cairosoft

CAIRO-soft sp. z o.o.
NIP 739-360-94-27
ul. Lubelska 36
10-408 Olsztyn
tel. 089 533 95 07
cairo@cairo.pl www.cairo.pl

KRS 0000275085, kapitał: 100 000 zł, Sąd Rejonowy w Olsztynie, VIII Wydział Gospodarczy

FALCON5 WEBSERVICE DOCUMENTATION v 1.21 (EN)

TABLE OF CONTENTS

CHANGELOG

1.21 *(25.04.2023)*	• In method getProductsInfo added request parameters: gtuCode (#34527) • In method doOrderProducts removed request parameters: routeInfo (#38317) • In method getMyReturns added response parameters: packageBarcode (#40153) • In method getMyBonuses added response parameters: type_bonus, is_ranking and rankingPromoList (#48802) • In method getReturnNowProductInfo added response parameters: returnSectorInfo (#47453) • In method doLogmCreatePackage added request parameters : x, y, z, weight and response parameters: courierLabelList (#44972) • In method doLogmCreatePackage added request parameters : collectivePackageInfo (#47020) and response parameters : dropshippingLabelList (#45984) • New method getTree (#33833)
1.20 *(15.09.2021)*	• In method getMyOrders added response parameters: priceCatExclTax, priceCatInclTax. In method doOrderItemEdit added response parameters: priceCatExclTax, priceCatInclTax. In method getMyInvoices added response parameters: priceCatExclTax, priceCatInclTax (#10665) • In method getMyPackages added request parameters: getTrackingUrl (#10553) • In method getMyCustomerInfo added response parameters: deliveryConfirmationDocTypes (#10499) • In method getMyRoutes response parameter isDefault was marked as „deprecated" and is always returning value "false" (#10477) • In method getMyReturns added response parameters: documentNumber (#10441) • In method getMyOrders added response parameters: isFromDeficiencies (#10437) • In method getMyOrders added response parameters: isExternalOrderInProgress (#10400) • In method doOrderProducts added request parameters: customerDestinationWarehouseId, isCustomerSupply (#10378) • In method getMyInvoices the range of values of request parameter closeStatus was changed (#10339) • In method getMyInvoices added request parameters: productIdList (#10128) • In method doOrderProducts added request parameters: superSupplier (#10000) • In method getProductsInfo added response parameters: motonet. In method doOrderProducts added response parameters: motonet (#9906) • In method getMyOrders added response parameters: motonet (for items). In method getProductsInfo added request parameters: motonet. In method doOrderProducts added request parameters: motonet. In method doBuyNow added request parameters: motonet. In method getReturnNowProductInfo added request parameters: motonet. In method doReturnNow added request parameters: motonet (#9896) • In method getMyCustomerInfo added response parameters: isErpSaleLocked (#9890) • In method doCreateReturn added request parameters: srcDocumentId. Added response errors: ERR_WRONG_SOURCE_DOCUMENT, ERR_WRONG_SOURCE_DOCUMENT_WAREHOUSE. In method getReturnQuantityAvailable added request parameters: detailedQuantity. Added response parameters: documentList (#9834) • In method getProductsInfo removed response parameters: deliveryTimeMax (#9772) • In method doOrderClose added response errors: ERR_BLOCKED_CONFIRM_RECEIPT_DELIVERY (#9764) • In method getMyPackages added response parameters: urlTracking (#9758)

	• In method getMyCustomerInfo added response parameters: creditLimit (#9730) • In method getMyBonuses added request parameters: getGroupPromoList, getProductPromoList. Added response parameters: groupPromoList, productPromoList (#9704) • Added method getProductStockChanges (#9582) • In method doReturnNow added response parameters: isSplitPayment (#9570) • In method doOrderClose added request parameters: routeId. Added response errors: ERR_CIP_ROUTE, ERR_ROUTE_UNAVAILABLE, ERR_WRONG_ROUTE_ID (#9464) • In method getMyCustomerInfo added response parameters: pointsExpired (#9450) • In method getMyOrders added response parameters: routeId. In method getMyInvoices added response parameters: routeId (#9400) • In method doLogmCreatePackage added request parameters: hasDeposit, depositBuyPrice, depositIlkod. Added response errors: ERR_INVALID_DEPOSIT_BUY_PRICE (#8967) • In method getProductsInfo added response parameters: hasProductDeposit (#8934) • In method doLogmCreatePackage added response parameters: specificationPdfData (#8924) • Added method getReturnNowProductInfo (#8874) • In method doOrderItemEdit added response errors: ERR_ITEM_HAS_DEPOSIT (#8868) • In method doOrderProducts added request parameters: handoverToDD. Added response errors: ERR_INVALID_HANDOVERTODD (#8820) • In method getMyCustomerInfo added response parameters: isDD (#8758) • In method getMyCustomerInfo added response parameters: isInvoice, isReceipt (#8709) • Added method doLogmCreatePackage (#8685) • In method doOrderProducts added request parameters: packMergeKey, srcDocId, retShelfMerge (#8408) • In method getMyInvoices removed request parameters: webCatUserId. In method doOrderProducts removed request parameters: webCatUserId (#8167) • In method doOrderProducts added request parameters: onlyFoundItems (#8082) • In method getMyPackages added response parameters: isClosed (for package) and Ilkod (for package item) (#7587)
1.19 *(05.11.2019)*	• New method getProductFileUrl (#7145) • In method doOrderProducts added parameters routeInfo, dhlde_postfilialePostfilialNumber, dhlde_postfilialepostNumber, dhlde_postfilialeZip, dhlde_postfilialeCity, dhlde_preferredLocation, dhlde_preferredNeighbour, dhlde_recipientEmailAddress, dhlde_namedPersonOnly, dhlde_identCheckSurname, dhlde_identCheckGivenName, dhlde_identCheckDateOfBirth, dhlde_identCheckMinimumAge, dhlde_visualCheckOfAge, dhlde_preferredDay, dhlde_preferredTime, dhlde_noNeighbourDelivery, dhlde_bulkyGoods, dhlde_additionalInsurance i dhlde_noticeOfNonDeliverability (#6836) • In method getMyCustomerInfo added return fields saleBlockInfo, status, code and msg (#6720) • In method getMyOrders added parameter getLoyaltyInfo and return fields isReward and loyaltyRewardPoints. In method getMyInvoices added return fields loyaltyRewardPoints and isReward (#6690) • In method doOrderProducts added parameter externalItemId (#6673) • In method getMyOrders and getMyInvoices added parameter and return field isReward (#6577)

