Pckasse
Webshop integration
Version 1.94

Implemented in Pckasse 3.1.5.50

Techical documentation

Date: 7. june 2021

PCK
www.pckasse.no

DELTASOFT

Pckasse "WebshopWSService" Description

Introduction

Pckasse connects to a SOAP webservice to exchange data with the webshop. Pckasse initiates all communication between the POS and the webshop. There is no way for the webshop to initialize conversation. This is because Pckasse is locally installed and often behind a firewall.

We need to synchronize the article register for all articles that is “Visible on web” after the webshop is created. A button in Pckasse transfers everything. This button in only used at the initial setup.

Every time an object relevant to the webshop is changed or created an appropriate function is called to synchronize the registers. This call is queued and removed if successful. If an error is returned that item will be flagged and resent when the error is corrected in Pckasse.

getOrders is called when the user chooses to download weborders in Pckasse.
(There is an url that can be called to initiate getOrders. Contact us if that is something you need. For example, takeaway, where the order should be made immediately)

Pckasse calls “updateOrderStatus”, one per order, after getOrders is called, with a status indicating success or failure.

Unless it is a credit order Pckasse expects the fetched orders to be new, valid and paid for. The next step is to deliver the items. For prepaid orders, Pckasse will call “UpdateOrderStatus” with the items delivered. Status is set to “part delivered” or “fully delivered”. If status is “fully delivered” then delete the remaining order lines, if any. Because payment can be declined, Pckasse waits for the response to see if delivery is accepted.

Package number can be specified, or it can be sent later with “UpdatePackageInfo”

If the customer returns the goods, Pckasse will use CreditOrder to return payment to the customer. It is possible to specify an additional amount, but the total cannot exceed the captured amount.

For the Webshop API to work, it must be activated in the Pckasse Licence for 3. party implementations.

Technical Info

Integrated webshop WSDL:
http://webshop.deltasoft.no/DeltaWebshop/WS/WebshopWS?wsdl

If you want to make your own webshop, you must make a WSDL equal to this, and override webshop URL in Pckasse.
There are tools available to generate stubs from wsdl (For example WSDL.exe in visual studio)

Following is a description of the functions that you need to implement for Pckasse to work.

Changes since 1.93

Better documentation

- Funtion **updateOrderStatus**
    - Class **updateOrderResponse**
        - Property a**mount**

Shared classes

- Class **insertUpdateResponse**
  - Property **deltaId** as **Integer**
    ID of the involved object in the webshop
  - Property **errorHelpLink** as **String**
    Optional (not used in Pckasse yet)
  - Property **errorMessage** as **String**
    Used if humanErrorMessage is blank
  - Property **humanErrorMessage** as **String**
    Shown to the user if operationResult < > 0
  - Property **operationResult** as **Integer**
    0 = OK
    1 = Permanent error (Do not try again)
    2 = Temporary error, try again soon (5 min)
    3 = Temporary error, try again later (1 hour)
    4 = Temporary error, try again tomorrow (1 day)

Shared fields (used by many methods)

- Property **Login** as **integer**
  Credentials
- Property **Password** as **string**
  Credentials
- Property **Timestamp** as **long**
  If smaller timestamp is received on same object, discard call.

Methods

Methods used by Pckasse*: sendArticle, sendImage, sendImageColor, sendArticleGroup, sendManufacturer, sendSize, sendColor, updateStockCount, removeAricle, getOrders, updateOrderStatus, updatePackageInfo, creditOrder, sendProductLine, getAllPaymentTypes, getCreditApplicants, GetStatus, getWelcomeMailTemplate, sendCustomerInfo, sendDiscount, getReceiptURL, getArticleURL, getOrderInfoURL*

- **createWebshop**(*webcompany* As **webCompany**) As **createDeltasWebshopResponse**
  This method is called when the create webshop button is clicked in Pckasse. If implemented the webshop should create a new webshop and return login info.
  If a webshop already exists resend login info to the registered admin e-mail for this webshop, if the login. If this method is not implemented then return information about how to create a webshop in the humanErrorMessage in the insertUpdateResponse object and set operationResult to 1 to indicate an error.

