkers kredittkortgjeld og forbrukslån.</description>
    <name>coapplicant-credit-card-monthly-cost</name>
    <is-mandatory>false</is-mandatory>
    <default>0</default>
    <type>integer</type>
    <min-value>0</min-value>
    <max-value>9999999</max-value>
    <unit>money</unit>
    <length>7</length>
    <format-message>Oppgi gyldig verdi (0 - 9999999 kr).</format-message>
</element>

<element java-property="loanPurpose">
    <label>Formålet med lånet</label>
    <description></description>
    <name>purpose-of-loan</name>
    <type>radio</type>
    <is-mandatory>true</is-mandatory>
    <items>
        <item>
            <label>Bil/MC</label>
            <value>MOTOR</value>
        </item>
        <item>
            <label>Bryllup</label>
            <value>WEDDING</value>
        </item>
        <item>
            <label>Båt</label>
            <value>BOAT</value>
        </item>
        <item>
            <label>Investering (f. eks aksjer)</label>
            <value>INVESTMENT</value>
        </item>
        <item>
            <label>Førerkort</label>
            <value>DRIVERS_LICENSE</value>
        </item>
        <item>
            <label>Innredning (f.eks møbler, elektronikk)</label>
            <value>INTERIOR</value>
        </item>
        <item>
            <label>Oppussing/Renovering</label>
            <value>RENOVATION</value>
        </item>
    </items>
</element>

```

```

<item>
    <label>Reise/Fritid</label>
    <value>TRAVEL</value>
</item>
<item>
    <label>Samle lån/kreditt</label>
    <value>CONSOLIDATE_LOANS</value>
</item>
<item>
    <label>Helse/tannpleie</label>
    <value>DENTAL_OR_OTHER_CARE</value>
</item>
<item>
    <label>Studier</label>
    <value>STUDIES</value>
</item>
<item>
    <label>Øvrig konsum</label>
    <value>OTHER</value>
</item>
<item>
    <label>Kjøp av bolig</label>
    <value>HOUSING_MORTGAGE_LOAN</value>
</item>
</items>
<format>^
(MOTOR|WEDDING|BOAT|HOUSING_MORTGAGE_LOAN|INVESTMENT|DRIVERS_LICENSE|INTERIOR|RENOVATION|TRAVEL|CONSOLIDATE_LOAN
S|DENTAL_OR_OTHER_CARE|STUDIES|OTHER)$</format>
<format-message>Velg et av alternativene.</format-message>
</element>

<!-- ***** Huvudsökande - Civilstånd --&gt;

&lt;element java-property="applicant.maritalStatus"&gt;
    &lt;label&gt;Sivilstatus&lt;/label&gt;
    &lt;description&gt;&lt;/description&gt;
    &lt;name&gt;applicant-marital-status&lt;/name&gt;
    &lt;type&gt;radio&lt;/type&gt;
    &lt;default&gt;&lt;/default&gt;
    &lt;is-mandatory&gt;true&lt;/is-mandatory&gt;
    &lt;items&gt;
        &lt;item&gt;
            &lt;label&gt;Gift/Registrert partnerskap&lt;/label&gt;
            &lt;value&gt;MARRIED&lt;/value&gt;
        &lt;/item&gt;
        &lt;item&gt;
            &lt;label&gt;Samboer&lt;/label&gt;
            &lt;value&gt;DOMESTIC_PARTNER&lt;/value&gt;
        &lt;/item&gt;
        &lt;item&gt;
            &lt;label&gt;Skilt/separert&lt;/label&gt;
            &lt;value&gt;DIVORCED&lt;/value&gt;
        &lt;/item&gt;
        &lt;item&gt;
            &lt;label&gt;Aleneboende&lt;/label&gt;
            &lt;value&gt;SINGLE&lt;/value&gt;
        &lt;/item&gt;
        &lt;item&gt;
            &lt;label&gt;Enke-/mann&lt;/label&gt;
            &lt;value&gt;WIDOW&lt;/value&gt;
        &lt;/item&gt;
    &lt;/items&gt;
    &lt;format&gt;^(MARRIED|DOMESTIC_PARTNER|SINGLE|DIVORCED|WIDOW)$&lt;/format&gt;
    &lt;format-message&gt;Velg et av alternativene.&lt;/format-message&gt;
&lt;/element&gt;

&lt;element java-property="householdNoOfChildren"&gt;
    &lt;label&gt;Antall barn under 18 år&lt;/label&gt;
    &lt;description&gt;Om du ikke har barn, oppgi 0.&lt;/description&gt;
    &lt;name&gt;number-children-not-of-age&lt;/name&gt;
    &lt;is-mandatory&gt;true&lt;/is-mandatory&gt;
</pre>

```

```
<type>list</type>
<items>
  <item>
    <label>0</label>
    <value>0</value>
  </item>
  <item>
    <label>1</label>
    <value>1</value>
  </item>
  <item>
    <label>2</label>
    <value>2</value>
  </item>
  <item>
    <label>3</label>
    <value>3</value>
  </item>
  <item>
    <label>4</label>
    <value>4</value>
  </item>
  <item>
    <label>5</label>
    <value>5</value>
  </item>
  <item>
    <label>6</label>
    <value>6</value>
  </item>
  <item>
    <label>7</label>
    <value>7</value>
  </item>
  <item>
    <label>8</label>
    <value>8</value>
  </item>
  <item>
    <label>9</label>
    <value>9</value>
  </item>
  <item>
    <label>10</label>
    <value>10</value>
  </item>
  <item>
    <label>11</label>
    <value>11</value>
  </item>
  <item>
    <label>12</label>
    <value>12</value>
  </item>
  <item>
    <label>13</label>
    <value>13</value>
  </item>
  <item>
    <label>14</label>
    <value>14</value>
  </item>
  <item>
    <label>15</label>
    <value>15</value>
  </item>
  <item>
    <label>16</label>
    <value>16</value>
  </item>
<item>
```

```

        <label>17</label>
        <value>17</value>
    </item>
    <item>
        <label>18</label>
        <value>18</value>
    </item>
    <item>
        <label>19</label>
        <value>19</value>
    </item>
    <item>
        <label>20</label>
        <value>20</value>
    </item>
</items>
<unit>kpl</unit>
<format>^([01]?[0-9]|20)$</format>
<format-message>Oppgi et gyldig antall barn under 18 (mellan 0 og 20).</format-message>
</element>

<element java-property="applicant.habitationType">
<label>Boform</label>
<description></description>
<name>applicant-habitation-form</name>
<is-mandatory>true</is-mandatory>
<type>radio</type>
<default></default>
<items>
    <item>
        <label>Eier leilighet/bolig</label>
        <value>DETACHED_HOUSE</value>
    </item>
    <item>
        <label>Borettslag</label>
        <value>CONDOMINIUM</value>
    </item>
    <item>
        <label>Leier</label>
        <value>APARTMENT</value>
    </item>
    <item>
        <label>Bor hos foreldre</label>
        <value>BY_PARENTS</value>
    </item>
</items>
<format>^(DETACHED_HOUSE|APARTMENT|CONDOMINIUM|BY_PARENTS)$</format>
<format-message>Velg et av alternativene.</format-message>
</element>

<!-- ***** Huvudsökande - Arbete -->
<element java-property="applicant.employmentType">
<label>Arbeidsforhold</label>
<description></description>
<name>applicant-employment-type</name>
<is-mandatory>true</is-mandatory>
<type>radio</type>
<default></default>
<items>
    <item>
        <label>Fast ansatt</label>
        <value>FAST_ANSTALLD</value>
    </item>
    <item>
        <label>Midlertidig ansatt/vikar</label>
        <value>VIKARIAT</value>
    </item>
    <item>
        <label>Selvstendig næringsdrivende</label>
        <value>EGEN_FORETAGARE</value>
    </item>
</items>

```

```

<item>
    <label>Pensjonist</label>
    <value>PENSIONAR</value>
</item>
<item>
    <label>Student</label>
    <value>STUDENT</value>
</item>
<item>
    <label>Uføreretrygd</label>
    <value>SJUKPENSIONAR</value>
</item>
<item>
    <label>Arbeidsledig</label>
    <value>ARBETSLOS</value>
</item>
<item>
    <label>Arbeidsavklaringspenger</label>
    <value>REHABILITATION</value>
</item>
<item>
    <label>Annet</label>
    <value>LONG_TERM_SICKNESS</value>
</item>
</items>
<format>^
(FAST_ANSTALLD|VIKARIAT|EGEN_FORETAGARE|PENSIONAR|STUDENT|SJUKPENSIONAR|ARBETSLOS|REHABILITATION|LONG_TERM_SICKNESS)$</format>
<format-message>Velg et av alternativene.</format-message>
</element>

<element java-property="applicant.citizenship">
    <label>Norsk statsborger</label>
    <description>Stasborgerskap</description>
    <name>applicant-nationality</name>
    <type>radio</type>
    <default></default>
    <is-mandatory>true</is-mandatory>
    <items>
        <item>
            <label>Norsk</label>
            <value>NORWEGIAN</value>
        </item>
        <item>
            <label>Utenlandske</label>
            <value>FOREIGN</value>
        </item>
    </items>
    <format>^(NORWEGIAN|FOREIGN)$</format>
</element>

<element java-property="applicant.grossMonthlyIncome">
    <label>Månedsinntekt før skatt </label>
    <description>Oppgi din månedsinntekt før skatt.</description>
    <name>applicant-monthly-income-gross</name>
    <is-mandatory>true</is-mandatory>
    <type>integer</type>
    <min-value>0</min-value>
    <max-value>99999999</max-value>
    <unit>money</unit>
    <format-message>Oppgi gyldig verdi (0 - 9999999 kr).</format-message>
</element>

<element java-property="applicant.netMonthlyIncome">
    <label>Månedsinntekt etter skatt</label>
    <description>Oppgi din månedsinntekt etter skatt.</description>
    <name>applicant-monthly-income-net</name>
    <is-mandatory>true</is-mandatory>
    <type>integer</type>
    <min-value>0</min-value>
    <max-value>99999999</max-value>

```

```

<unit>money</unit>
<format-message>Oppgi gyldig verdi (0 - 9999999 kr).</format-message>
</element>

<element java-property="applicant.otherMonthlyIncome">
  <label>Annen månedsinntekt</label>
  <description>Oppgi ekstra månedsinntekt som barnebidrag, leieinntekter etc</description>
  <name>applicant-monthly-income-other</name>
  <is-mandatory>false</is-mandatory>
  <type>integer</type>
  <min-value>0</min-value>
  <max-value>9999999</max-value>
  <unit>money</unit>
  <format-message>Oppgi gyldig verdi (0 - 9999999 kr).</format-message>
</element>

<element java-property="applicant.monthlyRentalCost">
  <label>Husleie/Fellesutgifter pr måned</label>
  <description>Oppgi dine totale månedlige husleie/fellesutgifter per måned.</description>
  <name>applicant-monthly-rental-cost</name>
  <is-mandatory>false</is-mandatory>
  <type>integer</type>
  <min-value>0</min-value>
  <max-value>9999999</max-value>
  <format-message></format-message>
</element>

<element java-property="applicant.mortgageDebt">
  <label>Bolig/hytte lån lånesaldo</label>
  <description>Husholdningens samlede lån</description>
  <name>applicant-mortgage-debt</name>
  <is-mandatory>true</is-mandatory>
  <type>integer</type>
  <min-value>0</min-value>
  <max-value>99999999</max-value>
  <unit>nok</unit>
  <format-message>Oppgi gyldig verdi (0 - 99999999 kr).</format-message>
</element>

<element java-property="applicant.mortgageMonthlyCost">
  <label>Bolig/hytte lån - månedskostnad (din andel)</label>
  <description>Fyll ut hvor mye du betaler på boliglån hver måned (kun din andel dersom du deler med noen andre).</description>
  <name>applicant-mortgage-monthly-cost</name>
  <is-mandatory>false</is-mandatory>
  <default>0</default>
  <type>integer</type>
  <min-value>0</min-value>
  <max-value>99999999</max-value>
  <unit>money</unit>
  <length>8</length>
  <format-message>Oppgi gyldig verdi (0 - 99999999 kr).</format-message>
</element>

<element java-property="consolidatedLoansTenor">
  <label>Nedbetalingstid på lån du skal refinansiere</label>
  <description>Om flere lån, gjennomsnittlig ant år legges inn</description>
  <name>consolidated-loans-tenor</name>
  <is-mandatory>false</is-mandatory>
  <min-value>0</min-value>
  <max-value>9999999</max-value>
  <type>integer</type>
</element>

<element java-property="applicant.mortgageDebtCeiling">
  <label>Kredittramme bolig/hytte lån</label>
  <description>Sett inn den avtalte øvre lånegrensen for din kredittramme</description>
  <name>applicant-mortgage-debt-ceiling</name>
  <is-mandatory>false</is-mandatory>
  <type>integer</type>
  <min-value>0</min-value>

```

```

<max-value>49999999</max-value>
<format-message>Oppgi gyldig verdi (0 - 49.999.999 kr).</format-message>
</element>

<element java-property="applicant.creditCardDebtCeiling">
    <label>Kredittramme forbrukslån og kredittkort</label>
    <description>Sett inn den avtalte øvre kredittgrensen for kredittkort/forbrukslån</description>
    <name>applicant-credit-card-debt-ceiling</name>
    <is-mandatory>false</is-mandatory>
    <type>integer</type>
    <min-value>0</min-value>
    <max-value>49999999</max-value>
    <format-message>Oppgi gyldig verdi (0 - 4.999.999 kr).</format-message>
</element>

<element java-property="applicant.carLoanSum">
    <label>Bil/MC/båtlån lånesaldo</label>
    <description>Husholdningens samlede lån</description>
    <name>applicant-car-loan-sum</name>
    <is-mandatory>true</is-mandatory>
    <type>integer</type>
    <min-value>0</min-value>
    <max-value>99999999</max-value>
    <unit>nok</unit>
    <format-message>Oppgi gyldig verdi (0 - 9999999 kr).</format-message>
</element>

<element java-property="applicant.carLoanMonthlyCost">
    <label>Bil/MC/Båt - månedskostnad (din andel)</label>
    <description>Fyll ut hvor mye du betaler på lån med pant i bil eller andre kjøretøy hver måned (kun din andel dersom du deler med noen andre).</description>
    <name>applicant-car-loan-monthly-cost</name>
    <is-mandatory>false</is-mandatory>
    <default>0</default>
    <type>integer</type>
    <min-value>0</min-value>
    <max-value>99999999</max-value>
    <unit>money</unit>
    <length>7</length>
    <format-message>Oppgi gyldig verdi (0 - 9999999 kr).</format-message>
</element>

<element java-property="applicant.studentLoanSum">
    <label>Studielån lånesaldo</label>
    <description>Søker sin(e) samlede studielån</description>
    <name>applicant-student-loan-sum</name>
    <is-mandatory>true</is-mandatory>
    <type>integer</type>
    <min-value>0</min-value>
    <max-value>99999999</max-value>
    <unit>nok</unit>
    <format-message>Oppgi gyldig verdi (0 - 9999999 kr).</format-message>
</element>

<element java-property="applicant.studentLoanMonthlyCost ">
    <label>Studielån - månedskostnad</label>
    <description>Fyll ut hvor mye du betaler på studielånet hver måned.</description>
    <name>applicant-student-loan-monthly-cost</name>
    <is-mandatory>false</is-mandatory>
    <default>0</default>
    <type>integer</type>
    <min-value>0</min-value>
    <max-value>99999999</max-value>
    <unit>money</unit>
    <length>7</length>
    <format-message>Oppgi gyldig verdi (0 - 9999999 kr).</format-message>
</element>