	• In method `doOrderProducts` added parameter `isReward` (#6572) • In method `doOrderItemsMove` added return field `destOrderId` (#6511) In method `getMyInvoices` added parameters `wmsStatus`, `getTrackingNrList` and added return fields `externalId` i `trackingNrList` (#6459) • In method `doCreateComplaint` added parameters and return fields `compensationValue` i `compensationCurrency` (#6418) • In method `getMyCustomerInfo` parameters `dateFrom`, `dateTo` are deprecated (#6151)
1.18 *(17.04.2019)*	• In method `getMyInvoices` added parameter `deliveryConfirmationStatus`, and added return field pole `isDeliveryConfirmed`. In method `getMyCustomerInfo` added return fields `isDeliveryConfirmationRequired`, `deliveryConfirmationPeriod`. New method `doSetDeliveryConfirmation` (#6148) • In method `doOrderProducts` added parameter `shipmentType` (#6024) • In method `doOrderProducts` dodano nowy rodzaj błędu `ERR_INVALID_CUSTOMER` (#5938) • In method `getMyInvoices` added return field `country` (#5925) • In method `getMyCustomerInfo` added return field `nextRouteDt` (#5813) • In method `getMyCustomerInfo` deleted a field `isColorGroupPromo` (#5804) • In method `getMyInvoices` added return fields : `weight_net`, `weight_gross`, `customs_code` (#5799) • New method `doSetRodoStatus` (#5600) • In method `getMyCustomerInfo` added return field `rodoStatus` (#5592) • In method `getMyInvoices` added return field and parameter `isDoReturnNow` (#5550) • New method `doReturnNow` (#5358) • In method `getProductsInfo` change extended functionality of field `isGroupPromo`, changed value of `ERR_PRODUCTLIST_OVERSIZED` (#5300) • In method `getMyCustomerInfo` added return field `isActive` (#5277) • In method `getMyBonuses` extended functionality of field `type` (#5233) • In method `doOrderProducts` changed error value `ERR_PRODUCTORDERLIST_OVERSIZED` (#5231) • In method `doCreateComplaint` added parameters: `powerValue`, `powerUnit`, `installedBy`, `uninstalledBy`, `faultFoundBy`, `faultWhen`, `refuseAddCosts`. (#5174)
1.17 *(19.09.2017)*	• In method `getMyCustomerInfo` added return field `isColorGroupPromo` (#5016) • In method `getMyBonuses` added return field `targetIncrease` (#4987) • In method `getMyInvoices` and `getMyOrders` added return field `justNumber` (#4907) • In method `getProductsInfo` added return field `isGroupPromo` (#4823) • In method `getMyBonuses` added struct `targetLevelList` (#4880) • In method `getMyBonuses` extended functionality of field `type` (#4804) • In method `getMyInvoices` added: parameter `getLoyaltyInfo` and return field `loyaltyPoints` (#4786) • In method `getMyCustomerInfo` added return struct `groupPromoList` (#4777)
1.16 *(04.04.2017)*	• In method `getMyInvoices` added return field `loyaltyPoints` (#4738) • In method `getMyCustomerInfo` added struct `loyaltyInfo` (#4734) • New method `doDocumentPrint` (#4687) • New method `doComplaintPrint` (#4672) • In method `doCreateComplaint` added new error code `ERR_CAR_DATA_INACTIVE` (#4669) • In method `getMyCustomerInfo` added return fields: `isComplaintEditCar`, `isComplaintEditDesc`, `isComplaintDescRequired` (#4664) • In method `getMyReturns` added return field `canEdit` (#4655)