  - Class **webCompany**
    - Property **deliveryAddressLine1** as **String**
    - Property **deliveryAddressLine2** as **String**
    - Property **deliveryCity** as **String**
    - Property **deliveryZipCode** as **String**

- Property **demo** as **Boolean**
  Not used from within Pckasse.
- Property **email** as **String**
  The company's email address
- Property **emailWebshopInfo** as **String**
  Email to the webshop administrator, to use for important info, error messages, login info etc.
- Property **fax** as **String**
- Property **invoiceAddressLine1** as **String**
- Property **invoiceAddressLine2** as **String**
- Property **invoiceCity** as **String**
- Property **invoiceZipCode** as **String**
- Property **legalName** as **String**
- Property **licenseNo** as **String**
- Property **name** as **String**
- Property **orgNo** as **String**
- Property **password** as **String**
  If webshop exists and password is correct, then update info
- Property **phone** as **String**
- Property **postAddressLine1** as **String**
- Property **postAddressLine2** as **String**
- Property **postCity** as **String**
- Property **postZipCode** as **String**

➢ Class **createDeltasWebshopResponse**

- Property **adminUserName** as **String**
  The username of the admin user used to login to web administration
  Not used by Pckasse. Often sent in e-mail to webshop admin
- Property **adminUserPassword** as **String**
  The password og the admin user used on web
- Property **deltasoftId** as **Integer**
  The id of the created webshop
- Property **insertUpdate** as **insertUpdateResponse**
- Property **password** as **String**
  The password of the created webshop

❖ **creditOrder**(*login* As **Integer**, *password* As **String**, *orderId* As **Integer**, *orderLine* As **orderLineUpdate()**, *amount* As **Decimal**, *reason* As **String**) As **updateOrderResponse**
Used to credit items allready delivered.

➢ Parameters

- **amount** as **Decimal**
  Additional amount to credit (other than specified in orderLineUpdates)
  Total cannot exceed reserved amount.
- **orderId** as **Integer**
  OrderID in the webshop

- Property **reason** as **String**
  Message to customer

- Class **orderLineUpdate**
  - Property **amount** as **Integer**
    Number of items credited
  - Property **orderLineId** as **Integer**
    Webshop orderlineId (-10 is freight, -11 is extracost)
  - Property **Info** as **string**
    Info abount serialnumbers (Serienr: n[,n])

- Class **updateOrderResponse**
  - Property **amount** as **Decimal**
    Total amount credited (including extraCost and freightCost)
  - Property **authorzationId** as **String**
    Payment AuthorizationID from card issuer. Used as reference for accounting
  - Property **extraCost** as **Decimal**
    Part of amount that is extraCost
  - Property **freightCost** as **Decimal**
    Part of amount that is freightCost
  - Property **insertUpdate** as **insertUpdateResponse**
  - Property **paymentMethod** as **String**
    Name of paymentmethod. Ex VISA, Mastercard

- **getAllPaymentTypes**(*login* As **Integer**, *password* As **String**) As **getPaymentTypesResponse**
  Returns all possible paymenttypes from the webshop. It is used to specify an account number when sending journal to accounting. ***Since paymenttype is sent with each updateOrderResponse this function can be skipped by sending an empty array***.

  - Class **getPaymentTypesResponse**
    - Property **insertUpdate** as **insertUpdateResponse**
    - Property **payments** as **paymentType()**

  - Class **paymentType**
    - Property **name** as **String**
    - Property **paymentId** as **Integer**

- **getOrders**(*login* As **Integer**, *password* As **String**, *computerName* As **String**) As **webOrdersReturn**
  Return a list of new orders, from the webshop
  DO NOT send order twice! (Pckasse has duplicat check, but try not to send twice)
  If order is received OK Pckasse will call UpdateOrderStatus with OrderStatusId=4
  NB! Old versions of Pckasse will not send OrderStatusId=4, check computerName.
  When old versions requests orders you should flag them as successfully sent and not wait for a status of 4. If it fails Pckasse will send 7,8 or 10
  If an error occurs it will return UpdateOrderStatus with OrderStatusId=7, 8 or 10 and a

message with info why it failed.
The webshop system should send an email to the person responsible for the webshop if OrderStatusId=7 or to the customer if OrderStatusId=8
If the order has allready been recveived the order is skippet silently and OrderStatusId=4 is sent.
If a credit order (paymenttype=3) has a contactId that is not found in Pckasse it will send UpdateOrderStatus with OrderStatusId=10 (Resend Credit Applicant)
See UpdateOrderStatus for more info.
If paymentMethod=2 (COD) and storePickup is true the order is saved as a "parked" order in Pckasse, so that it can be picked up at the cash register and finished it there. For example takeout orders, where customer pays at pickup. No further communication with webshop about deliveries etc is sent.
If paymentMethod=3 (Credit), the order is saved as a normal credit order and no furhter communication to webshop about deliveries etc is sent.
If freightCost and/or extraCost is specified, special ordrelines will be added to the order (Flagged with webshopOrderLineId -10 / -11)

- Parameters
  - Property **computerName** as **String**
    <computername>\<username>{orderversion:2}
    Old versions of Pckasse does not have the {orderversion:2} suffix

- Class **webOrdersReturn**
  - Property **insertUpdate** as **insertUpdateResponse**
  - Property **listWebOrders** as **order()**