<element java-property="applicant.creditCardSum">
    <label>Kredittkortgjeld/forbruslån lånesaldo</label>
    <description>Søker sin(e) samlede kredittkort-/forbrukslån gjeld</description>

```

```

<name>applicant-credit-card-sum</name>
<is-mandatory>true</is-mandatory>
<type>integer</type>
<min-value>0</min-value>
<max-value>9999999</max-value>
<unit>nok</unit>
<format-message>Oppgi gyldig verdi (0 - 9999999 kr).</format-message>
</element>

<element java-property="applicant.creditCardMonthlyCost">
    <label>Kredittkortgjeld/forbrukslån - månedskostnad</label>
    <description>Fyll ut hvor mye du betaler hver måned på din kredittkortgjeld og dine forbrukslån.</description>
        <name>applicant-credit-card-monthly-cost</name>
        <is-mandatory>false</is-mandatory>
        <default>0</default>
        <type>integer</type>
        <min-value>0</min-value>
        <max-value>9999999</max-value>
        <unit>money</unit>
        <length>7</length>
        <format-message>Oppgi gyldig verdi (0 - 9999999 kr).</format-message>
</element>

<element java-property="applicant.monthlyDebtRepayment">
    <label>Totale månedskostnader alle lån</label>
    <description></description>
    <name>applicant-monthly-debt-repayment</name>
    <is-mandatory>true</is-mandatory>
    <type>integer</type>
    <min-value>0</min-value>
    <max-value>9999999</max-value>
    <unit>nok</unit>
    <format-message>Oppgi gyldig verdi (0 - 9999999 kr).</format-message>
</element>

<element java-property="applicant.creditRemarks">
    <name>applicant-credit-remarks</name>
    <label>Har du gjeld hos inkassoselskap?</label>
    <type>radio</type>
    <is-mandatory>false</is-mandatory>
    <items>
        <item>
            <label>Ja</label>
            <value>true</value>
        </item>
        <item>
            <label>Nei</label>
            <value>false</value>
        </item>
    </items>
    <format>^(?:true|false)$</format>
</element>

<element java-property="payoutAccountNo">
    <label></label>
    <name>payout-account-number</name>
    <is-mandatory>false</is-mandatory>
    <type>string</type>
    <length>11</length>
    <format>^[\d]{11}$</format>
    <format-message>Oppgi gyldig verdi - 11 siffer uten tegn.</format-message>
</element>

<element java-property="isFinancingOtherLoan">
    <label>Vil du refinansiere/samle gjeld?</label>
    <description></description>
    <name>finance-other-loans</name>
    <type>checkbox</type>
    <is-mandatory>true</is-mandatory>
    <items>

```

```

<item>
    <label>Ja</label>
    <value>true</value>
</item>
</items>
<format>^(true|)$</format>
</element>

<element java-property="balanceOfConsolidatedLoans">
    <label>Hvor mye vil du refinansiere?</label>
    <description></description>
    <name>finance-other-loans-balance-of-loans</name>
    <is-mandatory>false</is-mandatory>
    <type>integer</type>
    <min-value>0</min-value>
    <max-value>999999</max-value>
    <unit>€</unit>
    <format-message>Oppgi gyldig verdi (0 - 999999 kr).</format-message>
</element>

<element>
    <type>group</type>
    <label>Betalingsforsikring</label>
    <description>Forsikringen skal gi deg og din familie trygghet om du skulle bli utsatt for langtidssykdom, ufrivillig arbeidsledighet/permittering, sykehuisinngang eller dødsfall på grunn av ulykke. Premien er 0,67% av din aktuelle lånesaldo. Forsikringen dekker din månedskostnad på lånet inntil 12 måneder. Ved dødsfall som følge av en ulykke betaler forsikringen gjelden på tidspunktet for dødsfallet (opptil 50 000 kr).</description>
</element>

<element java-property="isInsured">
    <label>Ja takk! Jeg vil tegne betalingsforsikring. Send meg mer informasjon og søknad.</label>
    <description></description>
    <name>insurance</name>
    <type>confirmation</type>
    <is-mandatory>false</is-mandatory>
    <items>
        <item>
            <label></label>
            <value>true</value>
        </item>
    </items>
    <format>^(true|)$</format>
</element>

<element java-property="campaignCode">
    <label>Kampanjekode</label>
    <description></description>
    <name>campaign-code</name>
    <is-mandatory>false</is-mandatory>
    <type>string</type>
    <length>10</length>
    <format>^([A-ZÅÖÜ\\-\\.\\\$\\d a-zAÖÜ]{2,5})?$_</format>
    <format-message></format-message>
</element>

<element java-property="termsAgreedTo">
    <label>
        Ved å inngå avtale med Resurs Bank inngår du i Resurskonsernets kunderegister. Hovedformålet med vår behandling av personopplysninger er å samle inn, kontrollere og registrere de opplysningsene som kreves før avtaleinngåelse samt dokumentere, administrere og gjennomføre avtalen. Personopplysninger kan også benyttes for markedsføringsformål. Når behandling skjer for markedsføringsformål kan også profilering forekomme for å rette tilpassede tilbud til deg hvorved du kan tilbys alternativt produkt eller kontaktes av en av Resurs Banks samarbeidspartnere. Fullstendig informasjon om vår behandling av personopplysninger finner du i våre vilkår for behandling av personopplysninger.
    <value>
        Jeg er innforstått med at jeg gjennom å kontakte Resurs Bank kan reservere meg mot direkte markedsføring. Undertegnede låntaker(e) erkjenner med dette å skynde banken det lånebeløp som er angitt ovenfor. Beløpet med tillegg av renter og omkostninger, beregnet på grunnlag av de satser som til enhver tid
    </value>
</label>

```

gjelder i låneperioden, tilbakebetales i samsvar med betingelsene ovenfor og de alminnelige vilkår. Jeg forsikrer at oppgitte opplysninger er riktige og fullstendige og tillater at søknaden kredittprøves og at arbeidsgiver kan komme til å kontaktes. Jeg forsikrer videre at jeg før jeg søkte har mottatt og lest Standardiserte europeiske opplysninger om forbrukerkreditt (SEF-skjema) og Resurs Banks alminnelige vilkår for privat konto- og kortkreditt samt forbrukslån for Norge, hvilke bestemmelser herved vedtas. Avtalen, som består av dette dokumentet og ovenstående dokumenter, er tilgjengelig på «Mine Sider» på www.resursbank.no. Inndrivelse kan skje uten søksmål, jf. tvangsfyllbyrdesloven § 7-2 bokstav a eller g, som også omfatter renter og utenrettslige inndrivingskostnader.

```
</label>
<name>approve-conditions</name>
<is-mandatory>true</is-mandatory>
<type>confirmation</type>
<items>
    <item>
        <label></label>
        <value>true</value>
    </item>
</items>
<format>^(true)$</format>
<format-message>Dette feltet er obligatorisk.</format-message>
</element>

<element java-property="applicant.consentEmailMarketing">
    <name>applicant-consent-email-marketing</name>
    <label>Jeg ønsker fremtidige tilbud fra Resurs Bank på e-post.</label>
    <type>checkbox</type>
    <is-mandatory>false</is-mandatory>
    <items>
        <item>
            <label></label>
            <value>true</value>
        </item>
    </items>
    <format>^(true| )$</format>
</element>

<element>
    <type>group</type>
    <label>Välj signeringssmetod</label>
</element>

<element java-property="agreementSignType">
    <label>Signeringssmetode</label>
    <description></description>
    <name>agreement-sign-type</name>
    <type>radio</type>
    <is-mandatory>true</is-mandatory>
    <items>
        <item>
            <label>Elektronisk signering</label>
            <value>E_SIGN</value>
        </item>
        <item>
            <label>Post</label>
            <value>MAIL</value>
        </item>
    </items>
    <format>^(MAIL|E_SIGN)$</format>
    <format-message>Vennligst velg signeringssmetode.</format-message>
</element>

<!-- ***** Avslutning -->
<element>
    <label>Send søknad</label>
    <imagesrc/>
    <name>send-application</name>
    <type>submit</type>
</element>
</resurs-form>
```

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns2:submitApplicationExtResponse xmlns:ns2="http://consumerloan.resurs.com/v1/msg/application" xmlns:ns3="http://consumerloan.resurs.com/v1/msg/exception">
      <submitApplicationExtResult>
        <applicationReference>lacca31b-c4c8-466a-9f7e-51c8c06591dd</applicationReference>
        <decision>APPROVED</decision>
        <approvedAmount>100000</approvedAmount>
        <interest>22.30</interest>
        <tenor>72</tenor>
        <effectiveInterest>25.31</effectiveInterest>
        <monthlyCost>2598.00</monthlyCost>
        <consolidationDemand>10000.0</consolidationDemand>
        <adminFee>19.00</adminFee>
        <arrangementFee>399.00</arrangementFee>
        <documentTypes>ID</documentTypes>
        <documentTypes>PAYMENT_SLIP</documentTypes>
        <totalRepaymentAmount>187007.00</totalRepaymentAmount>
        <monthlyAccountFee>0</monthlyAccountFee>
        <totalFeesAndInterest>87007.00</totalFeesAndInterest>
      </submitApplicationExtResult>
    </ns2:submitApplicationExtResponse>
  </soap:Body>
</soap:Envelope>
```

submitApplicationExt SE

SE

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:app="http://consumerloan.resurs.com/v1/msg/application">
  <soapenv:Header/>
  <soapenv:Body>
    <app:submitApplicationExt>
      <xml><!CDATA[<resurs-response>
<amount>10000</amount>
<time>120</time>
<applicant-government-id>ENTER GOVID</applicant-government-id>
<applicant-mobile-number>0708123456</applicant-mobile-number>
<applicant-email-address>test@resurs.se</applicant-email-address>
<purpose-of-loan>HOUSING_MORTGAGE_LOAN</purpose-of-loan>
<applicant-marital-status>MARRIED</applicant-marital-status>
<number-children-not-of-age>1</number-children-not-of-age>
<applicant-habitation-form>DETACHED_HOUSE</applicant-habitation-form>
<applicant-employment-type>FAST_ANSTALLD</applicant-employment-type>
<applicant-employer>Resurs Bank</applicant-employer>
<applicant-employment-year-from>1011</applicant-employment-year-from>
<applicant-employment-year-until>2002</applicant-employment-year-until>
<applicant-monthly-income-gross>32000</applicant-monthly-income-gross>
<household-living-expenses>8950</household-living-expenses>
<household-other-loan-expenses>1200</household-other-loan-expenses>
<approve-conditions>true</approve-conditions>
<agreement-sign-type>MAIL</agreement-sign-type>
</resurs-response>]]></xml>
      <consumerIp>1.2.3.4</consumerIp>
      <!--Optional:-->
      <externalReference>?</externalReference>
    </app:submitApplicationExt>
  </soapenv:Body>
</soapenv:Envelope>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<resurs-form>

  <!-- ***** Lånebelopp -->
  <element>
    <label>Lånebelopp</label>
    <name>subtitle-terms</name>
    <type>subtitle</type>
  </element>

  <element java-property="amount">
    <label>Lånebelopp</label>
    <name>amount</name>
    <type>integer</type>
    <is-mandatory>true</is-mandatory>
    <unit>kr</unit>
    <min-value>10000</min-value>
    <max-value>500000</max-value>
  </element>

  <element java-property="paymentTerms">
    <label>Önskad avbetalningstid</label>
    <description>Välj den önskade avbetalningstiden i listan.</description>
    <name>time</name>
    <type>list</type>
    <is-mandatory>true</is-mandatory>
    <default>144</default>
    <unit>år</unit>
    <items>
      <item>
        <label>1</label>
        <value>12</value>
      </item>
    </items>
  </element>
```

```

</item>
<item>
    <label>2</label>
    <value>24</value>
</item>
<item>
    <label>3</label>
    <value>36</value>
</item>
<item>
    <label>4</label>
    <value>48</value>
</item>
<item>
    <label>5</label>
    <value>60</value>
</item>
<item>
    <label>6</label>
    <value>72</value>
</item>
<item>
    <label>7</label>
    <value>84</value>
</item>
<item>
    <label>8</label>
    <value>96</value>
</item>
<item>
    <label>9</label>
    <value>108</value>
</item>
<item>
    <label>10</label>
    <value>120</value>
</item>
<item>
    <label>11</label>
    <value>132</value>
</item>
<item>
    <label>12</label>
    <value>144</value>
</item>
<item>
    <label>13</label>
    <value>156</value>
</item>
<item>
    <label>14</label>
    <value>168</value>
</item>
<item>
    <label>15</label>
    <value>180</value>
</item>
</items>
<format>^(12|24|36|48|60|72|84|96|108|120|132|144|156|168|180)$</format>
</element>

<!-- ***** Huvudsökande - Personalia --&gt;
&lt;element&gt;
    &lt;label&gt;Huvudsökande&lt;/label&gt;
    &lt;name&gt;subtitle-applicant&lt;/name&gt;
    &lt;type&gt;subtitle&lt;/type&gt;
&lt;/element&gt;

&lt;element java-property="applicant.governmentId"&gt;
    &lt;label&gt;Personnummer&lt;/label&gt;
    &lt;description&gt;Vänligen ange i format ÅÅMMDD-XXXX&lt;/description&gt;
</pre>

```

```

<name>applicant-government-id</name>
<is-mandatory>true</is-mandatory>
<type>string</type>
<length>11</length>
<format>^([0-9]{2})(0[1-9]|1[0-2])([0][1-9]|[1-2][0-9]|3[0-1])(\\-|\\+)?([\\d]{4})$</format>
<format-message>Ogiltigt personnummer. Vänligen ange ditt personnummer med 10 siffror, ÅÅMMDD-XXXX.</format-message>
</element>

<element java-property="applicant.mobilePhoneNumber">
    <label>Mobilnummer</label>
    <description>Vänligen skriv in endast siffror</description>
    <name>applicant-mobile-number</name>
    <is-mandatory>true</is-mandatory>
    <type>string</type>
    <length>16</length>
    <format>^((0|\\+46|0046)[ |-]?(200|20|70|73|76|74|[1-9][0-9]{0,2})([ |-]?[0-9])\{5,8}\)?$</format>
    <format-message>Vänligen kontrollera ditt mobilnummer, endast siffror ska skrivas in.</format-message>
</element>

<element java-property="applicant.emailAddress">
    <label>E-postadress</label>
    <description>Vänligen ange din e-postadress</description>
    <name>applicant-email-address</name>
    <is-mandatory>true</is-mandatory>
    <type>string</type>
    <length>50</length>
    <format>^([A-Za-z0-9!#%&'*/+=?^_`~-]+(\\\.[A-Za-z0-9!#%&'*/+=?^_`~-]+)*@[A-Za-z0-9]+)([\\.\\\-]?[a-zA-Z0-9]+)\\\.( [A-Za-z]{2,}))?\$</format>
    <format-message>Vänligen kontrollera din e-postadress</format-message>
</element>

<element java-property="loanPurpose">
    <label>Syfte med lån?</label>
    <name>purpose-of-loan</name>
    <type>list</type>
    <is-mandatory>true</is-mandatory>
    <items>
        <item>
            <label>Bil/MC</label>
            <value>MOTOR</value>
        </item>
        <item>
            <label>Bröllop</label>
            <value>WEDDING</value>
        </item>
        <item>
            <label>Båt</label>
            <value>BOAT</value>
        </item>
        <item>
            <label>Investering (t.ex. aktier)</label>
            <value>INVESTMENT</value>
        </item>
        <item>
            <label>Körkort</label>
            <value>DRIVERS_LICENSE</value>
        </item>
        <item>
            <label>Inredning (t.ex. möbler, hemelektronik)</label>
            <value>INTERIOR</value>
        </item>
        <item>
            <label>Renovering</label>
            <value>RENOVATION</value>
        </item>
        <item>
            <label>Resa/Fritid</label>
            <value>TRAVEL</value>
        </item>
        <item>

```