	• In method getMyCustomerInfo added return field isReturnActive (#4646) • New method getReplacements (#4645)
1.15 *(14.02.2017)*	• In method getProductsInfo always return fields: quantityRealShow, quantityReal (#4620) • In method getInvoicesComplaintProduct added return field quantityAvailable (#4615) • In method getMyCustomerInfo added return field isComplaintActive (#4613) • In method getMyOrders added parameters: additionalInfo, dateAddInfo (#4581) • In method doOrderEdit added parameters: additionalInfo, dateDelivery and new error code ERR_ADDITIONAL_INFO_INACTIVE (#4580) • In method getProductsInfo added return fields: reference2, tecidd, tecnum, group2, isReplacement, isPhoto, producer (#4569) • In method getVersion added return fields: ownum, verFull (#4548) • In method getMyCustomerInfo added return fields: isGoodsIssueActive, isExternalStockAvailable, stockQuantityFormat, isEdocumentActive. In method getMyInvoices added return field isPrinted (#4540) • In method getMyCustomerInfo added return field warehouseList (#4529) • New method doBuyNow (#4525) • In method getMyInvoices added return fields: recipientId, recipientName, recipientShortName (#4521)
1.14 *(21.12.2016)*	• New method getMyBonuses (#4504) • In method getMyComplaints added return field decisionComment (#4469) • In method getMyCustomerInfo added return fields: isPromoActive, subTerminalId, showPaymentsOnStart, isBonusesActive, introText (#4464) • New method doReturnSendPackage (#4451) • New method doReturnGetPackageLabel and in method doReturnCreatePackage added new error codes: ERR_INVALID_RETURN_ID, ERR_INVALID_RETURN_STATE, ERR_INVALID_RETURN_STATUS, ERR_INTERNAL (#4441) • In method getMyInvoices added return fields: type, eDocumentStatus (#4435) • In method getMyComplaints added return fields: storeExt, currency, productionYear, brandVehicle, registerNumber, engineNumber, vinNumber, engineCapacity, installDate, counterStateInstall, workshopInstallName, workshopInstallAddress, uninstallDate, counterStateUninstall, workshopUninstallName, workshopUninstallAddress (#4417) • New method doCreateReturn (#4415) • In method getMyReturns in struct filterOptions added fields: dateFrom, dateTo and new error codes: ERR_INVALID_DATERANGE, ERR_INVALID_RETURN_ID, ERR_INVALID_RETURN_STATE, ERR_INVALID_RETURN_STATUS (#4413) • New method doReturnClearPackage (#4409) • New method doEditReturne (#4408) • New method getReturnQuantityAvailable (#4407) • New method doDeleteReturn (#4394) • New method doReturnCreatePackage (#4370) • In method doCreateComplaint added parameters: productionYear, brandVehicle (#4365) • New method getMyReturns (#4354) • In method getInvoicesComplaintProduct added struct itemList (#4306) • New method getInvoicesComplaintProduct (#4280) • In method doCreateComplaint added parameters: registerNumber,

	engineNumber, vinNumber, engineCapacity, installDate, counterStateInstall, workshopInstallName, workshopInstallAddress, uninstallDate, counterStateUninstall, workshopUninstallName, workshopUninstallAddress (#4238) • New method getComplaintQuantityAvailable (#4232) • In method getMyComplaints added return fields: dateClose, productionYear, brandVehicle (#4218)
1.13 *(22.07.2016)*	• In method getMyComplaints added return fields dateCreate, dateClose, decision, decisionText, decisionType (#4032) • In method getMyInvoices added struct pieceList (#4001) • In method doOrderProducts added struct pickupPointAddress (#3921)
1.12 *(10.03.2016)*	• In method doOrderProducts added parameter sendOnSaturday (#3913) • In method doOrderProducts added parameter deliveryOnSaturday and new error code ERR_COURIER_INFO_EDIT_INACTIVE (#3848) • In method getMyCustomerInfo added struct balance (#3833)
1.11 *(22.12.2015)*	• In method getMyPayments added parameters isRelDocument, relDocumentId, relDocumentType, payItemList (#3758) • In method getMyInvoices and getMyOrders added an option to filter the field externalId (#3731) • New method getMyPackages (#3728) • In method doOrderProducts added struct additionalDeliveryAddress containing the parameters: name, company, street, postcode, city, country, phone, email; and new error code ERR_ADDITIONAL_DELIVERY_ADDRESS_EMPTY_NAME_AND_COMPANY (#3683) • In method getMyInvoices added parameter srcDocId (#3665) • In method doLogin added new error code ERR_LOGIN_BLOCKED (#3663)
1.10 *(14.09.2015)*	• New method getMyComplaints (#3571) • New method doCreateComplaint (#3566) • In method getMyInvoices added parameters closeStatus and isClosed (#3560) • In method doOrderProducts added parameters packageProductId, dataExpire and new error codes (#3544): ○ ERR_PACKAGE_PRODUCT_INACTIVE ○ ERR_PACKAGE_PRODUCT_ID_INVALID
1.09 (17.08.2015)	• English version of documentation. • In method doOrderProducts added parameter routeId (#3483)
1.08 *(07.07.2015)*	• In method doOrderProducts added parameter payment and new error codes (#3415 and #3442): ○ ERR_PAYMENT_EDIT_NOT_AVAILABLE ○ ERR_PAYMENT_INVALID_TYPE ○ ERR_PAYMENT_TYPE_NOT_AVAILABLE
1.07 *(30.06.2015)*	• In method getMyInvoices added filter paymentStatus (#3402) • New method getMyCustomerInfo (#3402) • In method doOrderProducts added parameters: incotermsCode, additionalInfo and new error codes (#3374): ○ ERR_INCOTERMS_INACTIVE ○ ERR_INCOTERMS_INVALID_CODE ○ ERR_ADDITIONAL_INFO_INACTIVE
1.06 *(11.06.2015)*	• In method doSearchProducts added treeid search (#3364) • In method getProductsInfo added return fields: groupSymbol, groupName, expectedDeliveryDate (#3364) • In method getMyInvoices added return struct correctedItem (possibility to get sales invoice corrections) (#3364)