- Class **order**
  - Property **contactAddressline1** as **String**
    Used as customer address in Pckasse if create customer option is enabled.
  - Property **contactAddressline2** as **String**
    Used as customer address2 in Pckasse if create customer option is enabled.
  - Property **contactId** as **Integer**
    Required if credit order
    Used as customer externalId in Pckasse if create customer option is enabled.
  - Property **contactName** as **String**
    Used as customer name in Pckasse if create customer option is enabled.
  - Property **contactPostCity** as **String**
    Used as customer city in Pckasse if create customer option is enabled.
  - Property **contactPostNo** as **String**
    Used as customer zipcode in Pckasse if create customer option is enabled.
  - Property **deliveryAddressLine1** as **String**
    Used as Pckasse order delivery address
  - Property **deliveryAddressLine2** as **String**
    Used as Pckasse order delivery address

- Property **deliveryName** as **String**
  Used as Pckasse order delivery address
- Property **deliveryPostCity** as **String**
  Used as Pckasse order delivery address
- Property **deliveryPostNo** as **String**
  Used as Pckasse order delivery address
- Property **deliveryPhone** as **string**
  Used as Pckasse order delivery phone.
  If this is NULL or missing then Phone is used as order delivery phone in Pckasse
- Property **deliveryEmail** as **string**
  NOT USED YET
- Property **deltaOrderId** as **Integer**
  Webshop orderid. Must be supplied. Pckasse uses this when orderlines is delivered etc.
- Property **email** as **string**
  Used as customer email in Pckasse if create customer option is enabled.
- Property **extraCost** as **Decimal**
  Possible extra cost, ex fees or other costs
- Property **extraCostDescription** as **String**
  If specified used as orderline description overriding articlename from Pckasse
- Property **freightCost** as **Decimal**
  Total delivery cost including possible COD amounts, credit card fees ex
- Property **freightCostDescription** as **String**
  If specified used as orderline description overriding articlename from Pckasse
- Property **message** as **String**
  Message from customer placing the order.
- Property **orderLines** as **orderLine()**
- Property **paymentMethod** as **Integer**
  2 = COD
  3 = Credit Order
  All other values means its prepaid
- Property **phone** as **String**
  Used as Pckasse order delivery phone
- Property **storePickup** as **boolean**
  If true, specifies that order is to be picked up at the store.
- Property **taxExempt** as **Boolean**
  If true, Pckasse flags order as vat free.
- Property **WantedDeliveryTime** as **Date**
  Wanted delivery date
- Property **alternativeTax** as **Boolean**
  If true, Pckasse uses alternativeTax on all articles that has this option.
  For example food that has 25% default vat and 15% alternative vat
- Property **deliveredItemsAndCapturedPaymentInfo** as **deliveredItemsAndCapturedPaymentInfo()**

If set, Pckasse will assume that the items are allready processed and flag them as delivered and forward the sale to accounting and statistics.

- Class **orderLine**
  - Property **articleId** as **Integer**
    Pckasse ArticleId
  - Property **count** as **Integer**
    Number of items on this row
  - Property **discount** as **Decimal**
    Used as discountpercentage when paymentMethod<>3
    25 = 25%
  - Property **info** as **String**
    Extra info about the orderline
  - Property **orderLineId** as **Integer**
    Webshop OrderLineId. Must be supplied. Used when communication with webshop at a later time.
  - Property **price** as **Decimal**
    Used as price when paymentMethod<>3
  - Property **sizeColorId** as **Integer**
  - Property **warehouseId** as **Integer**
    If deliveredItemsAndCapturedPaymentInfo is set (Pckasse does not handle the delivery of the articles) this is used as the warehouse id to reduce the stock from.

- Class **deliveredItemsAndCapturedPaymentInfo**
  - Property **amount** as **decimal**
    Total amount captured from customer ***including extra cost and freight cost***
  - Property **authorzationId** as **string**
    CardIssuer Authorization Id
  - Property **extraCost** as **decimal**
    Part of Amount that is extracost
    Special ordreline for extraCost is updated if it exists, if not this value is ignored and there will be a difference between debet and credit in the accounting journal.
  - Property **freightCost** as **decimal**
    Part of Amount that is freight
    Special ordreline for freightCost is updated if it exists, if not this value is ignored and there will be a difference between debet and credit in the accounting journal.
  - Property **paymentMethod** as **string**
    Name og payment method, ex Visa, Master.
    It is created in Pckasse if it does not exist.

- **removeAricle**(*login* As **Integer**, *password* As **String**, *articleid* As **Integer**) As **insertUpdateResponse**
  (Yes. It's correct and misspelled, will be replaced in the future so make two declarations one for removeAricle and one for removeArticle)

Sent to the webshop if an article is deleted from Pckasse. The webshop should delete article if possible or set visible on web to false.