```

        <label>Samla lån/kredit</label>
        <value>CONSOLIDATE_LOANS</value>
    </item>
    <item>
        <label>Sjuk- eller tandvård</label>
        <value>DENTAL_OR_OTHER_CARE</value>
    </item>
    <item>
        <label>Studier</label>
        <value>STUDIES</value>
    </item>
    <item>
        <label>Övrig konsumtion</label>
        <value>OTHER</value>
    </item>
    <item>
        <label>Köp av bostad</label>
        <value>HOUSING_MORTGAGE_LOAN</value>
    </item>
</items>
<format>
(MOTOR|WEDDING|BOAT|INVESTMENT|DRIVERS_LICENSE|INTERIOR|RENOVATION|TRAVEL|CONSOLIDATE_LOANS|DENTAL_OR_OTHER_CARE
|STUDIES|OTHER|HOUSING_MORTGAGE_LOAN)$</format>
<format-message>Vänligen välj ett alternativ</format-message>
</element>

<element java-property="applicant.maritalStatus">
    <label>Civilstånd</label>
    <name>applicant-marital-status</name>
    <type>list</type>
    <is-mandatory>true</is-mandatory>
    <items>
        <item>
            <label>Gift/Registrerad partner</label>
            <value>MARRIED</value>
        </item>
        <item>
            <label>Sambo</label>
            <value>DOMESTIC_PARTNER</value>
        </item>
        <item>
            <label>Singel</label>
            <value>SINGLE</value>
        </item>
    </items>
<format>^(SINGLE|MARRIED|DOMESTIC_PARTNER)$</format>
<format-message>Vänligen välj ett alternativ</format-message>
</element>

<element java-property="applicant.numberChildrenNotOfAge">
    <label>Antal barn under 18 år</label>
    <description>Om du inte har barn, vänligen ange 0.</description>
    <name>number-children-not-of-age</name>
    <is-mandatory>true</is-mandatory>
    <type>list</type>
    <items>
        <item>
            <label>0</label>
            <value>0</value>
        </item>
        <item>
            <label>1</label>
            <value>1</value>
        </item>
        <item>
            <label>2</label>
            <value>2</value>
        </item>
        <item>
            <label>3</label>
            <value>3</value>
        </item>
    </items>
</element>
```

```

        </item>
        <item>
            <label>4</label>
            <value>4</value>
        </item>
        <item>
            <label>5</label>
            <value>5</value>
        </item>
        <item>
            <label>6</label>
            <value>6</value>
        </item>
        <item>
            <label>7</label>
            <value>7</value>
        </item>
        <item>
            <label>8</label>
            <value>8</value>
        </item>
        <item>
            <label>9</label>
            <value>9</value>
        </item>
        <item>
            <label>10 eller fler</label>
            <value>10</value>
        </item>
    </items>
<format>^(0|1|2|3|4|5|6|7|8|9|10)$</format>
<format-message>Vänligen ange ett giltigt antal barn under 18 (mellan 0 och 10 eller fler)</format-
message>
</element>

<element java-property="applicant.habitationType">
    <label>Boendeform</label>
    <name>applicant-habitation-form</name>
    <type>list</type>
    <is-mandatory>true</is-mandatory>
    <items>
        <item>
            <label>Hyresrätt</label>
            <value>APARTMENT</value>
        </item>
        <item>
            <label>Bostadsrätt</label>
            <value>CONDOMINIUM</value>
        </item>
        <item>
            <label>Egen fastighet</label>
            <value>DETACHED_HOUSE</value>
        </item>
    </items>
<format>^(DETACHED_HOUSE|APARTMENT|CONDOMINIUM)$</format>
<format-message>Vänligen välj ett alternativ</format-message>
</element>

<!-- ***** Huvudsökande - Arbete --&gt;
&lt;element java-property="applicant.employmentType"&gt;
    &lt;label&gt;Anställningsform&lt;/label&gt;
    &lt;name&gt;applicant-employment-type&lt;/name&gt;
    &lt;type&gt;list&lt;/type&gt;
    &lt;is-mandatory&gt;true&lt;/is-mandatory&gt;
    &lt;items&gt;
        &lt;item&gt;
            &lt;label&gt;Fast anställd&lt;/label&gt;
            &lt;value&gt;FAST_ANSTALLD&lt;/value&gt;
        &lt;/item&gt;
        &lt;item&gt;
            &lt;label&gt;Visstidsansställd&lt;/label&gt;
</pre>

```

```

        <value>VIKARIAT</value>
    </item>
    <item>
        <label>Eget företag</label>
        <value>EGEN_FORETAGARE</value>
    </item>
    <item>
        <label>Pensionär</label>
        <value>PENSIONAR</value>
    </item>
    <item>
        <label>Saknar arbete</label>
        <value>ARBETSLOS</value>
    </item>
</items>
<format>^(FAST_ANSTALLD|VIKARIAT|EGEN_FORETAGARE|PENSIONAR|ARBETSLOS)$</format>
<format-message>Vänligen välj ett alternativ</format-message>
</element>

<element java-property="applicant.employmentTypeQualifying">
    <label>Är din visstidsanställning inom utbildning eller vård & omsorg?</label>
    <name>applicant-employment-type-qualifying</name>
    <type>radio</type>
    <is-mandatory>false</is-mandatory>
    <items>
        <item>
            <label>Ja</label>
            <value>true</value>
        </item>
        <item>
            <label>Nej</label>
            <value>false</value>
        </item>
    </items>
    <format>^(true|false)$</format>
</element>

<element java-property="applicant.employerName">
    <label>Arbetsgivare</label>
    <description>Vänligen ange företagsnamnet</description>
    <name>applicant-employer</name>
    <is-mandatory>false</is-mandatory>
    <type>string</type>
    <length>100</length>
    <format>^[\s\S]{0,100}$</format>
    <format-message>Vänligen skriv in din arbetsgivare</format-message>
</element>

<element java-property="applicant.employedSince">
    <label>Anställd/Eget företag/Pensionär sedan</label>
    <description>Vänligen ange i format ÅÅMM. Välj år och månad för nuvarande anställning, när du startade eget eller blev pensionär.</description>
    <name>applicant-employment-year-from</name>
    <is-mandatory>false</is-mandatory>
    <type>string</type>
    <length>4</length>
    <format>^([0-9]{2})(0[1-9]|1[012])$</format>
    <format-message>Vänligen välj ett alternativ</format-message>
</element>

<element java-property="applicant.employedTo">
    <label>Anställd t o m</label>
    <description>Vänligen välj år och månad då du slutar din visstidsanställning</description>
    <name>applicant-employment-year-until</name>
    <is-mandatory>false</is-mandatory>
    <type>string</type>
    <length>4</length>
    <format>^([0-9]{2})(0[1-9]|1[012])$</format>
    <format-message>Vänligen välj ett alternativ</format-message>
</element>

```

```

<element java-property="applicant.grossMonthlyIncome">
    <label>Inkomst per månad före skatt</label>
    <description>Vänligen skriv in endast siffror</description>
    <name>applicant-monthly-income-gross</name>
    <is-mandatory>true</is-mandatory>
    <type>integer</type>
    <min-value>0</min-value>
    <max-value>999999</max-value>
    <unit>kr</unit>
    <format-message>Vänligen ange din inkomst per månad före skatt i siffror (0 - 999999)</format-message>
</element>

<element java-property="householdLivingExpenses">
    <label>Boendekostnad per månad</label>
    <description>Vänligen ange din del av boendekostnad (hyra, bolåneränta, amortering, avgift, el, värme) per månad i siffror.</description>
    <name>household-living-expenses</name>
    <is-mandatory>true</is-mandatory>
    <type>integer</type>
    <min-value>0</min-value>
    <max-value>99999</max-value>
    <unit>kr</unit>
    <format-message>Vänligen ange ett värde i siffror (0-99999)</format-message>
</element>

<element java-property="householdLoanExpenses">
    <label>Kostnader övriga lån/krediter per månad exkl. bolån</label>
    <description>Vänligen ange din del av månadskostnad för övriga lån/krediter, exkl. bolån.</description>
    <name>household-other-loan-expenses</name>
    <is-mandatory>true</is-mandatory>
    <type>integer</type>
    <min-value>0</min-value>
    <max-value>99999</max-value>
    <unit>kr</unit>
    <format-message>Vänligen ange ett värde i siffror (0-99999)</format-message>
</element>

<element java-property="financeOtherLoansAmount">
    <label>Ange vilket belopp du vill samla</label>
    <name>finance-other-loans-balance-of-loans</name>
    <is-mandatory>false</is-mandatory>
    <type>integer</type>
    <min-value>0</min-value>
    <max-value>500000</max-value>
    <unit>money</unit>
    <format-message>Vänligen ange ett värde i siffror (0-500000)</format-message>
</element>

<element java-property="payoutClearingNo">
    <label>Clearingnummer</label>
    <description>Ett clearingnummer består av 4 eller 5 siffror. För att kunna e-signera ditt avtal måste du uppgive clearingnummer.</description>
    <name>payout-clearing-number</name>
    <is-mandatory>false</is-mandatory>
    <type>string</type>
    <length>5</length>
    <format>^\d{4,5}\$</format>
    <format-message>Vänligen ange ett giltigt clearing nummer (4 eller 5 siffror)</format-message>
</element>

<element java-property="payoutAccountNo">
    <label>Kontonummer</label>
    <description>Vänligen ange ditt kontonummer utan bindestreck eller punkter. För att kunna e-signera ditt avtal måste du uppgive kontonummer.</description>
    <name>payout-account-number</name>
    <is-mandatory>false</is-mandatory>
    <type>string</type>
    <length>12</length>
    <format>^\d{6,12}\$</format>
    <format-message>Vänligen fyll i ett giltigt kontonummer (utan bindestreck eller punkter)</format-message>
</element>

```

```

</element>

<element java-property="payoutClearingNo">
    <label>Clearingnummer</label>
    <description>Partners clearingnummer för hybridlån. Ett clearingnummer består av 4 eller 5 siffror. För att kunna e-signera ditt avtal måste du uppge clearingnummer.</description>
    <name>partner-clearing-number</name>
    <is-mandatory>false</is-mandatory>
    <type>string</type>
    <length>5</length>
    <format>^[\d]{4,5}$</format>
    <format-message>Vänligen ange ett giltigt clearing nummer (4 eller 5 siffror)</format-message>
</element>

<element java-property="payoutAccountNo">
    <label>Kontonummer</label>
    <description>Partners kontonummer för hybridlån. Vänligen ange ditt kontonummer utan bindestreck eller punkter. För att kunna e-signera ditt avtal måste du uppge kontonummer.</description>
    <name>partner-disbursement-account-number</name>
    <is-mandatory>false</is-mandatory>
    <type>string</type>
    <length>12</length>
    <format>^[\d]{6,12}$</format>
    <format-message>Vänligen fyll i ett giltigt kontonummer (utan bindestreck eller punkter)</format-message>
</element>

<!-- ***** Medsökande - Personalia --&gt;
&lt;element&gt;
    &lt;label&gt;Eventuell medsökande&lt;/label&gt;
    &lt;name&gt;subtitle-coapplicant&lt;/name&gt;
    &lt;type&gt;subtitle&lt;/type&gt;
&lt;/element&gt;

&lt;element java-property="coApplicant.governmentId"&gt;
    &lt;label&gt;Personnummer&lt;/label&gt;
    &lt;description&gt;Vänligen ange i format ÅÅMMDD-XXXX&lt;/description&gt;
    &lt;name&gt;coapplicant-government-id&lt;/name&gt;
    &lt;is-mandatory&gt;false&lt;/is-mandatory&gt;
    &lt;type&gt;string&lt;/type&gt;
    &lt;length&gt;11&lt;/length&gt;
    &lt;format&gt;^([0-9]{2})(0[1-9]|1[0-2])([0][1-9]|1-2)[0-9]|3[0-1])(\|-|\+)?([\d]{4})$&lt;/format&gt;
    &lt;format-message&gt;Ogiltigt personnummer. Vänligen ange ditt personnummer med 10 siffror, ÅÅMMDD-XXXX.&lt;/format-message&gt;
&lt;/element&gt;

&lt;element java-property="coApplicant.mobilePhoneNumber"&gt;
    &lt;label&gt;Mobilnummer&lt;/label&gt;
    &lt;description&gt;Vänligen skriv in endast siffror&lt;/description&gt;
    &lt;name&gt;coapplicant-mobile-number&lt;/name&gt;
    &lt;is-mandatory&gt;false&lt;/is-mandatory&gt;
    &lt;type&gt;string&lt;/type&gt;
    &lt;length&gt;16&lt;/length&gt;
    &lt;format&gt;^((0|\+46|0046)[-]?(20|20|70|73|76|74|[1-9][0-9]{0,2})([-]?[0-9])\{5,8\})?&lt;/format&gt;
    &lt;format-message&gt;Vänligen kontrollera ditt mobilnummer, endast siffror ska skrivas in.&lt;/format-message&gt;
    &lt;depends-on&gt;coapplicant-government-id&lt;/depends-on&gt;
&lt;/element&gt;

&lt;element java-property="coApplicant emailAddress"&gt;
    &lt;label&gt;E-postadress&lt;/label&gt;
    &lt;description&gt;Vänligen ange din e-postadress&lt;/description&gt;
    &lt;name&gt;coapplicant-email-address&lt;/name&gt;
    &lt;is-mandatory&gt;false&lt;/is-mandatory&gt;
    &lt;type&gt;string&lt;/type&gt;
    &lt;length&gt;50&lt;/length&gt;
    &lt;format&gt;^([A-Za-z0-9!#%&amp;'*/=?^_`~-]+(\.\[A-Za-z0-9!#%&amp;'*/=?^_`~-]+)*@[A-Za-z0-9]+)([\\.\-]?[a-zA-Z0-9]+)*\\.( [A-Za-z]{2,}))?&lt;/format&gt;
    &lt;format-message&gt;Vänligen ange din e-postadress&lt;/format-message&gt;
    &lt;depends-on&gt;coapplicant-government-id&lt;/depends-on&gt;
&lt;/element&gt;</pre>

```

```

<!-- ***** Medsökande - Civilstånd -->
<element java-property="coApplicant.maritalStatus">
    <label>Civilstånd</label>
    <name>coapplicant-marital-status</name>
    <is-mandatory>false</is-mandatory>
    <type>list</type>
    <is-mandatory>false</is-mandatory>
    <items>
        <item>
            <label>Gift registrerad partner</label>
            <value>MARRIED</value>
        </item>
        <item>
            <label>Sambo</label>
            <value>DOMESTIC_PARTNER</value>
        </item>
        <item>
            <label>Singel</label>
            <value>SINGLE</value>
        </item>
    </items>
    <format>^(MARRIED|DOMESTIC_PARTNER|SINGLE)?$</format>
    <depends-on>coapplicant-government-id</depends-on>
</element>

<!-- ***** Medsökande - Bostad -->
<element java-property="coApplicant.habitationType">
    <label>Boendeform</label>
    <name>coapplicant-habitation-form</name>
    <is-mandatory>false</is-mandatory>
    <type>list</type>
    <items>
        <item>
            <label>Hyresrätt</label>
            <value>APARTMENT</value>
        </item>
        <item>
            <label>Bostadsrätt</label>
            <value>CONDOMINIUM</value>
        </item>
        <item>
            <label>Egen fastighet</label>
            <value>DETACHED_HOUSE</value>
        </item>
    </items>
    <format>^(DETACHED_HOUSE|APARTMENT|CONDOMINIUM)$</format>
    <format-message>Vänligen välj ett alternativ</format-message>
    <depends-on>coapplicant-government-id</depends-on>
</element>