	• In method `doOrderProducts` added struct `deliveryAddress` and new error codes (#3284): ○ `ERR_DELIVERY_ADDRESS_INACTIVE` ○ `ERR_DELIVERY_ADDRESS_EMPTY_NAME_AND_COMPANY` • In method `getMyInvoices` added return field `wmsStatus` (#3283) • In methods `getProductsInfo`, `doOrderProducts` added parameter `ILkod` (#3276) • In method `doOrderItemsMove` added new error code: `ERR_WRONG_DEST_ORDER_ID` (wrong target order id) (#3252) • In methods `doOrderProducts` and `doOrderItemEdit` added parameter `description` (#3203)
1.05 *(27.02.2015)*	• In method `doLogin` added a limit of opened session for a single user (#3126) • In method `getInvoices` added new error code: `ERR_NOT_ALLOWED_FINALDOCUMENTS_TYPE` (#3069) • Added new error codes in methods used for managing reservations (#3066): ○ `ERR_SALE_BLOCK` (Order cannot be realized. Sale blocked.) ○ `ERR_ISSUE_BLOCK` (Order cannot be realized. Issuing vouchers blocked.) ○ `ERR_ISSUES_OVERDUE` (Order cannot be realized. Issuing vouchers term exceeded) ○ `ERR_PAYMENT_OVERDUE` (Order cannot be realized. Payment term exceeded) ○ `ERR_CREDIT_LIMIT` (Order cannot be realized. Granted credit limit exceeded)
1.04 *(29.12.2014)*	• Added method `doOrderItemsMove` for changing order items positions (#2957) • In method `getProductsInfo` added return struct `subProductList` that contains list of subproducts if a new parameter `showSubProductsInfo = 1` is passed (#2931) • In method `getMyInvoices` added return field `isEDocument` that contains information about availability of digital version of document (#2929) • Added method `getEDocument`, that returns digital version of document (#2928) • In method `getMyInvoices` added parameter `finalDocuments` that allow to define type of returned documents (#2908, #2926) • In method `getProductsInfo` added return struct `photoList` that contains information about product photos if a new parameter `showPhotos = 1` is passed (#2850)
1.03 *(18.08.2014)*	• In method `getProductsInfo` added return fields: `priceBeforePromotionExclTax, priceBeforePromotionInclTax` • In method `getProductsInfo` added return struct `additionalInfo` with field `sale`; additional information of product is returned if a new parameter `showAdditionalInfo = 1` is passed • Added method `getMyRoutes` that returns list of available routes for logged user
1.02 *(12.08.2014)*	• “Change Log” changed to “Changelog” • Added dates to changelog • In method `doOrderProducts` added parameter `externalID` in struct `newOrderInfo`
1.01	• Added changelog • Added pages numeration • Added links in table of content
1.00	• First version of documentation

COMMUNICATION SCHEMA

Request:

```
<?xml version="1.0" encoding="utf-8" ?>
<getVersion>
   parameters
</getVersion>
```

Response:

```
HTTP/1.0 200 OK
(…)
<?xml version="1.0" encoding="utf-8" ?>
<getVersionResponse>
   response
</getVersionResponse>
```

Response in case of error:

```
HTTP/1.0 500 Internal Server Error
(…)
<?xml version="1.0" encoding="utf-8" ?>
<error>
   <code>error code</code>
   <msg>error message</msg>
</error>
```

Required encoding: utf-8

COMMUNICATION EXAMPLE

Request:

```
   POST /ws HTTP/1.1
   Host: 127.0.0.1
   Content-Type: text/xml; charset=utf-8
   Content-Length: 81
   User-Agent: FL
   SOAPAction: "getVersion"

<?xml version="1.0" encoding="utf-8" ?><getVersion />
```