- Parameters
  - Property **articleid** as **Integer**
    ArticleID in Pckasse

- **sendArticle**(*login* As **Integer**, *password* As **String**, *article* As **article**) As **insertUpdateResponse**
  Sends all info about an article when it is created or changed.
  Only sends info if it is flagged with "Visible on web", or "Visible on web" is changed
  SendImage or SendImageColor is called if an article has an image or the image is changed

  - Parameters
    - Property **article** as **article**
      ArticleID in Pckasse

  - Class **article**
    - Property **alternativePrice** as **decimal**
      This is field is inc. vat.
    - Property **alternativePrice2** as **decimal**
      This is field is inc. vat.
      If this is set, then this value should be used as price when alternativeVat is defined and the order is takeaway (alternativeTax), so that the price is nice and round instead of a calculated uneven price.
      NB! Not to be used if alternativeVat is null
    - Property **articleGroup** as **articleGroup**
      If articleGroup is null then do not update.
      If ArticleGroup Id = 0, then remove it from the article
      If ArticleGroup does not exist, create it
    - Property **articleGroup2** as **articleGroup**
      See articleGroup info above
    - Property **articleGroup3** as **articleGroup**
      See articleGroup info above
    - Property **articleId** as **Integer**
      Pckasse articleID
    - Property **articleNo** as **String**
    - Property **articleStatus** as **Integer**
      0-Active, 1-Passive, 2-Expired, 3- Blocked
    - Property **articleWebAction** as **Integer**
      0-Normal, 1- ContactToPurchase, 2-ContactForPrice, 3-PromotionalItem
    - Property **confirmedDelivery** as **Boolean**
      True if ExpectedDeliveryDate is confirmed
    - Property **costPrice** as **decimal**
    - Property **description** as **String**
    - Property **discount** as **Decimal**
      Offer price

- Property **discountFrom** as **Long**
  Offer price valid from (number of milliseconds from 1/1/1970 00:00)
- Property **discountTo** as **Long**
  Offer price valid to (number of milliseconds from 1/1/1970 00:00)
- Property **eans** as **String()**
- Property **expectedDeliveryAmount** as **Integer**
  Number of items to be delivered from supplier.
- Property **expectedDeliveryDate** as **DateTime**
  If null then unknown deliverydate.
  See ConfirmedDeliveryDate
- Property **externalGroupId** as **Integer**
  Pckasse ArticleGroupID used by SendDiscount.categoryID
- Property **externalGroupId2** as **Integer**
  Pckasse ArticleGroup2ID used by SendDiscount.category2ID
- Property **externalLink** as **String**
- Property **height** as **decimal**
- Property **hideWhenOutOfStock** as **Boolean**
  If it is true hide the article from web if it is not in stock.
  Ex season articles.
- Property **length** as **decimal**
- Property **manufacturer** as **manufacturer**
- Property **manufacturerArticleNo** as **String**
- Property **name** as **String**
- Property **noDiscount** as **boolean**
  Order discount field should not be used.
  This does not affect offer price.
  (offer price is not a discount but a different price)
- Property **nonStockItem** as **boolean**
  This article in external stock (Use a custom text if zero stock)
- Property **nonStockItemDays** as **integer**
  Days to get the article from external stock
- Property **price1** as **decimal**
- Property **price2** as **decimal**
- Property **price3** as **decimal**
- Property **price4** as **decimal**
- Property **price5** as **decimal**
- Property **price6** as **decimal**
- Property **price7** as **decimal**
- Property **price8** as **decimal**
- Property **price9** as **decimal**
- Property **price10** as **decimal**
- Property **productLine** as **productLine**
- Property **purchasePrice** as **decimal**
- Property **recommendedProduct** as **Boolean**
  Suggested use: Show article as a recommended product. Ex at start page
- Property **salesPrice** as **Decimal**
  Price in the webshop inc. vat

- Property **shippingType** as **integer**
  0 – Not specified
  3 – Cannot be shipped
  5 – Freight per article
- Property **sizeColorInUse** as **Boolean**
- Property **sizeColors** as **sizeColor()**
- Property **stockCount** as **Integer**
  Quantity in stock
- Property **stockDetails** as **stockDetail()**
  Quantity in stock specified by warehouse
- Property **storePrice** as **Decimal**
  Price in the physical shop inc. vat
- Property **subtitle** as **String**
- Property **suggestedPrice** as **Decimal**
- Property **timestamp** as **Long**
- Property **vat** as **Decimal**
  Vat percentage (25, 15, 0 etc)
- Property **visibleOnWeb** as **Boolean**
- Property **volume** as **Decimal**
- Property **webshippingPrice** as **Decimal**
- Property **webstockLimit** as **Integer**
  Suggested use: Subtract this from stockCcount before displaying availiable stock in the webshop
- Property **weight** as **Decimal**
- Property **width** as **Decimal**
- Property **alternativeVat** as **Decimal**
  Set if this article has alternative vat defined otherwise null.
  Ex food that has 25% default vat and 15% alternative vat.
  If Order is sent with flag "alternativeTax" this rate is used instead of the default
- Property **info1** as **String**
- Property **info2** as **String**
- Property **info3** as **String**