<!-- ***** Medsökande - Arbete -->
<element java-property="coApplicant.employmentType">
    <label>Anställningsform</label>
    <name>coapplicant-employment-type</name>
    <is-mandatory>false</is-mandatory>
    <type>list</type>
    <items>
        <item>
            <label>Fast anställd</label>
            <value>FAST_ANSTALLD</value>
        </item>
        <item>
            <label>Visstidsansställd</label>
            <value>VIKARIAT</value>
        </item>
        <item>
            <label>Eget företag</label>
            <value>EGEN_FORETAGARE</value>
        </item>
        <item>
            <label>Pensionär</label>

```

```

        <value>PENSIONAR</value>
    </item>
    <item>
        <label>Saknar arbete</label>
        <value>ARBETSLOS</value>
    </item>
</items>
<format>^(FAST_ANSTALLD|VIKARIAT|EGEN_FORETAGARE|PENSIONAR|ARBETSLOS)$</format>
<format-message>Vänligen välj ett alternativ</format-message>
<depends-on>coapplicant-government-id</depends-on>
</element>

<element java-property="coApplicant.employmentTypeQualifying">
    <label>Är din visstidsanställning inom utbildning eller vård & omsorg?</label>
    <name>coapplicant-employment-type-qualifying</name>
    <type>radio</type>
    <is-mandatory>false</is-mandatory>
    <items>
        <item>
            <label>Ja</label>
            <value>true</value>
        </item>
        <item>
            <label>Nej</label>
            <value>false</value>
        </item>
    </items>
    <format>^(true|false)$</format>
</element>

<element java-property="coApplicant.employerName">
    <label>Arbetsgivare</label>
    <description>Vänligen ange företagsnamnet</description>
    <name>coapplicant-employer</name>
    <is-mandatory>false</is-mandatory>
    <type>string</type>
    <length>100</length>
    <format>^[\s\S]{0,100}$</format>
    <format-message>Vänligen skriv in din arbetsgivare</format-message>
    <depends-on>coapplicant-government-id</depends-on>
</element>

<element java-property="coApplicant.employedSince">
    <label>Anställd/Eget företag/Pensionär sedan</label>
    <description>Vänligen ange i format ÅÅMM. Välj år och månad för nuvarande anställning, när du startade eget eller blev pensionär.</description>
    <name>coapplicant-employment-year-from</name>
    <is-mandatory>false</is-mandatory>
    <type>string</type>
    <length>4</length>
    <format>^(([0-9]{2})([0-9]{2}))?</format>
    <format-message>Vänligen välj ett alternativ</format-message>
    <depends-on>coapplicant-government-id</depends-on>
</element>

<element java-property="coApplicant.employedTo">
    <label>Anställd t o m</label>
    <description>Vänligen välj år och månad då du slutar din visstidsanställning</description>
    <name>coapplicant-employment-year-until</name>
    <is-mandatory>false</is-mandatory>
    <type>string</type>
    <length>4</length>
    <format>^(([0-9]{2})([0-9]{2}))?</format>
    <format-message>Vänligen välj ett alternativ</format-message>
    <depends-on>coapplicant-government-id</depends-on>
</element>

<element java-property="coApplicant.grossMonthlyIncome">
    <label>Inkomst per månad före skatt</label>
    <description>Vänligen skriv in endast siffror</description>
    <name>coapplicant-monthly-income-gross</name>

```

```

<is-mandatory>false</is-mandatory>
<type>integer</type>
<min-value>0</min-value>
<max-value>9999999</max-value>
<unit>kr</unit>
<format-message>Vänligen ange din inkomst per månad före skatt i siffror (0 - 999999)</format-message>
<depends-on>coapplicant-government-id</depends-on>
</element>

<element java-property="isInsured">
    <label>Låneskydd</label>
    <name>insurance</name>
    <type>checkbox</type>
    <is-mandatory>false</is-mandatory>
    <items>
        <item>
            <label>Ja</label>
            <value>true</value>
        </item>
    </items>
    <format>^(true| )$</format>
</element>

<!-- ***** Godkännande -->
<element>
    <label>Villkor</label>
    <name>subtitle-conditions</name>
    <type>subtitle</type>
</element>

<element java-property="termsAgreedTo">
    <format-message>Du måste godkänna villkoren för att kunna gå vidare.</format-message>
    <label>Villkorsbekräfelse</label>
    <name>approve-conditions</name>
    <is-mandatory>true</is-mandatory>
    <type>confirmation</type>
    <items>
        <item>
            <label>
                Jag accepterar Resurs Banks allmänna villkor.  

                Vid tecknande av avtal med Resurs Bank ingår du i Resurskoncernens kundregister.  

                Inhämtade personuppgifter kan användas och distribueras för marknadsföring.  

                Jag är införstådd med att jag genom att kontakta Resurs Bank kan förbjuda behandling  

                av personuppgifter för marknadsföring. Jag försäkrar att de uppgifter som lämnats är  

                riktiga, fullständiga och att jag uppfyller de allmänna förutsättningarna för  

                Privatlån. Jag tillåter kreditgivaren att göra sedvanlig kreditprövning varvid  

                arbetsgivaren kan komma att kontaktas. Kreditgivaren förbehåller sig fri  

                prövningsrätt. Jag godkänner även att jag vid avslag kan komma att erbjudas  

                alternativ produkt eller kontaktas av en av Resurs Banks samarbetspartners. Jag  

                försäkrar vidare att jag skriftligen eller på www.resursbank.se tagit del av samt  

                mottagit kopia av vid tidpunkten för detta avtals ingående gällande Standardiserad  

                europeisk konsumentkreditinformation (SEKKI) med aktuella Specialkontovillkor samt  

                Resurs Banks allmänna villkor för privat konto- och kortkredit samt privatlån för  

                Sverige, vars bestämmelser härmed accepteras.
            </label>
            <value>true</value>
        </item>
    </items>
    <format>^(true)$</format>
</element>

<element java-property="agreementSignType">
    <label>Signeringsmetod</label>
    <name>agreement-sign-type</name>
    <type>radio</type>
    <is-mandatory>true</is-mandatory>
    <items>
        <item>
            <label>E-signering</label>
            <value>E_SIGN</value>
        </item>
    </items>
</element>
```

```

<item>
    <label>Postalt avtal</label>
    <value>MAIL</value>
</item>
</items>
<format>^(MAIL|E_SIGN)$</format>
<format-message>Vänligen välj signeringsmetod</format-message>
</element>

<element>
    <label>Skicka</label>
    <imagesrc></imagesrc>
    <name>send-application</name>
    <type>submit</type>
</element>
</resurs-form>

```

```

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body>
        <ns2:submitApplicationExtResponse xmlns:ns2="http://consumerloan.resurs.com/v1/msg/application" xmlns:ns3="http://consumerloan.resurs.com/v1/msg/exception">
            <submitApplicationExtResult>
                <applicationReference>eabea8be-abda-432b-97ec-8199232d2613</applicationReference>
                <decision>APPROVED</decision>
                <approvedAmount>10000</approvedAmount>
                <interest>8.02</interest>
                <tenor>72</tenor>
                <effectiveInterest>14.05</effectiveInterest>
                <monthlyCost>202.00</monthlyCost>
                <consolidationDemand>1000.0</consolidationDemand>
                <adminFee>19.00</adminFee>
                <arrangementFee>399.00</arrangementFee>
                <documentTypes>ID</documentTypes>
                <documentTypes>PAYMENT_SLIP</documentTypes>
                <monthlyAccountFee>0</monthlyAccountFee>
                <totalFeesAndInterest>1000.00</totalFeesAndInterest>
            </submitApplicationExtResult>
        </ns2:submitApplicationExtResponse>
    </soap:Body>
</soap:Envelope>

```

PUT /api/callback/delivery-complete

```
{
    "publicApplicationReferenceId": "e99ea389-584f-49a9-bbc1-f51bcb62ff5e"
}
```

getApplicationQuote

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:app="http://consumerloan.resurs.com/v1/msg/application">
  <soapenv:Header/>
  <soapenv:Body>
    <app:getApplicationQuote>
      <applicationReference>eabeba8be-abda-432b-97ec-8199232d2613</applicationReference>
    </app:getApplicationQuote>
  </soapenv:Body>
</soapenv:Envelope>
```

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns2:getApplicationQuoteResponse xmlns:ns2="http://consumerloan.resurs.com/v1/msg/application" xmlns:ns3="http://consumerloan.resurs.com/v1/msg/exception">
      <applicationQuoteResult>
        <applicationReference>eabeba8be-abda-432b-97ec-8199232d2613</applicationReference>
        <decision>APPROVED</decision>
        <approvedAmount>10000</approvedAmount>
        <interest>8.02</interest>
        <tenor>72</tenor>
        <effectiveInterest>14.05</effectiveInterest>
        <monthlyCost>202.00</monthlyCost>
        <consolidationDemand>1000.0</consolidationDemand>
        <adminFee>19.00</adminFee>
        <arrangementFee>399.00</arrangementFee>
        <documentTypes>ID</documentTypes>
        <documentTypes>PAYMENT_SLIP</documentTypes>
        <monthlyAccountFee>0</monthlyAccountFee>
      </applicationQuoteResult>
    </ns2:getApplicationQuoteResponse>
  </soap:Body>
</soap:Envelope>
```

acceptQuote

To accept an offer with the ACCEPT decision, the function "acceptQuote" should be used.

**If information needs to be updated, use the "[updateApplication](#)" operation.
(This needs to be done before acceptQuote)**

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:app="http://consumerloan.resurs.com/v1/msg/application">
  <soapenv:Header/>
  <soapenv:Body>
    <app:acceptQuote>
      <applicationReference>reference-obtained-when-submitting-application</applicationReference>
    </app:acceptQuote>
  </soapenv:Body>
</soapenv:Envelope>
```

```
<acceptQuoteResult>
  <accepted>true</accepted>
  <message>Offer accepted</message>
</acceptQuoteResult>
```

updateApplication

One or several fields can be sent / updated in the request.
(This request must be done prior to accepting the offer.)

Fields that can be updated are:

- applicantEmail
- coApplicantEmail
- account>clearing
 - If the clearing consists of 5 digits and where the last digit is a control number it must be included, the control number cannot be omitted.
- account>number
 - Control number must be included, it cannot be omitted.
- iban
- requestedSigningMethod



E_SIGN Clearing- & account number

If customer is to choose **E_SIGN** as signing method, it is mandatory to input clearing- and account number in either submitApplicationExt or updateApplication prior to acceptQuote.
Email is also mandatory since link for signing will be sent by email.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:app="http://consumerloan.resurs.com/v1/msg/application">
  <soapenv:Header/>
  <soapenv:Body>
    <app:updateApplication>
      <applicationReference>?</applicationReference>
      <application>
        <!--Optional:-->
        <applicantEmail>?</applicantEmail>
        <!--Optional:-->
        <coApplicantEmail>?</coApplicantEmail>
        <!--Optional:-->
        <account>
          <clearing>?</clearing>
          <number>?</number>
        </account>
        <!--Optional:-->
        <iban>?</iban>
        <!--Optional:-->
        <requestedSigningMethod>?</requestedSigningMethod>
      </application>
    </app:updateApplication>
  </soapenv:Body>
</soapenv:Envelope>
```

```
<updateApplicationResult>
  <updated>true</updated>
  <message>Applicationupdated</message>
</updateApplicationResult>
```

Callbacks and polling status

There are two ways to get information regarding the loan status - to poll the loan application reference or to register a callbackURL which will automatically send you an update when a change is made to the status.

The following events will trigger a callback; CREDIT_DECISION_UPDATED, SIGNED, WAITING_FOR_ADDITION and PAID_OUT.

Below is both methods explained with example requests & responses.

Important!

In order to test callback properly, you'll need to use your specific webservice-credentials that has been sent to you. testse, testno, testdk, testfi will not work, since integrators will overwrite eachothers callback-url.

Retrieve status change with callback

Register the callback-URL

To set a callbackURL, you send in the desired URL in xml-format. The callbackUrl is to be sent in only once! See example below. There is also an option to set credentials for the callback, this is optional. The auth methods are either BASIC and BEARER, with a base 64 encoded credentials

setCallback-example request

```
<setCallback>
    <callbackConfiguration>
        <callbackUrl>http://agent.com/url-to-callback</callbackUrl>
        <callbackCredentials>#The base64 encoded credentials#</callbackCredentials>
        <callbackAuthMethod>(BASIC|BEARER)</callbackAuthMethod>
    </callbackConfiguration>
</setCallback>
```

setCallback-example response

```
<setCallbackResponse>
    <setCallbackResult>
        <callbackSet>true</callbackSet>
        <message>Set callback successful</message>
    </setCallbackResult>
</setCallbackResponse>
```

How to trigger a callback in test environment with Postman-example

On the following Swagger-link, you can find how to setup the triggering of a callback with a GET - https://apigw.integration.resurs.com/api_docs/external_test_utils

To trigger a specific callback, the applicationReference from the submitApplicationExt-response must be provided.
If you run your test thru Postman, simply configure GET and insert the end point; no authorization nor body is required - see the example below;

i NOTE! Once one or more callbacks has been triggered, the application may not be treated as normal since there can be side effects due the "manual" triggering

Example callbacks - all sent in JSON-format

SIGNED – when the signed agreement is received

SIGNED

```
{
  "externalApplicationEvent" : {
    "applicationReference" : "ABC-123",
    "applicationEventType" : "SIGNED"
  },
  "jwsData" : "eyJJOXAiOLADikz.eyJpc3MiJgbDQrm4.djjft4CP-mbW1"
}
```

CREDIT_DECISION_UPDATED – APPROVED – when initial credit decision manual_inspection is changed to approved

CREDIT_DECISION_UPDATED – APPROVED

```
{
  "externalApplicationEvent" : {
    "applicationReference" : "abc-123-def-567",
    "applicationEventType" : "CREDIT_DECISION_UPDATED",
    "creditDecision" : {
      "creditDecision" : "APPROVED",
      "approvedAmount" : 10000,
      "interest" : 12.3,
      "tenor" : 12,
      "effectiveInterest" : 23.3,
      "monthlyCost" : 1234,
      "consolidationDemand" : 2300,
      "adminFee" : 199,
      "arrangementFee" : 599.99,
      "totalRepaymentAmount" : 13000,
      "monthlyAccountFee" : 0,
      "totalFeesAndInterest": 3000.00,
      "documentTypes" : [ "ID", "PAYMENT_SLIP" ]
    }
  },
  "jwsData" : "eyJJOXAiOLADikz.eyJpc3MiJgbDQrm4.djjft4CP-mbW1"
}
```

CREDIT_DECISION_UPDATED – REJECTED – when initial credit decision manual_inspection is changed to rejected

CREDIT_DECISION_UPDATED – REJECTED

```
{  
  "externalApplicationEvent" : {  
    "applicationReference" : "abc-123-def-567",  
    "applicationEventType" : "CREDIT_DECISION_UPDATED",  
    "creditDecision" : {  
      "creditDecision" : "REJECTED",  
      "approvedAmount" : 0  
    }  
  },  
  "jwsData" : " eyJOXAiOLADikzeyJpc3MiJgbDQrm4.djjft4CP-mbW1"  
}
```

WAITING_FOR_ADDITION – when we are awaiting additional information from the customer

WAITING_FOR_ADDITION

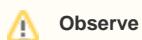
```
{  
  "externalApplicationEvent" : {  
    "requiredDocuments" : [ "ID" ],  
    "applicationReference" : "abc-123-def-567",  
    "applicationEventType" : "WAITING_FOR_ADDITION"  
  },  
  "jwsData" : " eyJOXAiOLADikzeyJpc3MiJgbDQrm4.djjft4CP-mbW1"  
}
```

PAID_OUT - When Resurs has paid out the loan. (Disclaimer! This callback cannot be generated in test environment)

PAID_OUT