Response:

```
   HTTP/1.0 200 OK
   Date: Thu, 14 Feb 2013 14:15:03 GMT
   Server: dbfd.WebServer
   Connection: close
   Content-Length: 85
   Cache-Control: private
   Content-Type: text/xml; charset=utf-8

<?xml version="1.0" encoding="utf-8" ?>
<getVersionResponse>
   <version>dbfd2 5.36.712 2013-02-13</version>
</getVersionResponse>
```

ERROR CODES

`ERR_UNKNOWN`

Unknown error

`ERR_INTERNAL`

Internal server error

`ERR_KEY_WRONG`

Can't read message (decoding message failure) / Wrong transmission key

`ERR_REQUEST`

Wrong request syntax

`ERR_METHOD_UNAVAILABLE`

Method unavailable (user-type connects from wrong address / method does not exist)

`ERR_LOGIN`

Wrong login or password

`ERR_SESSION`

Session expired or does not exist

LEGEND

(?)	Optional parameter
*	Details in comments

AVAILABLE METHODS

GENERAL

`getVersion`

Get system version number

`getMyCustomerInfo`

Get information about logged user

`doLogin`

Login user to system

`getMyBonuses`

Get information about promotions / bonuses

`doSetRodoStatus`

Toggle agreement of the processing personal data

`getTree`

Get products categories tree

PRODUCTS

`getProductsInfo`

Get information about products and services

`getProductFileUrl`

Put file to download and return an URL adress

`getProductStockChanges`

Get paged information about product stock changes

`doSearchProducts`

Search products

`getReplacements`

Get product replacements

`doBuyNow`

Purchase method Buy Now by goods issue note.

COMPLATINS

`doCreateComplaint`

Make a complaint

`getMyComplaints`

Get list of complaint of logged user

`getComplaintQuantityAvailable`

Get quantity can be complained for document item..

getInvoicesComplaintProduct

Get list of sales documents by the specified product and complaint date range.

doComplaintPrint

Generate PDF printout of complaint

RESERVATIONS

doOrderProducts

Make reservation for products

getMyOrders

Get order list for logged user

getOrderStockInfo

Get information about availability of order items

doOrderClose

Close order

doOrderItemDelete

Delete order item

doOrderItemEdit

Edit order item

doOrderEdit

Edit order parameters

doOrderDelete

Delete order

doOrderItemsMove

Move items between orders

DOCUMENTS

getMyInvoices

Get list of sale documents of logged user

getEDocument

Get digital version of document (PDF)

doDocumentPrint

Generate PDF printout of document

doSetDeliveryConfirmation

Delivery confirmation for WZ document

ROUTES

getMyRoutes

Get list of users routes

LOGISTICS

getMyPackages

Get list of users packages

doLogmCreatePackage

New fast sales method – create package based on the contents of another package and generate the PDF label

PAYMENTS

getMyPayments

Get list of payments

RETURNS

getMyReturns

Get list of users returns

doReturnCreatePackage

Create package number for list of returns

doDeleteReturn

Delete registered return

doEditReturn

Edit Return

doCreateReturn

Create new return

getReturnQuantityAvailable

Get available quantity product possible to return

getReturnNowProductInfo

Additional information related to product quick return

doReturnClearPackage

Delete information about package from list of returns

doReturnGetPackageLabel

Get package label for return

doReturnSendPackage

Mark return from package as sent

doReturnNow

Quick return

GENERAL

getVersion

Get system version number

Parameters:

none

Return:

1	version	(string)	Version number DBFD
2	ownum	(string)	The installation ID
3	verFull	(string)	The full version number

Error codes:

none

XML example:

```
   <?xml version="1.0" encoding="utf-8" ?>
   <getVersion />

Response:
   <?xml version="1.0" encoding="utf-8" ?>
   <getVersionResponse>
      <version>dbfd2 5.36.712 2013-02-13</version>
   </getVersionResponse>
```

JSON example:

```
   {"getVersion": {}}

Response:
   {"getVersionResponse": {
      "version": "dbfd2 5.36.712 2013-02-13 ENC:SXE JSON"
   }}
```

`getMyCustomerInfo`

Get information about logged user

Parameters:

1 sesionId	(string)	Session id obtained with doLogin method

Return:

1 customerId	(string)	User id
2 name	(string)	Name
3 shortName	(string)	Short name
4 date	(date)	[format: yyyy-mm-dd] Register date
5 FKId	(string)	Number fk
6 FKId2	(string)	Number fk2
7 taxId	(string)	Tax id
8 payer	(struct)	information about payer
1 name	(string)	Name
2 shortName	(string)	Short name
3 taxId	(string)	Tax id
4 address	(struct)	Address
1 name	(string)	Name and surname
2 companyName	(string)	Company name
3 street	(string)	Street
4 zipCode	(string)	Post code
5 city	(string)	City
6 country	(string)	Country
7 phone	(string)	Phone
8 email	(string)	Email
9 fax	(string)	Fax
9 accountManager	(string)	Manager (name + surname)
10 address	(struct)	Address
1 name	(string)	Name and surname
2 companyName	(string)	Company name
3 street	(string)	Street
4 zipCode	(string)	Post code
5 city	(string)	City
6 country	(string)	Country
7 phone	(string)	Phone
8 email	(string)	Email
9 fax	(string)	Fax
11 mailingAddress	(struct)	Mailing address
1 name	(string)	Name and surname
2 companyName	(string)	Company name
3 street	(string)	Street
4 zipCode	(string)	Post code
5 city	(string)	City
6 country	(string)	Country
7 phone	(string)	Phone
8 email	(string)	Email
9 fax	(string)	Fax
12 deliveryAddress	(struct)	Delivery address
1 name	(string)	Name and surname
2 companyName	(string)	Company name
3 street	(string)	Street
4 zipCode	(string)	Post code
5 city	(string)	City
6 country	(string)	Country
7 phone	(string)	Phone