➢ Class **articleGroup**
- Property **articleGroupId** as **Integer**
  Pckasse ArticleGroupID
  Can be the same for different groupNumbers.
  groupNumber and articleGroupId together forms the unique key.
- Property **description** as **String**
  Suggested use: Show as a heading when selecting group
- Property **groupNumber** as **Integer**
  Articlegroup level 1,2 or 3
- Property **name** as **String**
- Property **timestamp** as **Long**

➢ Class **manufacturer**

- Property **manufacturerId** as **Integer**
  Pckasse Id
- Property **name** as **String**
- Property **timestamp** as **Long**

- Class **productLine**
  - Property **id** as **Integer**
    Pckasse Id
  - Property **name** as **String**
  - Property **number** as **Integer**

- Class **sizeColor**
  - Property **color** as **color**
  - Property **confirmedDelivery** as **Boolean**
    True if ExpectedDeliveryDate is confirmed
  - Property **eans** as **String()**
  - Property **expectedDeliveryAmount** as **Integer**
    Number of items to be delivered from supplier.
  - Property **expectedDeliveryDate** as **Date**
    If null then deliverydate is unknown
  - Property **info** as **String**
    Info about the size. Ex US 9 equals EU 40
  - Property **size** as **size**
  - Property **sizeColorId** as **Integer**
    Pckasse Size/Color combo Id
  - Property **sizeColorInUse** as **Boolean**
    False if discontinued
  - Property **stockCount** as **Integer**
    Availiable Stock with that size/color combo
  - Property **stockDetails** as **stockDetail()**
    Count in stock specified by warehouse
  - Property **timestamp** as **Long**

- Class **size**
  - Property **name** as **String**
  - Property **sizeId** as **Integer**
    Pckasse Id
  - Property **timestamp** as **Long**

- Class **color**
  - Property **code** as **String**
  - Property **colorId** as **Integer**
    Pckasse Id
  - Property **name** as **String**
  - Property **timestamp** as **Long**

- Class **stockDetail**
  - Property **warehouseId** as **Integer**

        - Property **count** as **Integer**
          Count in stock

- **sendArticleGroup**(*login* As **Integer**, *password* As **String**, *articleGroup* As **articleGroup**) As **insertUpdateResponse**
  Called when an article group is created or changed

    - Class **articleGroup**
      Specifed under sendArticle

- **sendColor**(*login* As **Integer**, *password* As **String**, *color* As **color**) As **insertUpdateResponse**
  Called when a color is created or changed in Pckasse

    - Class **color**
      Specifed under sendArticle

- **sendImage**(*login* As **Integer**, *password* As **String**, *image* As **Byte()**, *articleid* As **Integer**) As **insertUpdateResponse**
  Send image to webshop. If ArticleId=-10 then it is the company Logo

    - Properties
        - Property **articleid** as **Integer**
          Pckasse ArticleId (-10 is company logo)
        - Property **image** as **Byte()**
          Must support most common types (jpeg, png, gif)
          If empty byte array, delete from article

- **sendImageColor**(*login* As **Integer**, *password* As **String**, *image* As **Byte()**, *articleid* As **Integer**, *colorid* As **Integer**, *imageid* As **Integer**) As **insertUpdateResponse**
  Sends an image for a specified color of the article. Multiple images pr color is possible.

    - Parameters
        - Property **articleid** as **Integer**
          Articleid in Pckasse
        - Property **colorid** as **Integer**
          ColorId in Pckasse
        - Property **image** as **Byte()**
          If it is an empty byte array then delete the image from article/color
        - Property **imageid** as **Integer**
          ImageId in Pckasse

- **sendManufacturer**(*login* As **Integer**, *password* As **String**, *manufacturer* As **manufacturer**) As **insertUpdateResponse**
  Called when a manufacturer is created or changed in Pckasse

    - Class **manufacturer**
      Specifed under sendArticle

- **sendProductLine**(*login* As **Integer**, *password* As **String**, *size* As **productLine**) As **insertUpdateResponse**
  Called when a product line is created or changed in Pckasse