```
{  
  "externalApplicationEvent" : {  
    "applicationReference" : "ABC-123",  
    "applicationEventType" : "PAID_OUT"  
  },  
  "jwsData" : " eyJOXAiOLADikzeyJpc3MiJgbDQrm4.djjft4CP-mbW1"  
}
```

jwsData



Observe

Using jwsData is optional!

All questions regarding jwsData or API-key is to be sent to support@resurs.se, not our usual email onboarding@resurs.se

The value in jwsData is a signed and serialized JWT, known as a JWS (JSON Web Signature). jwsData is a data structure cryptographically securing a JWS Header and a JWS Payload with a JWS Signature (graphically explained below).



If you want to verify the JWS, you are to download our public keys from https://apigw.resurs.com/api/auth_service/jwks. Note that you must use an API-key, which is handed out by us, in order to call the mentioned URL.

For test environment, you can download the public key from https://apigw.integration.resurs.com/api/auth_service/jwks.

Below is an example of verification

Example-request for verification

```
curl -X GET "https://apigw.integration.resurs.com/api/auth_service/jwks" -H "accept: application/json" -H "apikey: e3331b3687xxxxxxxxxxd9d4197f30b9"
```

Found below is an example layout of a public key.

Example layout of a public key

```
{
  "keys": [
    {
      "kty": "RSA",
      "e": "AQAB",
      "use": "sig",
      "kid": "1cb06a4a-3465-4f11-ac3d-d8bb2bdb1eb7",
      "alg": "RS256",
      "n": "1LtFi6baHG0yJFWKCF1fY5hMlqUoqvTWwFyjsKckuI62WVwv-
GJ1vTPnCk9cptRjqGltQ4IJxv8FYOPPjaRi981i53BnaHRmg6pITF68OA1bnTpMn3_hqctSS_6XM2s-
PFMfcfbFuSj_W4IzC2f1FmCXJSONz16zxy6kvlyC-
ZVBLbN1WpWsheufGqR7tWibj780lgA8nwTQNHHXhdlydOU3CVfGEgs9IRx8vB53n0nTHwBdMI5i5ta9wEe2f7r6I4YqganJL_q_ilYfrbAB2yOCQ
z4AiWOpQzrhZCV54tiSGfz4lVfVS_oJOy17LapdSqxnBmJwX_R4uTR0g7BLWw"
    },
    {
      "kty": "RSA",
      "e": "AQAB",
      "use": "sig",
      "kid": "e434a27b-e1fb-4f7f-89d6-fb4e95619aac",
      "alg": "RS256",
      "n": "vUUUiYxqYwCHreCGlCBn0YjCBoLa-jrqfEhSLAWafCxaGFs5mAhrzb6Zzc_cfzNptNQMtA4x2qhF68copSbGyXxq8ZQ1-
10J0cjLJ2HW98-5jJiGit3MIR5cEGEjMrRzdaltCFiGSIEussdJMOvNtJk7-0GMQgxMVMJ6PK5zARsNauj85WuB2UyOg6EDBRP7pfmQ9V4bBRGM-
s3PQZbBCD-          Mcr1K0tQTc60wftrC78Ax1VubF60jpj611bcvpsld8uFlwVlrU8gjh65CL9KTXMmb261PLrv-op8TSjFMT-
DPw4Jb54RxqjbEO9ze6b6BJo-0eU7_suqzitMT4JaQ"
    }
  ]
}
```

To poll the application reference

Simply enter the application reference, which was retrieved in the submitApplicationExt-response, to see whether an update has been made. See example request & response below.

getApplicationStatus-request

```
<getApplicationStatus>
  <applicationReference>14321-21234-445674-434344-345445</applicationReference>
</getApplicationStatus>
```

getApplicationStatusResponse-example