8 email	(string)	Email
9 fax	(string)	Fax
13 comment	(string)	Comment
14 balance	(struct[])	Balance(lista z podziałem na waluty)
1 item	(struct)	Balance
1 currency	(string)	Currency
2 currencyRate	(string)	Currency rate
3 value	(float)	Value
4 overdue	(float)	Overdue value
15 isPromoActive	(int)	Promotion active (1-yes, 0-no)
16 subTerminalId	(string)	Subterminal id
17 showPaymentsOnStart	(int)	Balances at the start (1-yes, 0-no)
18 isBonusesActive	(int)	Bonuses active (1-yes, 0-no)
19 isComplaintActive	(int)	Complaints active (1-yes, 0-no)
20 isComplaintEditCar	(int)	Complaints – edition of vehicle data (1- yes, 0-no)
21 isComplaintEditDesc	(int)	Complaints – edition of defect description (1-yes, 0-no)
22 isComplaintDescRequired	(int)	Complaints - is description defect required (1-yes, 0-no)
23 isReturnActive	(int)	Active returns (1-yes, 0-no)
24 isGoodsIssueActive	(int)	Preview goods Issue note (1-yes, 0-no)
25 isExternalStockAvailable	(int)	External stock available
26 stockQuantityFormat	(int)	Stock quantity format (1 - none, 2 - '*' is anything in stock , 3 - 0-none, 1-one, 2-more, 4 – actual state, 7 – to 10 state, later >10, 8 - to 5 state, later > 5)
27 isEdocumentActive	(int)	Electronic version of invoices active (1-yes, 0-no)
28 rodoStatus	(int)	Is consent to the processing of personal data (1-yes, 0-no)
29 nextRouteDt	(string)	[yyyy-mm-dd hh:ii:ss] The closest time of departure
30 loyaltyInfo	(struct)	Information about loyalty program
1 isActive	(int)	Client active in loyalty system (1-yes, 0-no)
2 points	(int)	Available points
3 pointsExpired	(int)	Expired points
31 introText	(CDATA)	Welcome text
32 warehouseList	(struct[])	List of available warehouses
1 warehouse	(struct)	Information about warehouse
1 id	(string)	Warehouse id
2 name	(string)	Warehouse name
3 isDefault	(int)	Default warehouse (1-yes, 0-no)
4 canOrder	(int)	Can order in this warehouse (1-yes, 0-no)
5 canViewStock	(int)	Can check state stock in this warehouse (1-yes, 0-no)
2 warehouse …		
33 groupPromoList	(struct[])	List of promoted product groups
1 groupPromo	(struct)	Promoted product group
1 groupSymbol	(string)	Product group Id
2 groupName	(string)	Product group name
3 (*) dateFrom	(date)	[yyyy-mm-dd] Date of promotion start
4 (*) dateTo	(date)	[yyyy-mm-dd] Date of promotion end
2 groupPromo ...		
34 isDeliveryConfirmationRequired	(int)	Confirmation of receipt of delivery (1 - yes 0 - no)
35 deliveryConfirmationPeriod	(int)	Number of days to confirm receipt of delivery
36 deliveryConfirmationDocTypes	(string)	Delivery confirmation – document types
37 saleBlockInfo	(struct[])	Information about lock or warning
1 status	(int)	Sale lock (0 – no, 1 – yes, 2 – no, but with statement)
2 isErpSaleBlocked	(int)	Sale lock in cairo.ERP (0 – no, 1 – yes)
3 code	(string)	Error id
4 msg	(string)	Error description
38 isInvoice	(int)	Can customer use invoices (0 – no, 1 – yes)

39 isReceipt	(int)	Can customer use receipts (0 – no, 1 – yes)
40 isDD	(int)	Can customer use goods issue notes (0 – no, 1 – yes)
41 creditLimit	(struct)	Credit limit
1 value	(float)	Value of the credit limit
2 valueLeft	(float)	Value of the credit limit left
3 currency	(float)	Currency of the credit limit

Error codes:

ERR_METHOD_UNAVAILABLE	Method unavailable (user-type connects from wrong address / method does not exist)

Comments:

* deprecated (saved for compatibility with old versions)