  - Class **productLine**
    Specifed under sendArticle

- **sendSize**(*login* As **Integer**, *password* As **String**, *size* As **size**) As **insertUpdateResponse**
  Called when a size is created or changed in Pckasse

  - Class **size**
    Specifed under sendArticle

- **updateOrderStatus**(*login* As **Integer**, *password* As **String**, *updateOrder* As **updateOrder**) As **updateOrderResponse**
  Called when an order is received, items delivered, or an order has an error.
  Delivery statuses:
  OrderStatusId=3 - Fully delivered. Undelivered items should be cancelled (if any)
  OrderStatusId=5 – Normal delivery (Do not cancel rest)
  Receive order statuses:
  OrderStatusId=4 – Order is received successfully
  OrderStatusId=7 - Order failed - Send message to webshop admin.
  OrderStatusId=8 - Order failed - Send message to customer.
  OrderStatusId=10 - Order failed because of missing customer (only credit). Flag the customer for resending. Credit applicants are delivered to Pckasse with GetCreditApplicants.
  For 4,7,8 and 10 statuses only deltaOrderId, message and orderStatusId is sent

  - Class **updateOrder**
    - Property **deltaOrderId** as **Integer**
      Webshop OrderId
    - Property **message** as **String**
      Message to customer
    - Property **orderLines** as **orderLineUpdate()**
    - Property **orderStatusId** as **Integer**
      3 – Fully delivered (Cancel rest), 5 – normal delivery, 7 – Failed (message to admin), 8 – Failed (message to customer), 10 - ResendCreditApplicant
    - Property **packageNo** as **String**
      Transporter packageNo. Used for tracking
      If not supplied, it may be transferred later using UpdatePackageInfo, based on SendId
    - Property **packtrackURL** as **String**
    - Property **sendId** as **Integer**
      Pckasse deliveryId. UpdatePackageInfo may use this at a later time.
    - Property **timestamp** as **Long**
    - Property **transporterName** as **String**

- Class **orderLineUpdate**
  - Property **amount** as **Integer**
    Number of items delivered
  - Property **info** as **String**
    Used for serianlnumbers
  - Property **orderLineId** as **Integer**

- Class **updateOrderResponse**
  - Property **amount** as **Decimal**
    *Total amount captured from the customer for this delivery **including extra cost and freight cost if applicable***
    If this is a part delivery, the amount should only reflect delivered lines..
    Eks. Order of 2 golf balls (100,- a piece) with freight (99,-).
    1. Delivery: one ball, full freight (199,-)
    2. Delivery: one ball (100,-)
    Optionally splitting freight:
    1. Delivery: one ball, half freight rounded (150,-)
    2. Delivery: one ball, remaining freight (149,-)
  - Property **authorzationId** as **String**
    CardIssuer Authorization Id
  - Property **extraCost** as **Decimal**
    Part of Amount that is extracost
    Special ordreline for extraCost is updated if it exists, if not this value is ignored and there will be a difference between debet and credit in the accounting journal.
  - Property **freightCost** as **Decimal**
    Part of Amount that is freight
    Special ordreline for freightCost is updated if it exists, if not this value is ignored and there will be a difference between debet and credit in the accounting journal.
  - Property **insertUpdate** as **insertUpdateResponse**
  - Property **paymentMethod** as **String**
    Name og payment method, ex Visa, Master.
    It is created in Pckasse if it does not exist.

- **updatePackageInfo**(*login* As **Integer**, *password* As **String**, *packageNo* As **String**, *transporterName* As **String**, *packtrackURL* As **String**, *message* As **String**, *sentid* As **Integer**) As **insertUpdateResponse**
  Updates packackeno on a shipment if not send by UpdateOrderStatus

  - Parameters
    - Property **message** as **String**
      Message to Customer
    - Property **sentid** as **Integer**
      Same SentId used by UpdateOrderStatus.sendId

- **updateStockCount**(*login* As **Integer**, *password* As **String**, *updateStock* As **updateStock**) As **insertUpdateResponse**

Called when article stock amount changes. Ex sales, purchase, inventory
If a SizeColor is defined on an article, this function is called multiple times.
One for the total on the article, with sizecolor not specified, and one for each sizecolor.
If operationResult< >0, remaining sizecolors will not be sent.

- Class **updateStock**
    - Property **articleId** as **Integer**
      Pckasse ArticleId
    - Property **confirmedDelivery** as **Boolean**
    - Property **count** as **Integer**
      Number of items in stock
    - Property **expectedDeliveryAmount** as **Integer**
    - Property **expectedDeliveryDate** as **Date**
    - Property **sizeColorId** as **Integer**
    - Property **stockDetails** as **stockDetail()**
      Number of items in stock specified by warehouse
    - Property **timestamp** as **Long**

- **getCreditApplicants**(*login* As **Integer**, *password* As **String**) As **creditApplicantsReturn**
  Return a list of customerers who have applied for credit.
  Pckasse will return approved or not with SendCustomer.
  SendCustomer contains all discounts for the customer.