```
<getApplicationStatusResponse>
  <waitingForAddition>true|false</waitingForAddition>
  <signedWithMethod>E_SIGN|ANALOG</signedWithMethod>
  <requestedSigningMethod>E_SIGN|MANUAL</requestedSigningMethod>
  <paidOut>true|false</paidOut>
</getApplicationStatusResponse>
```

Reporting files example

Partners will receive reporting files from the different channels depending on payment method. Note that the csv-file below is only an example. To setup what your csv-file should contain, please discuss this matter with the designated account manager at Resurs.

For payments with Resurs Bank's methods, invoice, partpayment and national cards the file will be .csv



Legal requirements

When presenting a Resurs Bank payment method there are some legal requirements that should be adhered to:

1. The customer should be warned that a credit check might be done when using payment methods of type NEW_ACCOUNT and type INVOICE but not when using type CARD.
2. If data returned by [getCostOfPurchaseHtml](#) are used, it should be possible to reach within a click from where our payment methods are marketed.
3. If the links returned with [getPaymentMethod](#) are used, those should be shown where the corresponding payment methods are marketed.

The screenshot shows a user interface for selecting a payment method. At the top left is a small blue circular icon with a white 'R' and the word 'Resurs' stacked above 'Bank'. To its right is the text 'PAYMENT METHOD'. In the top right corner is a 'reset' button. Below this, there are three rows of options, each starting with the same blue icon. The first row contains 'Nytt kort' and a 'Läs mer >>' link. The second row contains 'Befintligt kort' and a 'Läs mer >>' link. The third row contains 'Företagsfaktura' and a 'Läs mer >>' link.

For further information and an example of legal documents and other credit cost information, please see: [Credit cost information](#)

Testing

TESTING IN PRODUCTION ENVIRONMENTS

Avoid running tests in the live environment. Read more below.

Usually representatives and integrators are tempted to run tests in the live environment.

 Please don't 

The main consequence is that credit applications are made and these are saved by credit agencies (like UC). Later, when new credits are considered, these old applications will be weighted in. A lot of credit applications in a short period of time looks suspicious. 

PS!

Note that the test data does not work for Store Solution API, only our ecommerce flows as well as Resurs Checkout POS/PUSH.

Resources for testing the integration are collected here.

- [Test URLs](#)
- [Verify integration](#)
- [Test Data - Sweden](#)
- [Test Data - Denmark](#)
- [Test Data - Finland](#)
- [Test Data - Norway](#)
- [Customer Field Validation \(regex\)](#)

SoapUI testing

Resurs Bank provides a set of public soapUI test cases to get you up and running quickly. soapUI is the leading free test tool for interaction with SOAP based WebServices.

By executing (and understanding) these test cases, you will gain a good understanding of how the Resurs Bank eCommerce service works, which will make the actual integration much easier.

Download suite:

[ecommerce-public-tests.xml](#)

Test URLs



Looking for production/live environment?

You can find the live URLs here: [URLs for prod/live with important checklist](#)

- Firewalling
- Payment Admin
- Merchant Portal
- Test URLs
 - Resurs Checkout
 - Hosted flow
 - Webservices
 - WSDL files
 - Simplified Shop Flow:
 - After Shop Flow:
 - Configuration:
 - Developer:
 - Service endpoints
 - Simplified Shop Flow:
 - After Shop Flow:
 - Configuration:
 - Developer:

Firewalling



How to configure firewalls

Do you have a strictly configured environment? [Take a look here](#) to get proper settings for your firewall/web services.
Link: [FAQ#HowdoIconfiguremyfirewall/network](#)

Payment Admin

For the Payment Admin go to:

<https://test.resurs.com/ecommerce-test/paymentadmin-v4/>

You can see the generated invoice for payment method INVOICE in Payment Admin, since the invoice is not sent by mail in the test-environment.

Merchant Portal

For the Merchant Portal go to:

<https://merchantportal.integration.resurs.com/login>

You can see the generated invoice for payment method INVOICE in Merchant Portal, since the invoice is not sent by mail in the test-environment.

Test URLs

Resurs Checkout

The following entry points can be used to access the Resurs Checkout API:

<https://omnitest.resurs.com/>

For more information about the usage, see [Resurs Checkout Web](#).

Hosted flow

<https://test.resurs.com/ecommerce-test/hostedflow/back-channel>

Webservices

WSDL files

Simplified Shop Flow:

<https://test.resurs.com/ecommerce-test/ws/V4/SimplifiedShopFlowService?wsdl>

After Shop Flow:

<https://test.resurs.com/ecommerce-test/ws/V4/AfterShopFlowService?wsdl>

Configuration:

<https://test.resurs.com/ecommerce-test/ws/V4/ConfigurationService?wsdl>

Developer:

<https://test.resurs.com/ecommerce-test/ws/V4/DeveloperWebService?wsdl>

Service endpoints

Simplified Shop Flow:

<https://test.resurs.com/ecommerce-test/ws/V4/SimplifiedShopFlowService>

After Shop Flow:

<https://test.resurs.com/ecommerce-test/ws/V4/AfterShopFlowService>

Configuration:

<https://test.resurs.com/ecommerce-test/ws/V4/ConfigurationService>

Developer:

<https://test.resurs.com/ecommerce-test/ws/V4/DeveloperWebService>

Verify integration

The following page is a checklist to define when an integration can be considered "Done".

Technical requirements

- XML validation - When using our services live, do not enable strict XML Schema validation as minor changes on our side can cause the integration to fail.
- Enable [preemptive authentication](#) in your HTTP client.

Legal requirements

- [Legal requirements](#)

Tests that should be performed by the integrator:

- Full integration where orders are fully managed in the integrating ERP system.

Test:	Expected result:
<i>If Invoice is to be used:</i> <ul style="list-style-type: none">• Complete an invoice purchase.• Debit the order in your ERP system.	Payment debit is successful. Invoice is received through email. (Can be viewed under the document section in Resurs Payment admin.) The invoice has correct information and logotype.
<i>If Invoice is to be used:</i> <ul style="list-style-type: none">• Complete an invoice purchase.• Debit the order in your ERP system.• Credit the complete order through your ERP system.	Crediting is possible. Credit Invoice is created with full amount. (Can be viewed under the document section in Resurs Payment admin.)
<i>If Invoice is to be used:</i> <ul style="list-style-type: none">• Complete an invoice purchase.• Annul the order either through your ERP system or in Resurs Payment admin.	Annulment is successful, verify that the order has got the correct state meaning that an annul callback reached the integrating system.
<i>If Invoice is to be used, and it is possible to select a different delivery address:</i> <ul style="list-style-type: none">• Complete an invoice purchase where the other delivery address is set.	Secure that signing is needed and that the integrating system handles both accepted and declined signing.
Complete purchase with all payment methods to be used.	The order is available in your ERP system. The order data is correctly formatted. For the payment methods " New card " and " Part payment " secure that signing is needed and that the integrating system handles both accepted and declined signing.
<i>If possible in you ERP system:</i> <ul style="list-style-type: none">• In your ERP system, complement an order with a new order row.	It is possible to complement the order.
Debit one order in your ERP system.	Debit is successful.
<i>If possible in you ERP system:</i> <ul style="list-style-type: none">• In your ERP system do a part debit of an order.	Part debit is successful.
Annul the order through your ERP system.	Annulment is successful.
<i>If possible in you ERP system:</i>	Part annulment is successful.

<ul style="list-style-type: none"> In your ERP system do a part annulment of an order. 	
<i>If Unfreeze callback is used:</i> <ul style="list-style-type: none"> Use civic test number where the payment automatically gets unfrozen. 	The callback unfreeze is received.
<i>If possible in you ERP system:</i> <ul style="list-style-type: none"> The unfreeze callback lets the order to go on to be delivered. 	Order is sent.
<i>If Automatic fraud control callback is used, IF waitForFraudControl is false in bookPayment (the order will then be sent):</i> <ul style="list-style-type: none"> Use civic test number where the payment automatically gets unfrozen. 	Callback is received with result thawed.
<ul style="list-style-type: none"> Debit one order in your ERP system. Credit the complete order through your ERP system. 	Crediting is possible.
<i>If possible in you ERP system:</i> <ul style="list-style-type: none"> In your ERP system debit an order and then credit part of the order. 	Part crediting is possible.
Complete a purchase that will not be accepted.	Denial is correctly handled.
Application form validation.	Validation works for civic and phone numbers.

- Integration where Resurs Payment admin GUI will be used to manage orders.

Test:	Expected result:
<i>If Invoice is to be used:</i> <ul style="list-style-type: none"> Complete an invoice purchase. Debit the order in Resurs Payment admin. 	<p>Payment debit is successful.</p> <p>Invoice is received through email. (Can be viewed under the document section in Resurs Payment admin.)</p> <p>The invoice has correct information and logotype.</p>
<i>If Invoice is to be used:</i> <ul style="list-style-type: none"> Complete an invoice purchase. Debit the order in Resurs Payment admin. Credit the complete order through Resurs Payment admin. 	<p>Crediting is possible.</p> <p>Credit Invoice is created with full amount. (Can be viewed under the document section in Resurs Payment admin.)</p>
<i>If Invoice is to be used:</i> <ul style="list-style-type: none"> Complete an invoice purchase. 	Annulment is successful, verify that the order has got the correct state meaning that an annul callback reached the integrating system.

<ul style="list-style-type: none"> Annul the order either through Resurs Payment admin. 	
<p><i>If Invoice is to be used, and it is possible to select a different delivery address:</i></p> <ul style="list-style-type: none"> Complete an invoice purchase where the other delivery address is set. 	Secure that signing is needed and that integrating system handles both accepted and declined signing.
Complete purchase with all payment methods to be used.	<p>The order is available in Resurs Payment admin.</p> <p>The order data is correctly formatted.</p> <p>For the payment methods "New card" and "Part payment" secure that signing is needed and that the integrating system handles both accepted and declined signing.</p>
Debit one order in Resurs Payment admin.	Debit is successful.
In Resurs Payment admin do a part debit of an order.	Part debit is successful.
Annul the order through Resurs Payment admin.	Annulment is successful.
<ul style="list-style-type: none"> Debit one order in Resurs Payment admin. Credit the complete order through Resurs Payment admin. 	Crediting is possible.
Complete a purchase that will not be accepted.	Denial is correctly handled.
Application form validation.	Validation works for civic and phone numbers.

Test Data - Sweden

Quick find

[Persons] [Organisations] [Cards] [Account] [Parameters when using Hosted flow] [Cards Payment Providers]

Phone number	0701-112233
--------------	-------------

Persons

Persons to use when testing.

 If you use `waitForFraudControl=false` the booking will get frozen automatically and you must register the callback `automaticFraudControl` to get informed when the booking is thawed.

Birthday	Gender	Civic number	Get address response	Simplified shop flow	Merchant API	Exception
1983-05-14	M	19830514 7715	Vincent Williamsson Alexandersson Glassgatan 15 41655 Göteborg	<code>bookPayment</code> returns <code>bookPaymentStatus=BOOKED</code> or <code>FINALIZED</code>	<code>Get payment</code> returns status = ACCEPTED Callback AUTHORIZED is received if callbacks are in use	Do not use this civic number when testing new- and existing card.
1950-12-02	M	19501202 6430	Oliver Liamsson Williamsson Makadamg 1 25024 Helsingborg	<code>bookPayment</code> returns <code>bookPaymentStatus=DENIED</code>	<code>Get payment</code> returns status = REJECTED Callback REJECTED is received if callbacks are in use	
1982-09-12	F	19820912 3705	Julia Liamsson Liamsson Makadamg 17 25024 Helsingborg	<code>bookPayment</code> returns <code>bookPaymentStatus=FROZEN</code> The payment will never be unfrozen.	<code>Get payment</code> returns status = FROZEN	Do not use this civic number when testing new- and existing card.
1980-01-01	F	19800101 0001	Stella Liamsson Eliasson Makadamg 3 41655 Göteborg	<code>bookPayment</code> returns <code>bookPaymentStatus=FROZEN</code> After 5 seconds the payment is unfrozen . Requires <code>waitForFraudControl=true</code>	<code>Get payment</code> returns status = FROZEN After 5 seconds the payment is unfrozen . Requires <code>handleFrozenPayments=true</code>	Do not use this civic number when testing new- and existing card.
1978-01-06	F	19780106 9241	Elsa Liamsson Alexandersson Ekslingan 20 11521 Stockholm	<code>bookPayment</code> returns <code>bookPaymentStatus=FROZEN</code> After 5 seconds the payment is annulled .	<code>Create payment</code> returns status = After 5 seconds the payment is annulled . Requires <code>handleFrozenPayments=true</code>	Do not use this civic number when testing new- and existing card.
1981-01-01	F	Coming soon	Elsa Williamsson Williamsson Ekslingan 11 41655 Göteborg	<code>bookPayment</code> returns <code>bookPaymentStatus=FROZEN</code> After 10 minutes the payment is unfrozen . Requires <code>waitForFraudControl=true</code>	<code>Create payment</code> returns status = After 10 minutes the payment is unfrozen . Requires <code>handleFrozenPayments=true</code>	Do not use this civic number when testing new- and existing card.
1989-09-19	M	19890919 4451	Vincent Liamsson Williamsson Glassgatan 12 11521 Stockholm	<code>bookPayment</code> returns <code>bookPaymentStatus=FROZEN</code> After 10 minutes the payment is annulled .	<code>Create payment</code> returns status = After 10 minutes the payment is annulled . Requires <code>handleFrozenPayments=true</code>	Do not use this civic number when testing new- and existing card.
1950-04-26	F	19500426 9741	Stella Liamsson Williamsson Makadamg 16 11521 Stockholm	<code>bookPayment</code> returns <code>bookPaymentStatus=DENIED</code>	<code>Create payment</code> returns status =	Do not use this civic number when testing new- and existing card.

1988-03-08	M	19880108 2382	Liam Liamsson Williamsson Makadamg 6 21149 Malmö	customer got no cards/accounts which allow new card/account	customer got no cards /accounts which allow new account	
------------	---	------------------	---	---	---	--

Organisations

Organisations to use when testing.

Organisation number	Gender	Civic number	Get address	Simplified shop flow	Shop Flow-(deprecated)
166997368573	M	198305147715	Pilsnerbolaget HB Glassgatan 17 25024 Helsingborg		
169468958195	M	198305147715	Grisfarmen Makadamg 12 11521 Stockholm	<code>bookPayment</code> returns <code>bookPaymentStatus=DENIED</code>	<code>submitLimitApplication</code> returns <code>decision=DENIED</code>
162177385255	M	198305147715	Pilsnerbolaget HB Glassgatan 5 25024 Helsingborg	<code>bookPayment</code> returns <code>bookPaymentStatus=DENIED</code>	<code>submitLimitApplication</code> returns <code>decision=TRIAL</code>
162830419400	M	198305147715	Grisfarmen Makadamg 15 41655 Göteborg	<code>bookPayment</code> returns <code>bookPaymentStatus=FROZEN</code>	

Cards

Card to use when testing.

Test card numbers	Government ID	Maximum limit / purchase
9000 0000 0000 0000	194601136098	0
9000 0000 0000 5000	194601136098	5 000
9000 0000 0001 0000	194601136098	10 000
9000 0000 0001 5000	194601136098	15 000
9000 0000 0002 0000	194601136098	20 000
9000 0000 0002 5000	194601136098	25 000
9000 0000 0005 0000	194601136098	50 000

To test VISA/Mastercard please see <https://developers.nets.eu/netaxeft/en-EU/docs/test-environment/#build-test-cards>

Account

Account, with the option set in the payment method to use only government ID: If agreed upon with Resurs Bank, the merchant can let the customer use an existing account without entering an account number, the government ID will fetch the account number and signing is mandatory in this case.

Parameters when using Hosted flow

The flag `allowCardPaymentWithoutCardNumber` set to `true` will only display the input field for government ID.
`allowCardPaymentWithoutCardNumber` set to `false` will display both the government ID and card number fields and the customer must enter a card number.

Government ID Account	Account number	Result
6706222616	9578405010835835	<code>bookPayment</code> returns <code>bookPaymentStatus=BOOKED</code> or <code>FINALIZED</code>

6611096337

9578105010831111

[bookPayment](#) returns bookPaymentStatus=DENIED

Cards Payment Providers

For NETS, see [this page](#).

Card number	Expire date	CVC	Result
4925 0000 0000 0004	> today	Any 3 digits	Success
4925 0000 0000 0087	> today	Any 3 digits	Reservation will fail

Test Data - Denmark

Phone number	39131600
--------------	----------

Persons

Persons to use when testing.

Birthday	Gender	Civic number	Address	Results	Shop flow (deprecated)
1985-02-14	M	140285-3877	Gorm Anker Bøgh Strøget 15 3100 Hornbæk	bookPayment returns bookPaymentStatus=BOOKED or FINALIZED	submitLimitApplication returns decision=GRANTED bookPayment returns fraudControlStatus=NOT_FROZEN
1950-12-02	M	021250-0003	Kaj Anker Anker Frederiksbergga de 1 1620 København	bookPayment returns bookPaymentStatus=DENIED	submitLimitApplication returns decision=TRIAL
1980-05-06	F	060580-3736	Kristen Bager Anker Frederiksbergga de 16 1620 København	bookPayment returns bookPaymentStatus=FROZEN The payment will never be unfrozen. Requires waitForFraudControl=true	bookPayment returns fraudControlStatus=FROZEN The payment will never be unfrozen. Requires waitForFraudControl=true
1981-04-14	F	140481-9652	Grethe Anker Anker Østergade 16 1620 København	bookPayment returns bookPaymentStatus=FROZEN After 5 seconds the payment is unfrozen . Requires waitForFraudControl=true	bookPayment returns bookPaymentStatus=FROZEN After 5 seconds the payment is unfrozen . Requires waitForFraudControl=true
1977-06-10	M	100677-2605	Preben Bager Anker Frederiksbergga de 16 1620 København	bookPayment returns bookPaymentStatus=FROZEN After 5 seconds the payment is annulled .	bookPayment returns bookPaymentStatus=FROZEN After 5 seconds the payment is annulled .
1983-11-01	F	011183-1432	Vibeke Anker Anker Strøget 16 1620 København	bookPayment returns bookPaymentStatus=FROZEN After 10 minutes the payment is unfrozen .	bookPayment returns bookPaymentStatus=FROZEN After 10 minutes the payment is unfrozen .
1984-03-24	F	240384-4340	Vibeke Anker Anker Strøget 2 3000 Helsingør	bookPayment returns bookPaymentStatus=FROZEN After 10 minutes the payment is annulled .	bookPayment returns bookPaymentStatus=FROZEN After 10 minutes the payment is annulled .
1976-12-02	M	290550-1913	Preben Anker Dunker Strøget 9 3250 Gilleleje	bookPayment returns bookPaymentStatus=DENIED	submitLimitApplication returns decision=DENIED
1988-04-23	M	2304881898		customer got no cards/accounts which allow new card/account	customer got no cards/accounts which allow new card/account

Cards

Card to use when testing.

Test card numbers	Government ID	Maximum limit / purchase
9000 0000 0000 0000	1502640867	0
9000 0000 0000 5000	1502640867	5 000
9000 0000 0001 0000	1502640867	10 000
9000 0000 0001 5000	1502640867	15 000
9000 0000 0002 0000	1502640867	20 000
9000 0000 0002 5000	1502640867	25 000
9000 0000 0005 0000	1502640867	50 000

Account

Account, with the option set in the payment method to use only government ID: If agreed upon with Resurs Bank, the merchant can let the customer use an existing account without entering an account number, the government ID will fetch the account and signing is mandatory in this case.

Parameters when using Hosted flow

The flag `allowCardPaymentWithoutCardNumber` set to `true` will only display the input field for government ID.
`allowCardPaymentWithoutCardNumber` set to `false` will display both the government ID and card number fields and the customer must enter a card number.

Government ID Account	Account number	Result
2503550949	9578205010835835	bookPayment returns bookPaymentStatus=BOOKED or FINALIZED
0505599889	9578105010831111	bookPayment returns bookPaymentStatus=DENIED

Test Data - Finland

Phone number	+3585005555127
--------------	----------------

Persons

Persons to use when testing.

Birthday	Gender	Civic number	Address	Simplified shop flow	Shop flow (deprecated)