    - Class **creditApplicantsReturn**
        - Property **insertUpdate** as **insertUpdateResponse**
        - Property **listCustomers** as **customer**()

    - Class **customer**
        - Property **contactId** as **Integer**
          Webshop contact Id
        - Property **email** as **String**
        - Property **fax** as **String**
        - Property **invoiceAddress** as **String**
        - Property **invoiceAddress2** as **String**
        - Property **invoiceOnEmail** as **Boolean**
        - Property **invoicePostCity** as **String**
        - Property **invoicePostNo** as **String**
        - Property **name** as **String**
        - Property **orgNo** as **String**
        - Property **phone1** as **String**

- **sendCustomerInfo**(*login* As **Integer**, *password* As **String**, *customerInfo* As **customerInfo**) As **insertUpdateResponse**
  Used when if customer is approved for credit when received by GetCreditApplicants or when Pckasse sends a new customer (deltaCustomerId is then 0)
  An invitation email should be sent with login and password to the customer if the welcomemessage field is set. Rember to add header (welcome customerName) and

footer (You Login is.....)
Remember to send the same id back in insertUpdateResponse.DeltaId
or the newly created id if deltaCustomerId was 0.

- Class **customerInfo**
  - Property **address1** as **string**
  - Property **address2** as **string**
  - Property **countryCode** as **Integer**
    ISO 3166-1
    http://no.wikipedia.org/wiki/ISO_3166-1
  - Property **creditApproved** as **Boolean**
    True if approved for credit.
    Suggested use: Inform the customer about result.
  - Property **customerGroup** as **customerGroup**
    Customers group in Pckasse
  - Property **deltaCustomerId** as **Integer**
    External Customer Id (Same as GetCreditApplicants.ContactId)
    If not set, this is a new customer and it should receive welcome email containing the customerInfo.welcomeMessage
  - Property **email** as **string**
  - Property **listDiscounts** as **discount**()
    All discounts belonging to the customer.
  - Property **name** as **string**
  - Property **orgNo** as **string**
  - Property **pckCustomerId** as **Integer**
    Pckasse internal Customer Id
  - Property **phoneNo** as **string**
  - Property **postCity** as **string**
  - Property **postNo** as **string**
  - Property **welcomeMessage** as **string**
    Message that should be sent to customer together with header and footer

- Class **customerGroup**
  - Property **customerGroupId** as **Integer**
  - Property **name** as **String**

- Class **discount**
  - See sendDiscount for info.

- **sendDiscount**(*login* As **Integer**, *password* As **String**, *discount* As **discount**) As **insertUpdateResponse**
  Updates discountline if changed or new

  The most specified discountrow should be applied.
  Update discount lines for "guest" from time to time or on a new session
  Fetch discount lines for a customer when he/she logges in and store in session.
  For use in SQL:

```
select CategoryId, Category2Id, ManufacturerId, ArticleID, Count, ValidUntil, PriceType,
PriceAdjustment, Discount from CustomerDiscounts where (ValidUntil is null or
ValidUntil>@DateTimeNow) and (CustomerGroupID is null or
CustomerGroupID=@CustomerGroupID) and (CustomerID is null or CustomerID=@CustomerID)
order by ArticleID desc, Category2Id desc, CategoryId desc, ManufacturerId desc, CustomerID desc,
CustomerGroupID desc, PriceType desc, Discount desc
```

To show price / add to cart

```
Loop Discounts
{
  If Discount.CategoryId is NULL or Article.ExternalGroupId =Discount.CategoryId
    If Discount.Category2Id is NULL or Article.ExternalGroupId2 =Discount.Category2Id
      If Discount.ManufacturerId is NULL or Article.ManufacturerId =Discount.ManufacturerId
        If Discount.ArticleId is NULL or Article.Id=Discount.ArticleId
          If OrderLine.Count >= Discount.Count
            If Discount.ValidUntil is NULL or Discount.ValidUntil >= Now
              Apply price and/or discount
              Exit loop
}
```