1980-05-23	M	230580-7335	Olavi Korhonen Nieminen Kansakoulukatu 9 00100 Helsinki	<code>bookPayment</code> returns <code>bookPaymentStatus=BOOKED</code> or <code>FINALIZED</code>	<code>submitLimitApplication</code> returns <code>decision=GRANTED</code> <code>bookPayment</code> returns <code>fraudControlStatus=NOT_FROZEN</code>
1950-12-02	M	220950-9256	Juhani Korhonen Mäkelä Kalevankatu 15 00100 Helsinki	<code>bookPayment</code> returns <code>bookPaymentStatus=DENIED</code>	<code>submitLimitApplication</code> returns <code>decision=TRIAL</code>
1984-05-14	F	140584-4785	Anneli Korhonen Korhonen Kalevankatu 1 00100 Helsinki	<code>bookPayment</code> returns <code>bookPaymentStatus=FROZEN</code>	<code>bookPayment</code> returns <code>fraudControlStatus=FROZEN</code>
1981-06-02	F	020681-1008	Johanna Virtanen Virtanen Fredrikinkatu 7 00100 Helsinki	<code>bookPayment</code> returns <code>bookPaymentStatus=FROZEN</code> After 5 seconds the payment is unfrozen . Requires <code>waitForFraudControl=true</code>	<code>bookPayment</code> returns <code>bookPaymentStatus=FROZEN</code> After 5 seconds the payment is unfrozen . Requires <code>waitForFraudControl=true</code>
1981-06-02	M	300881-051B	Juhani Korhonen Nieminen Kalevankatu 9 00100 Helsinki	<code>bookPayment</code> returns <code>bookPaymentStatus=FROZEN</code> After 5 seconds the payment is a nullled .	<code>bookPayment</code> returns <code>bookPaymentStatus=FROZEN</code> After 5 seconds the payment is annulled .
1980-09-10	F	100980-576X	Maria Korhonen Korhonen Kansakoulukatu 1 00100 Helsinki	<code>bookPayment</code> returns <code>bookPaymentStatus=FROZEN</code> After 10 minutes the payment is unfrozen . Requires <code>waitForFraudControl=true</code>	<code>bookPayment</code> returns <code>bookPaymentStatus=FROZEN</code> After 10 minutes the payment is unfrozen . Requires <code>waitForFraudControl=true</code>
1980-09-10	F	230982-3064	Helena Virtanen Mäkelä Fredrikinkatu 10 33100 Tampere	<code>bookPayment</code> returns <code>bookPaymentStatus=FROZEN</code> After 10 minutes the payment is annulled .	<code>bookPayment</code> returns <code>bookPaymentStatus=FROZEN</code> After 10 minutes the payment is annulled .
1978-01-02	F	180650-344E	Kaarina Virtanen Virtanen Kalevankatu 7 00100 Helsinki	<code>bookPayment</code> returns <code>bookPaymentStatus=DENIED</code>	<code>submitLimitApplication</code> returns <code>decision=DENIED</code>
1987-09-15	M	150987-069L		customer got no cards/accounts which allow new card/account	customer got no cards/accounts which allow new card/account

Organisations

No payment methods for Finish organisations exist today. Contact your Resurs Bank sales representative if you want to support Finish organisations.

Organisation number	Gender	Civic number	Get address	Shop Flow	Simplified shop flow
6014969-1	M	230580-7335	Sika Maatila Fredrikinkatu 3 00100 Helsinki		
4967996-6	M	230580-7335	Olutpanimo Kansakoulukatu 9 00100 Helsinki	<code>submitLimitApplication</code> returns <code>decision=DENIED</code>	<code>bookPayment</code> returns <code>bookPaymentStatus=DENIED</code>
1105814-4	M	230580-7335	Olutpanimo Kansakoulukatu 5 00100 Helsinki	<code>submitLimitApplication</code> returns <code>decision=TRIAL</code>	<code>bookPayment</code> returns <code>bookPaymentStatus=DENIED</code>

Cards

Card to use when testing.

Maximum limit / purchase is € 5000.

Test card numbers	Government ID	€
9000 0000 0000 0000	100370-897V	0
9000 0000 0000 0500	100370-897V	500
9000 0000 0000 1000	100370-897V	1000
9000 0000 0000 1500	100370-897V	1500
9000 0000 0000 2000	100370-897V	2000
9000 0000 0000 2500	100370-897V	2500
9000 0000 0000 5000	100370-897V	5000

To test VISA/Mastercard please see :<https://shop.nets.eu/web/partners/test-cards>

Account

Account, with the option set in the payment method to use only government ID: If agreed upon with Resurs Bank, the merchant can let the customer use an existing account without entering an account number, the government ID will fetch the account number and signing is mandatory in this case.

Parameters when using Hosted flow

The flag `allowCardPaymentWithoutCardNumber` set to `true` will only display the input field for government ID.
`allowCardPaymentWithoutCardNumber` set to `false` will display both the government ID and account number fields and the customer must enter a account number.

Government ID Account	Account number	Result
260781-3930	9578105010835835	bookPayment returns bookPaymentStatus=BOOKED or FINALIZED
140863-121M	9578105010831111	bookPayment returns bookPaymentStatus=DENIED

Test Data - Norway

Phone number	22563733
--------------	----------



All phone numbers are intended for test of functionality between the merchant and Resurs, not for tests between the merchant and end customer point. At creation time they are tested not to be real but at any time, without notice, they might be picked up for use by tele companies. If they then are used for tests, real people might get notifications, depending on use. The numbers have no logical functionality at Resurs other than to fetch an address, so if you want to test towards end customer point you can use your own private numbers.

Persons

Persons to use when testing.

Birthday	Gender	Civic number	Phone number	Simplified shop flow	Shop flow (deprecated)
1972-08-18	M	180872-48794	40000007	bookPayment returns bookPaymentStatus=BOOKED or FINALIZED	submitLimitApplication returns decision=GRANTED bookPayment returns fraudControlStatus=NOT_FROZEN
1949-01-07	M	010249-24986	40000002	bookPayment returns bookPaymentStatus=DENIED	submitLimitApplication returns decision=TRIAL
1949-08-02	F	020849-29428	40000003	bookPayment returns bookPaymentStatus=FROZEN	bookPayment returns fraudControlStatus=FROZEN
1982-06-23	F	230682-01608	40000004	bookPayment returns fraudControlStatus=FROZEN After 5 seconds the payment is unfrozen. Requires waitForFraudControl=true	bookPayment returns fraudControlStatus=FROZEN After 5 seconds the payment is unfrozen. Requires waitForFraudControl=true
1978-01-05	F	050178-18440	40000013	bookPayment returns fraudControlStatus=FROZEN After 5 seconds the payment is annulled.	bookPayment returns fraudControlStatus=FROZEN After 5 seconds the payment is annulled.
1982-07-01	F	010782-12868	40000005	bookPayment returns fraudControlStatus=FROZEN After 10 minutes the payment is unfrozen. Requires waitForFraudControl=true	bookPayment returns fraudControlStatus=FROZEN After 10 minutes the payment is unfrozen. Requires waitForFraudControl=true
1977-04-03	M	030477-05311	40000014	bookPayment returns fraudControlStatus=FROZEN After 10 minutes the payment is annulled.	bookPayment returns fraudControlStatus=FROZEN After 10 minutes the payment is annulled.
1949-02-26	M	260249-14002	40000006	bookPayment returns bookPaymentStatus=DENIED	submitLimitApplication returns decision=DENIED
1988-02-27	M	270288-09552		customer got no cards/accounts which allow new card/account	customer got no cards/accounts which allow new card/account

Organisations

No payment methods for Norwegian organisations exist today. Contact your Resurs Bank sales representative if you want to support Norwegian organisations.

Organisation number	Gender	Civic number	Shop Flow	Simplified shop flow
892831270	M	180872-48794		
996030962	M	180872-48794	submitLimitApplication returns decision=DENIED	bookPayment returns bookPaymentStatus=DENIED
950576839	M	180872-48794	submitLimitApplication returns decision=TRIAL	bookPayment returns bookPaymentStatus=DENIED

Cards

Card to use when testing.

Test card numbers	Government ID	Phone number	Maximum limit / purchase
9000 0000 0000 0000	16066405994	40000010	0
9000 0000 0000 5000	16066405994	40000010	5 000
9000 0000 0001 0000	16066405994	40000010	10 000
9000 0000 0001 5000	16066405994	40000010	15 000
9000 0000 0002 0000	16066405994	40000010	20 000
9000 0000 0002 5000	16066405994	40000010	25 000
9000 0000 0005 0000	16066405994	40000010	50 000

To test VISA/Mastercard please see: <https://shop.nets.eu/web/partners/test-cards>

Account

Account, with the option set in the payment method to use only government ID: If agreed upon with Resurs Bank, the merchant can let the customer use an existing account without entering an account number, the government ID will fetch the account number and signing is mandatory in this case.

Parameters when using Hosted flow

The flag `allowCardPaymentWithoutCardNumber` set to `true` will only display the input field for government ID.
`allowCardPaymentWithoutCardNumber` set to `false` will display both the government ID and card number fields and the customer must enter a card number.

Government ID Account	Phone number	Account number	Result
25019218190	40000011	9578305010835835	<code>bookPayment</code> returns bookPaymentStatus=BOOKED or FINALIZED
01046017362	40000012	9578105010831111	<code>bookPayment</code> returns bookPaymentStatus=DENIED

Testing Vipps

Installing the Vipps test app on your phone

In order to test Vipps payments, we need to use the test app "Vipps MT" on a phone. Follow these instructions to install it: [Vipps test apps](#)

If using an iPhone; when TestFlight is installed, follow this Vipps invitation link and then continue with step 2 from the guide above - <https://testflight.apple.com/join/hTAYrwea>

Test phone number that is to be used: 97340944

Code/pin to that is to be used: 1236

NOTE

This number is shared so someone else might be using it at the same time. If the payment has not appeared in your Vipps test app within 30 seconds, redo the purchase from scratch

Paying with Vipps

To test a Vipps payment, create a payment and choose Vipps as payment method and continue with the purchase, see below:

1. Use the test number to retrieve address.
2. Select "Vipps" as the payment option and pay.
3. Enter the test number if it is not autofilled:

4. Press "Next" and open the app on your phone.
5. Using the code (1236), log in and confirm the payment.

Customer Field Validation (regex)

The correct location for this is in the Development section:

[Customer data - Regular expressions](#)

FAQ

Questions we often receive includes:

- My environment seems to be unavailable for Resurs
 - Are your site available from internet?
 - Are you using ngrok?
 - Are you using proxy?
 - Are you hard blocking unknown traffic in your firewall?
 - Are you allowing traffic from certain traffic in your webserver?
 - Are you using correct TLS version?
- How do I configure my firewall/network?
 - Local networks / Hosts (or "your callbacks is not working")
- "Limitforms"?
- Payments methods: You mean 'invoice'?
- Should I set preferred payment ID?
- Can I skip the prepare signing part?
- Client libraries for my programming language, where?
- Good to know
 - Can I skip registering ANNULMENT and FINALIZATION callback and make UPDATE handle the work?
 - Simplified flow and payment providers
 - Signing processes and mismatching orders in the interfaces
 - How long will you wait for callbacks (during the connection itself) to arrive?
- Troubleshooting
 - First step of error checking
 - Security warnings
 - When I add DISCOUNT in hosted flow I'm not allowed to make a purchase
 - Slow callbacks in Magento 2
- The jungle of SSL
 - Where and whats about SSL
 - Where do I find a verified issuer?
 - Visualization of above
- Do we support IPv6?
 - Missing a question? Contact us!



Do NOT use TLS 1.1 or lower!

TLS below TLSv1.2 is no longer supported. Trying to connect to our services with lower versions will fail!

My environment seems to be unavailable for Resurs

Here's a checklist for what you should test before reporting problems to Resurs

- **Are your site available from internet?**

Make sure your site are publicly available from internet (DNS configuration). Please see the section about local networks below.

- **Are you using ngrok?**

ngrok is a reverse-proxy service that many times helps testing things without having a publicly available server. However, ngrok has low reputation which also means that the host is blocked on firewall level at Resurs. Make sure you do not use ngrok when testing your API's. If the url is present in a payload, the traffic will be dropped by default.

- **Are you using proxy?**

In some cases, proxies are - at least in test - not supported. As of december 2021, proxy support is only available in production environments.

- **Are you hard blocking unknown traffic in your firewall?**

See below, how you should configure your firewall to receive traffic from Resurs.

- **Are you allowing traffic from certain traffic in your webserver?**

Like in the matter of firewalls, a webserver could also block traffic from unknown locations. See the firewall section how to configure this.

- **Are you using correct TLS version?**

As shown above, TLS below version 1.2 is not supported.

How do I configure my firewall/network?

If you have restricted firewalls that explicitly need to define what communication is allowed, you should consider open for the CIDR-blocks below.

Please note that the primary ports for a web-server should be 443 ([https](https://)) as Resurs no longer support plain text callbacks at port 80.

Network CIDR-block	Range equivalent	Broadcast	Netmask	Firewall settings	Allow ports	AS Number
192.121.110.0/24	192.121.110.1 - 192.121.110.254	192.121.110.255	255.255.255.0	Allow from 192.121.110.*	443 (https)	AS35814

194.68.237.0/24	194.68.237.1 - 194.68.237.254	194.68.237.255	255.255.255.0	Allow from 194.68.237.*	443 (https)	AS35814
91.198.202.0/24	91.198.202.1 - 91.198.202.254	91.198.202.255	255.255.255.0	Allow from 91.198.202.*	443 (https)	AS35814

Local networks / Hosts (or "your callbacks is not working")

If you are using local networks during tests, be aware that all teststings does not work. There are a few rules to consider for "locals":

- Local networks like 192.168.0.0/16, 172.16.0.0/16, 10.0.0.0/8, etc (see https://en.wikipedia.org/wiki/Private_network) are unreachable. Testing callbacks on such networks won't work.
- Hostnames assigned by local isolated DNS servers only does not work - we won't be able to resolve them unless they reside on a reachable internet connection. This includes local zones like localhost (example: my-dev.localhost, my-site.dev, etc).

"Limitforms"?

Our limit forms are a way to handle dynamic input from the customer. Normally the web service interface would specify which information that should be sent in when doing a credit application or a card purchase. We've instead chosen to return a set of fields we would like the customer to fill in. These fields carries information enough to render a HTML form in the web shop asking the customer for these values. Upon POST, these values are to be compiled into a xml document and sent to us.

In this way we can handle all kind of [Payment methods](#) in **one** flow, and we can request more data if needed.



Because of this, the shop never can assume that a particular field will or will not show up for a certain payment method. The shop **must** render the limit form *dynamically*.

[Read more about limit applications...](#)

Payments methods: You mean 'invoice'?

A common misunderstanding is that there is a fixed number of [payment methods](#) Resurs Bank offers, and that these payment methods have different 'flows', i.e. different functions needs to be invoked in, say, a card purchase and an invoice purchase. This is **not** the case. Resurs Bank has implemented [one flow](#) which incorporates all payment methods.

Resurs Bank can add and remove payment methods with configuration. It's up to the implementor to hard-code payment methods on the client side, but further programming would, of course, be required to change the payment method setup in this case.



Due to the dynamic nature of [payment methods](#), payment methods identities used in the [shop flow](#) might be different in live versus test! Do not hard-code them!

Should I set preferred payment ID?

To more easily identify a payment you should always set the `preferredPaymentId` to the identity of your order in [bookPayment](#). If this is not set, the payment will be getting a generated identity.

Can I skip the prepare signing part?

In Resurs Checkout, absolutely! But in simplifiedShopFlow, as signing is triggered by different rules, you should always implement the signing flow.

Client libraries for my programming language, where?

Many integrators expect Resurs Bank to provide client side libraries. Go check under the section [Development](#) to find out what we're building in house.

Good to know

Can I skip registering ANNULMENT and FINALIZATION callback and make UPDATE handle the work?

Yes. UPDATE is fired each time an order status is changed. This is why you also can not ignore neither BOOKED (created) nor UNFREEZE (not a status update), since they don't specifically update the status of a payment.

Simplified flow and payment providers

When we use Swish, we get a redirect-url on SIGNING that leads us to swish and then back to a success-page, almost like in the case of INVOICE, etc. Do we have to run a bookSignedPayment in those cases?

No. The booking of signed payments are completed at Resurs side in such cases, even if the SIGNING-url is used and is it handled automatically. When you get a SIGNING where bookSignedPayment is required, the payment will first be redirected a it always do to your signing-page.

Signing processes and mismatching orders in the interfaces

A rare problem, but still existing, is a case when an order is still active at Resurs Bank but cancelled in the store platform. This is something that usually should not be possible due to the way RCO works:

- Customer is completing the order - the normal case (like Woocommerce) is that the order is created in the webstore before Resurs Bank.
- If the customer hasn't completed the order in time, there's a slight chance that the order is now cancelled in the platform.
- If the customer is required to sign the payment before proceeding, the order is in this moment not created at Resurs Bank yet. So if the order is cancelled in the platform, it should be in order.
- If the customer signs the payment, it is also created at Resurs Bank. A callback is sent back to the platform to update the payment regardless if the customer returns to the store on success.
- The order, depending on the platform should be properly updated or rejected (Woocommerce updates the cancelled order as successful now). So in this case, the order **should** exist in both places.
- If your platform is rejecting cancelled orders they should still be there.

How long will you wait for callbacks (during the connection itself) to arrive?

The default callback timeout are set to 5 seconds as of april 24 2019. If you wonder for how long we try to resend callbacks to you, [look here instead](#).

Troubleshooting

First step of error checking

Make sure that you're checked with [The "not a bug" list](#) before considering the problem as a remote error. You can also take a look at [Errors, problem solving and corner cases](#), to see if it might be other problems that can be "self solvable".

XML validation

When using our services live, do not enable strict XML Schema validation as minor changes on our side can cause the integration to fail.

Signing troubles

When you implement our signing requirement, please *redirect* the customer to the link the `prepareSigning` function provides. **Do not** try to embed the signing page in the webshop as an `iframe`.

Security warnings

Check that you use HTTPS and that you have a valid and issued certificate, not a self signed one. If you run PHP and anything else but RCO, make sure you also have SoapClient and XML-support installed.

When I add DISCOUNT in hosted flow I'm not allowed to make a purchase

Did you add VAT to the discount line? If so, this might be the cause. The flow does not allow negative values when discount is added with VAT.

Slow callbacks in Magento 2

In some cases (relatively rare), callbacks might feel a bit slow. The best indicator of this is when the BOOKED callback are triggered from Resurs Bank and Magento's order status notifications are repeatedly filled with the same BOOKED notice several times.

This is usually not bound to the plugin itself. Instead most of it is about the e-mail rendering and how long it takes for a customer confirmation mail to be sent during the callback. The first bottle neck can be found in the e-mail sending itself; if the mail server are misconfigured, this also could cause a noticeable slowdown in the system. Usually this depends on resolving and the e-mail communication which can not be fixed by us.

The second problem is bound to the e-mail rendering rather than the sending. This time the real problem could be found in the template rendering. A live example we have is the theme "Luma" and the demo store itself. When the callback BOOKED occurs in the platform, a bunch of templates starts to render before the e-mail is being sent. In our case, it all starts at `vendor/magento/module-sales/view/frontend/email/order_new_guest.html`, where this row are rendered:

```
{template config_path="design/email/header_template"} }
```

When this happens the template `vendor/magento/module-email/view/frontend/email/header.html` is being read, and amongst others some css-data and variables are fetched. This might take some time do do, and when we use the Luma theme it gets slower. We should say there's no other themes that does the same, but sometimes it could be good to check out the themes also.

The absolutely simplest, but cron-dependent solution is to make Magento go async. By means, e-mail are not being sent during actions like callback, but may also be delayed depending on how often cronjobs are running. *I'd say this solution is highly dependent on functional cronjobs.*

Async mode

```
bin/magento config:set sales_email/general/async_sending 1
```

To reactivate the synchronous sending again, you instead run this:

Sync mode

```
bin/magento config:set sales_email/general/async_sending 0
```

The jungle of SSL



Do NOT use TLS 1.1 or lower!

TLS below TLSv1.2 is no longer supported. Trying to connect to our services with lower versions will fail!

Where and whats about SSL



Callbacks over non-SSL links

When this FAQ is written, we still allow callbacks over http, but this will change soon.

It is necessary that your entire site runs with SSL support active (https), both in test and production. The reason is that our callbacks does not support http-calls.

I have certificate here, issued by myself. Is this OK?

No!

Resurs Bank does not communication with self signed certificates. Neither we do with invalidated (revoked) or expired certificates, so make sure you get it from a verified issuer. But here's some examples:

[Digicert](#)

[Network Solutions](#)

[Godaddy / Starfield Technologies](#)

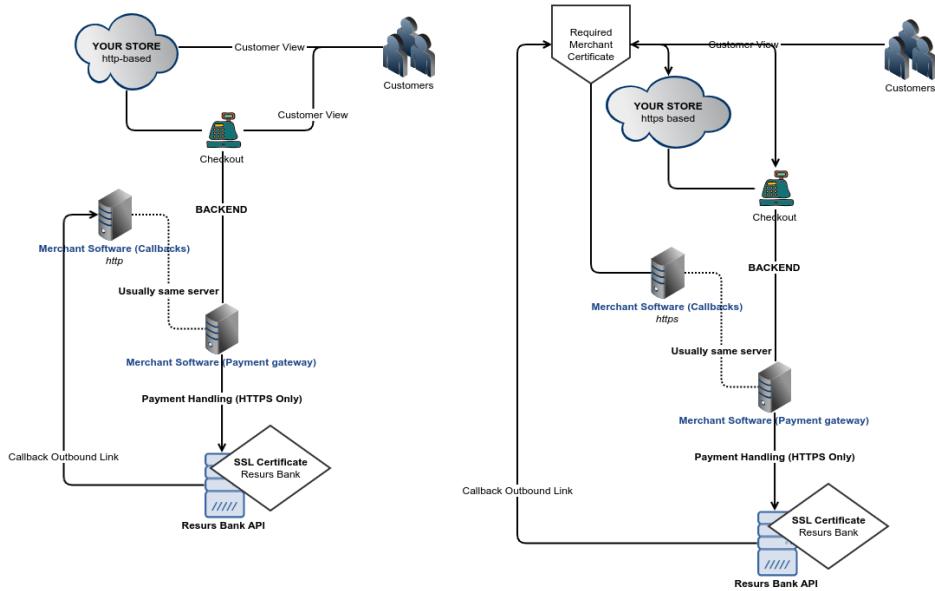
[Letsencrypt](#) (Issues free certificates, supports wildcard certificates, but requires renewal every three months)

Where do I find a verified issuer?

There's a lots of issuers available on the internet, so there's nearly impossible to recommend a verified issuer.

Visualization of above

SSL-StoreFront vs SSL-ResursBank



Do we support IPv6?

No - and partially yes.

Resurs Bank is currently running, inbound, on IPv4 layers only. However, IPv6 are supported if your site support both protocols. The support is based, halfway, on [paymentData](#) (simplified flow) and [getAddress \(SE\)](#) (simplified flow). "Half way" means that you can add the customer remote ip address into the payment information when finishing orders. You can also add the remote ip during a getAddress-lookup. There are no other connectivity support than this. It is important to know, that if you run a site with [IPv6 connectivity only](#) (which is actually fully possible to do as of today) your solution will fail: *there are no such support available at all*. Hosted flow and Resurs Checkout is also IPv4 only.

Missing a question? [Contact us!](#)

The "not a bug" list

On this page, we keep a list of errors and problems that is normally not related to plugins or installed modules. Sometimes errors are generated due to missing components or external access is denied from your platform. Sometimes this happens due to unmet requirements. Here's some of them. If this page does not cover your problem, you can also take a look at [Errors, problem solving and corner cases](#).

Error/Code	Description	Solution (Eventually)
error: 14090086 :SSL routines:ssl3_get_server_certificate:certificate verify failed	Root certificate bundles are missing on your server, or is placed in a path where your server is not looking.	For PHP cases, this can be fixed by editing your server's php.ini: See the variables for openssl.cafile, openssl.capath. Another solution is to try to add the missing files. In many cases, the missing file is ca-certificates.crt, and in a "standard" Ubuntu platform, they are located in /etc/ssl/certs - openssl usually delivers this file by the dependency ca-certificates (<i>apt-get install ca-certificates</i>).
error: 14094410 :SSL routines:ssl3_read_bytes:sslv3 alert handshake failure	SSL handshake errors due to unsupported SSL version. This usually happens when your client tries to communicate with a disabled or non-existent SSL version.	Upgrade openssl and/or your platform.
ERROR: Error 35: error: 14077458 :SSL routines:SSL23_GET_SERVER_HELLO:reason(1112)	Almost like the one above. SSL protocol error.	Make sure your ssl supports the right protocol.
SOAP-ERROR: Parsing WSDL: Couldn't load from 'url?wsdl' : failed to load external entity "url?wsdl"	If this error is a "standalone" where only this message is visible, there might be an SSL problem again.	This is normally either a credential error or a sign that your system is missing enabled ssl-drivers. The EComPHP -library is normally trying to defeat this error message by revealing the "true" error since credential errors, especially on older PHP-systems, tend to hide such error messages when credentials are used. This is a bug in PHP, known since 2006 where 401-Unauthorized errors are being handled as notices rather than a "real" exceptions, and is normally not caught by PHP. This is probably not fixed since PHP 7+ should be able to handle this better (unconfirmed).

Contact

Production Support (incident):

support@resurs.se

After sales support

If you have problems with payments in production.

Mon – Fri: 08.00 – 18.00 CET

Call +46 42-38 20 50

ehandel@resurs.se

Technical support during onboarding and this documentation

onboarding@resurs.se

Sales

<https://www.resursbank.se/foretag/driving-retail-sales/>

Frequently asked questions

[Humanity](#)

Merchant API 1

NOTE!

This API is generally not to be implemented as of 2023. Instead, see Merchant API 2 here: [Merchant API 2.0](#)

Merchant API:

Documentation links:

[Merchant API Service](#)
[Merchant Credit Application Service](#)
[Merchant Payment Service](#)
[Merchant Event Service](#)
[Merchant Bonus Service](#)

[Describes the minimum application flow for a new Resurs Bank Card](#)

[Describes minimal payment with debit, at the same time as applying for a new Resurs Bank Card](#)

[Resurs Merchant API.postman_collection.json](#)

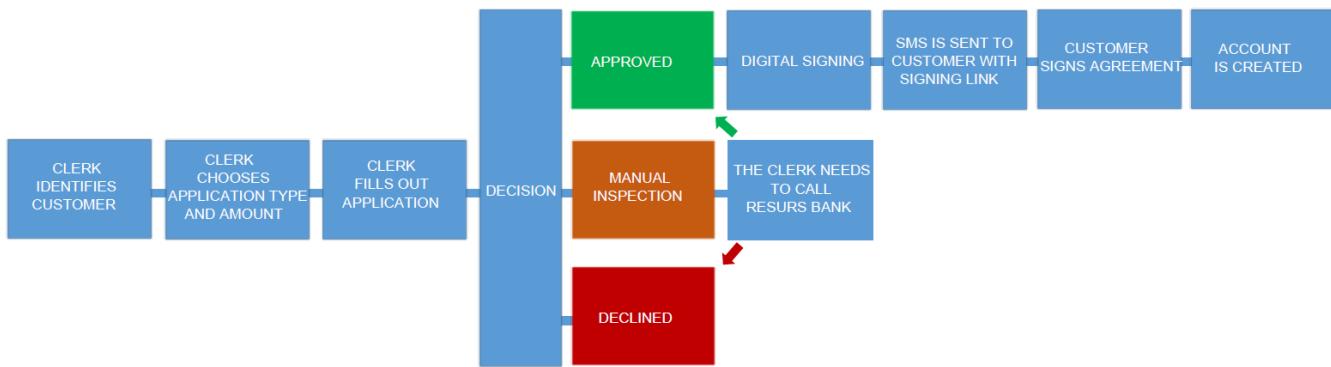
I Postman:

1. Import top left --> select file --> Import
2. Högerklicka i Collections-fliken --> Edit
3. Variables-fliken --> sätt ett värde på client_id och client_secret
4. Authorization-fliken --> klicka Get New Access Token

Kan nu anropa t.ex. *GET Basic information of accessible stores under Merchant API (Settings & Info)*.

Credit Application

The credit application flow is used when a customer wants to apply for an account without making a purchase at the same time as the application is made.



Typical flow

Authentication

Every request requires an authorization header with a Bearer-token. A token lasts for 3600 seconds (1 hour). To get a token you may use your test-credentials received from Resurs Bank:

```
client_id  
client_secret  
scope= mock-merchant-api
```

Curl to get token

```
curl --location --request POST 'https://apigw.integration.resurs.com/api/oauth2/token'  
--header 'accept: application/json' \  
--header 'Content-Type: application/x-www-form-urlencoded' \  
--data-urlencode 'client_id=' \  
--data-urlencode 'client_secret=' \  
--data-urlencode 'scope=mock-merchant-api' \  
--data-urlencode 'grant_type=client_credentials'
```

Step-by-step for creating an account after authentication

1. Get available stores

A client may have access to multiple stores, therefore we need to know which store to make the application or payment for. This can be done by getting the available stores. Each store has a store-id. This id will be used in the next step to specify for which store we would like to get the payment methods.

Curl to get available stores

```
curl --location --request GET 'https://apigw.integration.resurs.com/api/mock_merchant_api_service/stores'  
--header 'Authorization: Bearer <TOKEN>'
```

2. Get available payment methods

A store may have multiple payment methods available. The list of available payment methods will show what payment methods there are to apply from at the chosen store. Each payment method has a paymentmethod id, which will be used in the next step, when the application is created.

Curl to get available payment methods

```
curl --location --request POST 'https://apigw.integration.resurs.com/api/mock_merchant_api_service/stores/{store_id}/payment_methods'  
--header 'Content-Type: application/json' \  
--header 'Authorization: Bearer <TOKEN>' \  
--data-urlencode 'paymentmethod_id='
```

```
--data-raw '{
"countryCode": "SE, FI, NO, DK",
"customerType": "NATURAL"
}'
```

3. Create an application

ExternalId is will work as a searchable reference.

Curl to create an application

```
curl --location --request POST 'https://apigw.integration.resurs.com/api/mock_merchant_credit_application_service/applications'
--header 'Content-Type: application/json' \
--header 'Authorization: Bearer <TOKEN>' \
--data-raw '{
"applicationData": {},
"applicationType": "NEW_ACCOUNT",
"customer": {
"countryCode": "SE, FI, NO, DK",
"customerType": "NATURAL",
"governmentId": "",
"email": "test@test.com",
"mobile": "012345678",
"phone": "012345678"
},
"paymentMethodId": "",
"requestedAmount": "1200",
"storeId": "",
"externalId": "MyApplication_1"
}'
```

4. Complete application

In this step we are ready to complete the application. Creator is for example the clerk filling out the application for the customer. It is important to note the status after this completion;

APPROVED = the application was successful and are done

REJECTED = the application could not be approved, the application is unusable

INSPECTION = the customer must contact Resurs Bank for a manual inspection of the application

SIGNING_REQUIRED = the customer needs to approve the application by a digital signature

Curl to complete application

```
curl --location --request POST 'https://apigw.integration.resurs.com/api/mock_merchant_api
--header 'Content-Type: application/json' \
--header 'Authorization: Bearer <TOKEN>' \
--data-raw '{
"countryCode": "SE, FI, NO, DK",
"customerType": "NATURAL"
}'
```

5. Redirect to signature

The success- and fail-URLs will be called when the customer is done and the application is approved or rejected.

Curl to redirect to signature

```
curl --location --request POST 'https://apigw.integration.resurs.com/api/mock_merchant_cre
--header 'Content-Type: application/json' \
--header 'Authorization: Bearer <TOKEN>' \
--data-raw '{'
```

```
"failUrl": "https://google.com/search?q=fail",
"successUrl": "https://google.com/search?q=success"
}
```

Credit Application and Payment

The credit application and payment flow is used when a customer wants to apply for an account and making a purchase at the same time.



Authentication

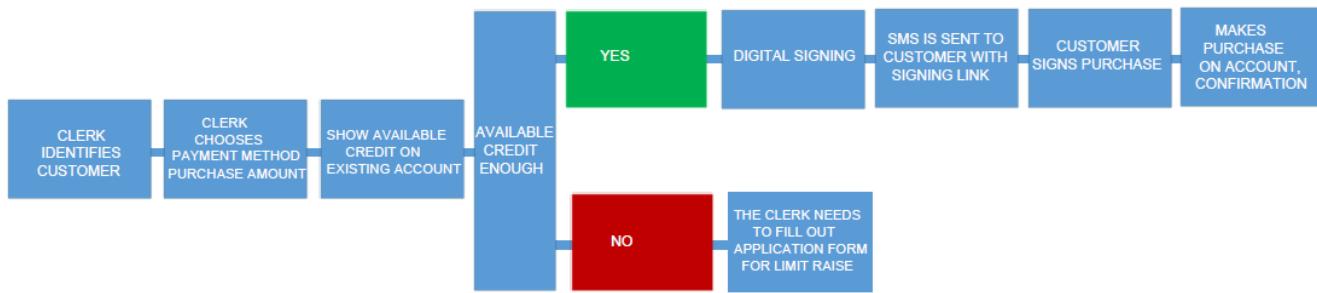
Every request requires an authorization header with a Bearer-token. A token lasts for 3600 seconds (1 hour). To get a token you may use your test-credentials received from Resurs Bank:

```
client_id  
client_secret  
scope= mock-merchant-api
```

Curl to get token

```
curl --location --request POST 'https://apigw.integration.resurs.com/api/oauth2/token' \  
--header 'accept: application/json' \  
--header 'Content-Type: application/x-www-form-urlencoded' \  
--data-urlencode 'client_id=' \  
--data-urlencode 'client_secret=' \  
--data-urlencode 'scope=mock-merchant-api' \  
--data-urlencode 'grant_type=client_credentials'
```

Payment with Existing Account



Typical flow

Authentication

Every request requires an authorization header with a Bearer-token. A token lasts for 3600 seconds (1 hour). To get a token you may use your test-credentials received from Resurs Bank:

```
client_id  
client_secret  
scope= mock-merchant-api
```

Curl to get token

```
curl --location --request POST 'https://apigw.integration.resurs.com/api/oauth2/token' \  
--header 'accept: application/json' \  
--header 'Content-Type: application/x-www-form-urlencoded' \  
--data-urlencode 'client_id=se_netonnet' \  
--data-urlencode 'client_secret=fsNwJ93CGc2LkXfJ' \  
--data-urlencode 'scope=mock-merchant-api' \  
--data-urlencode 'grant_type=client_credentials'
```

Step-by-step for creating, authorizing and debiting an account after authentication

1. Get available stores

A client may have access to multiple stores, therefore we need to know which store to make the application or payment for. This can be done by getting the available stores. Each store has a store-id. This id will be used in the next step to specify for which store we would like to get the payment methods.

Curl to get available stores

```
curl --location --request GET 'https://apigw.integration.resurs.com/api/mock_merchant_api' \  
--header 'Authorization: Bearer <TOKEN>'
```

2. Get available payment methods

A store may have multiple payment methods available. The list of available payment methods will show what payment methods there are to apply from at the chosen store. Each payment method has a paymentmethod id, which will be used in the next step, when the application is created.

Curl to get available payment methods

```
curl --location --request POST 'https://apigw.integration.resurs.com/api/mock_merchant_api' \  
--header 'Content-Type: application/json' \  
--header 'Authorization: Bearer <TOKEN>' \  
--data-raw '{  
"countryCode": "SE, FI, NO, DK",
```

```
"customerType": "NATURAL"  
}
```

3. Create a payment

Supply the storeId and paymentmethodid from previous steps. Take note of the id returned in the response. We can use it to debit, anull or authorize the payment.

Curl to create a payment

```
curl --location --request POST 'https://apigw.integration.resurs.com/api/mock_merchant_payment_service/payments' \  
--header 'Content-Type: application/json' \  
--header 'Authorization: Bearer {{merchant-api-token}}' \  
--data-raw '{  
  "cart": {  
    "cartItems": [  
      {  
        "quantity": "1",  
        "totalAmountIncVat": "750",  
        "totalAmountVat": "125",  
        "type": "NORMAL",  
        "unitPriceIncVat": "750",  
        "vatRate": "0.2",  
        "description": "TV",  
        "externalId": "tv_123"  
      }  
    ],  
    "orderId": "order-ref",  
    "totalAmountIncVat": "750",  
    "totalAmountVat": "125",  
    "totalExVat": "625"  
  },  
  "customer": {  
    "countryCode": "SE, NO, FI, DK",  
    "customerType": "NATURAL",  
    "governmentId": "",  
    "email": "test@test.com",  
    "mobile": "012345678"  
  },  
  "externalId": "payment-ref",  
  "paymentMethodId": "",  
  "storeId": ""  
}'
```

4. Debit the payment

This means that we are ready to complete the payment and charge the customer for the purchased goods. Creator is for example the clerk debiting customer.

Curl to debit a payment

```
curl --location --request POST 'https://apigw.integration.resurs.com/api/mock_merchant_payment_service/payments/e6f67bac-64b8-4790-b52e-724348503c1b' --header 'Content-Type: application/json' \  
--header 'Authorization: Bearer {{merchant-api-token}}' \  
--data-raw '{  
  "creator": "John"  
}'
```

4. Redirect to e-signing

URLs for test & production

URL for token:

Test: <https://apigw.integration.resurs.com/api/oauth2/token>

Production: <https://apigw.resurs.com/api/oauth2/token>

Logotypes new

Present a Resurs-logo with your Resurs [payment methods](#).

[Here are logos for download...](#)

Swish för Handel - Resurs Technical supplier

In order to offer Swish for the customers, there are a few steps that needs to be done prior to the payment method is available on your production account;

1. Verify with your salesperson that Swish is available for your chosen/designated integration
2. Sign an agreement with your bank for *Swish för Handel* and choose Resurs as technical supplier at your bank. (Selecting Resurs bank as technical provider is done through the bank that you have an agreement with.)
3. When done, Swish will automatically send an email to the Integration-team (onboarding@resurs.se) that your organizational number now has connected to Resurs for Swish för Handel
4. Resurs configures the payment method and uploads it to your production account

Depending on your integration, you may need to update/fetch your payment methods in order for the payment method to appear. If uncertain that is required in your unique case, please email onboarding@resurs.se