Apply price and/or discount logic:

```
If Discount.PriceType > 0 (Discount Only)
{
  If 1 (Netprice) and Article.NetPrice <> 0 then set OrderLine.NetPrice to Article.NetPrice * (1 +
Discount.PriceAdjustment / 100)
  If 2 (CostPrice) and Article.CostPrice <> 0 then set Orderline.NetPrice to Article.CostPrice * (1 +
Discount.PriceAdjustment / 100)
  …
  If 17 (Price10) and Article.Price10<> 0 then set Orderline.NetPrice to Article.Price10 * (1 +
Discount.PriceAdjustment / 100)
}
If Article.NoDiscount then set OrderLine.Discount to 0 (to be sure) else set OrderLine.Discount to
Discount.Discount
If Article.DiscountFrom/To is valid and Article.Discount (offerprice) is less than
Orderline.DiscountetPrice (Price minus discount) then use OfferPrice as Orderline.Price and set
OrderLine.Discount to 0
```

NB! PriceType 7 (AveragePurchasePrice) is not possible to use in the webshop. This is a "floating" price in Pckasse and is not recommended to use in customerDiscounts.

NB! In Pckasse there is a limit to the offerprice based on days in the week, but this is not sent to the webshop. If Customer thinks something is wrong, remember to check that.

- Class **Discount**
  - Property **articleId** as **Integer**
    Discount on a specified article
  - Property **category2Id** as **Integer**
    Discount on all articles in this category2
  - Property **categoryId** as **Integer**
    Discount on all articles in this category

        - Property **count** as **Integer**
          Discount only if orderline.count is equal or higher
        - Property **customerGroupId** as **Integer**
          Discount on entire group of customers
        - Property **customerId** as **Integer**
          Discount for specified customer
        - Property **deleteDiscount** as **Boolean**
          Remove discount row if true
        - Property **discount1** as **Decimal**
          Discount in percentage
        - Property **discountId** as **Integer**
          Pckasse internal Id
        - Property **manufacturerId** as **Integer**
          Discount for this manufacturer only
        - Property **priceAdjustment** as **Decimal**
          Add this percentage to price specified in priceType and set as orderlines Price if priceType>0
        - Property **priceType** as **Integer**
          0 - Discount Only
          1 – Netprice
          2 – Costprice
          3 – AlternativePrice2
          4 – PurchasePrice
          5 – SuggestedPrice
          6 – AlternativePrice
          7 – AveragePurchasePrice
          8 – Price1
          9 – Price2
          10- Price3
          11 – Price4
          12 – Price5
          13 – Price6
          14 – Price7
          15 – Price8
          16 – Price9
          17 – Price10
        - Property **validUntil** as **DateTime**
          Discount expires on this date

- **getStatus**(*login* As **Integer**, *password* As **String**) As **status**
  Gets current status of the webshop. Used only for info to the user.

    - Class **status**
        - Property **creditApplicants** as **Integer**
          Number of applicants waiting to be fetched
        - Property **message** as **String**
          Shown if operationResult <> 0

    - Property **onlineCustomers** as **Integer**
      Number of visitors on the web at the current time
    - Property **operationResult** as **Integer**
    - Property **orders** as **Integer**
      Number of orders not yet fetched by Pckasse.

- **getWelcomeMailTemplate**(*login* As **Integer**, *password* As **String**) As **mailTemplate**
  Used before a customer is sent to the webshop to setup welcome message.

  - Class **mailTemplate**
    - Property **footer** as **string**
      Cannot be changed in Pckasse
      Used to display the full message
    - Property **header** as **string**
      Cannot be changed in Pckasse
      Used to display the full message
    - Property **mesage** as **string** (Yes... misspelled)
      Can be edited and is returned to the webshop in SendCustomerInfo.welcomeMessage

- **getReceiptURL**(*login* As **Integer**, *password* As **String**, *orderid* As **Integer**) As **String**
  Used to get the url for the receipt for an order.
  Pckasse will tell Windows to "open" this url, so remember to prefix with http://
  If more than one receipt exists, because of multiple deliveries, a list should be displayed so the customer can choose.

  - Parameters
    - Property **orderid** as **integer**
      The orderid in the webshop to get receipt for

  - Output parameter
    - Customer is forwarded to this string.

- **getArticleURL**(*login* As **Integer**, *password* As **String**, *pckid* As **Integer**) As **String**
  Used to get the url to show article info.
  Pckasse will tell Windows to "open" this url, so remember to prefix with http://
  A page will typically display article name, image, stock etc.

  - Parameters
    - Property **pckid** as **integer**
      The articleid in Pckasse to return info for

  - Output parameter
    - Customer is forwarded to this string.

- **getOrderInfoURL**(*login* As **Integer**, *password* As **String**, *orderid* As **Integer**) As **String**
  Used to get the url to show order info.
  Pckasse will tell Windows to "open" this url, so remember to prefix with http://
  A page will typically display info about the order, much as the end customer sees when logging on the "My page"

  - Parameters
    - Property **orderid** as **integer**
      The orderid in webshop to return info for

  - Output parameter
    - Customer is forwarded to this string.