XML example:

```
 <?xml version="1.0" encoding="utf-8" ?>
 <getMyCustomerInfo>
    <sessionId>2BZY8zSxODOOUmMOylFKSZjMsgDG564YxTMvju_D</sessionId>
 </getMyCustomerInfo>

Response:
 <?xml version="1.0" encoding="utf-8" ?>
 <getMyCustomerInfoResponse>
    <name>KOWAL Jan Kowalski</name>
    <shortName>Jan Kowalski</shortName>
    <date>2011-02-21</date>
    <fkId>123456</fkId>
    <fkId2>234567</fkId2>
    <taxId>12112112</taxId>
    <payerInfo>nip nazwa adres</payerInfo>
    <accountManager>Jan Operator</accountManager>
    <comment>dopisany komentarz</comment>
    <address>
       <name>Jan Kowalski</name>
       <companyName>KOWAL Jan Kowalski</companyName>
       <street>Uliczna</street>
       <zipCode>12-345</zipCode>
       <city>Miasto</city>
       <country>Polska</country>
       <phone>123456789</phone>
       <email>jk@mail.pl</email>
       <fax>123456789</fax
    </address>
    <mailingAddress>
       <name>Jan Kowalski</name>
       <companyName>KOWAL Jan Kowalski</companyName>
       <street>Uliczna</street>
       <zipCode>12-345</zipCode>
       <city>Miasto</city>
       <country>Polska</country>
       <phone>123456789</phone>
       <email>jk@mail.pl</email>
       <fax>123456789</fax
    </mailingAddress>
    <deliveryAddress>
       <name>Jan Kowalski</name>
       <companyName>KOWAL Jan Kowalski</companyName>
       <street>Uliczna</street>
       <zipCode>12-345</zipCode>
       <city>Miasto</city>
       <country>Polska</country>
       <phone>123456789</phone>
       <email>jk@mail.pl</email>
       <fax>123456789</fax
    </deliveryAddress>
    <payer>
       <name>interland platnik</name>
       <shortName>platnik</shortName>
       <taxId>12112112</taxId>
       <address>
          <name>Mama Jana Kowalskiego</name>
          <companyName>mamusia</companyName>
          <street>Uliczna</street>
          <zipCode>12-345</zipCode>
          <city>Miasto</city>
          <country>Polska</country>
          <phone>123456789</phone>
          <email>mama@mail.pl</email>
          <fax>123456789</fax
       </address>
    </payer>
      </getMyCustomerInfoResponse>
```

doLogin

Login user to system

Parameters:

1 userLogin	(string)	User login
2 userPassword	(string)	Md5 of user password
3(?) languageId	(string)	Language id (default: user language)

Return:

1. sessionId	(string)	Session id

Erroe codes:

ERR_WRONG_LANGUAGE	Wrong user language
ERR_USERTYPE_DENIED	Access for specified user-type denied
ERR_TOO_MANY_SESSIONS	Too many opened sessions. Use session id from previous login or wait until one of sessions expire
ERR_LOGIN_BLOCKED	Access blocked. You have exceeded the maximum time without purchasing. Please contact your provider.

Comments:

Session is closed after specified time of inactivity. Sending an inactive session id to others methods will result in error code ERR_SESSION. In that case it is needed to log in again using doLogin method and get a new session key. Number of sessions for a single user is limited to 10. If this limit is exceeded, this method will return error code ERR_TOO_MANY_SESSIONS.

XML example:

```
 <?xml version="1.0" encoding="utf-8" ?>
 <doLogin>
    <userLogin>CAIRO</userLogin>
    <userPassword>86484e9f2ff98544f98b5fefe899c9c6</userPassword>
 </doLogin>

Response:
 <?xml version="1.0" encoding="utf-8" ?>
 <doLoginResponse>
    <sessionId>Gq0HVex1eHg9qVTeI5hCVXuGGMnFaH7GYYeJ2B_D</sessionId>
 </doLoginResponse>
```

JSON example:

```
 {"doLogin": {
    "userLogin": "CAIRO",
    "userPassword": "86484e9f2ff98544f98b5fefe899c9c6"
 }}

Response:
 {"doLoginResponse": {
    "sessionId": "Gq0HVex1eHg9qVTeI5hCVXuGGMnFaH7GYYeJ2B_D"
 }}
```

getMyBonuses

Get information about promotions / bonuses

Parameters:

1 sessionId	(string)	Session id obtained with doLogin method
2 (?) getGroupPromoList	(int)	Should the method return the list of promotions for product groups (1 – yes, 0 – no)
3 (?) getProductPromoList	(int)	Should the method return the list of promotions for products (1 – yes, 0 – no)

Return:

1 bonusList	(struct[])	List of promotions / bonuses
1 bonus	(struct)	Information about promotions / bonuses
1 id	(string)	Promotions / bonuses id
2 name	(string)	Promotion name
3 type_bonus	(int)	Promotion type: 0 – standard, 1 - description
4 dateFrom	(date)	[format: yyyy-mm-dd] Date of promotion start
5 dateTo	(date)	[format: yyyy-mm-dd] Date of promotion end
6 dateVisibleFrom	(date)	[format: yyyy-mm-dd] Date of visible promotion start
7 dateVisibleTo	(date)	[format: yyyy-mm-dd] Date of visible promotion end
8 type	(int)	Target type: 1-point, 2-quantitative, 3-value, 4-increase, 5-value increase
9 valueCompare	(float)	The value of the compare
10 currentValue	(float)	The current value obtained
11 currentShow	(string)	The current value to display
12 targetValue	(string)	Target Value
13 targetIncrease	(float)	Target increase
14 progress	(float)	Target progress
15 isAvailable	(int)	Is available (achieved) 0-no, 1-yes
16 desc	(string)	Description promotion
17 currency	(string)	Currency promotion
18 taxMode	(int)	Tax mode: 0-exclude, 1- include
19 targetLevelList	(struct[])	List of target levels
1 targetLevel	(struct)	Target level
1 level	(int)	Level
2 value	(float)	Value to achieve
3 description	(string)	Description
20 groupPromoList*	(struct[])	The list of promotions for product groups
1 groupPromo	(struct)	Promotion for a product group
1 symbol	(string)	Product group symbol
2 name	(string)	Product group name
2 groupPromo …		
21 productPromoList*	(struct[])	The list of promotions for products
1 productPromo	(struct)	Promotion for a product
1 id	(string)	Product id
2 reference	(string)	Product symbol
2 productPromo …		
22 isRanking	(int)	Customer ranking: 0-no, 1-yes
23 rankingPromoList	(struct[])	Customers ranking + 2 additional ranking places
1 rankingPromo	(struct)	Information about ranking
1 customerId	(string)	Customer Id
2 positionNumber	(int)	Position in ranking
3 currentValue	(int)	current score
2 rankingPromo …		
2 bonus ...		

Error codes:

none

Remarks:

* returns at most 500 items

`doSetRodoStatus`

Toggle agreement of the processing personal data

Parameters:

1 sessionId	(string)	Session Id
2 newStatus	(int)	Agreement status (1-agreement, 2-withdrawal)
3 clientIP	(string)	Client IP (*)

Return:

1 status	(int)	1-success

Error codes:

ERR_INVALID_STATUS	Invalid agreement status
ERR_INTERNAL	Internal server error
ERR_CLIENT_IP_NOT_AVAILABLE	Invalid client IP

Comments:

* Parameter clientIP is available only in selected connection types.

getTree

Get products categories tree

Parameters:

1 sessionId	(string)	Session id
2 parId	(string)	Parent category id (if empty, function returns main categories)

Return:

1 nodeList	(struct[])	Node List
1 node	(struct)	Node
1 catId	(string)	Category id
2 parId	(string)	Parent category id
3 name	(string)	Category name
14 hasChild	(int)	Has child categories (0 = NO, 1 = YES)

Error codes:

TREE_KATID_LEN	Invalid category id

Request example:

```
{
    "getTree": {
       "parid":"000001",
       "sessionId": "....."
   }
}
```

Response example:

```
{
    "getTreeResponse": {
       "nodeList": {
           "node": [
              {
               "catid": "000002",
               "name": "podkategoria 1",
               "parid": "000001",
               "hasChild": 1
              },
              {
               "catid": "000003",
               "name": "podkategoria 2",
               "parid": "000001",
               "hasChild": 0
              },
              {
               "catid": "000010",
               "name": "podkategoria 3",
               "parid": "000001",
               "hasChild": 1
              }
           ]
       }
    }
}
```

PRODUCTS

getProductsInfo

Get information about products

Parameters:

1 sessionId	(string)	Session id obtained with doLogin method
2 productList	(struct[])	List of products (max. 50)
1 product	(struct)	Information about product
1*(?) id	(string)	Product unique id
2*(?) reference	(string)	Product reference
3*(?) tecidd	(string)	TecDoc supplier id
4*(?) tecnum	(string)	TecDoc product id
5*(?) ILkod	(string)	ILKod product id
6*(?) motonet	(string)	Motonet number
2 product ...		
3 showExternalStockInfo	(int)	Return information about external stock (0-no, 1-only local, 2-local and remote).
4 showAdditionalInfo	(int)	Return additional information about product (0-no, 1-yes)
5 showPhotos	(int)	Return product photos (0-no, 1-yes)
6 showSubProductsInfo	(int)	Return secondary products (0-no, 1-yes)

Return:

1 productInfoList	(struct[])	Product list
1 productInfo	(struct)	Information about product
1 id	(string)	Product unique id
2 type	(int)	Product type (0-article, 1-service)
3 reference	(string)	Product reference
4(?) reference2	(string)	Product reference 2
5(?) ILkod	(string)	ILkod product id
6(?) motonet	(string)	Motonet number
7(?) tecidd	(string)	TecDoc supplier id
8(?) tecnum	(string)	TecDoc product id
9 name	(string)	Product name in user language
10 groupSymbol	(string)	Product group symbol
11 isGroupPromo	(int)	Is group promoted for client (always 0)
12 groupName	(string)	Product group name
13(?) group2	(string)	Group 2 symbol
14 decimals	(int)	Decimal places (0-3)
15 tax	(float)	Tax rate
16(?) priceCatExclTax	(float)	Catalog price without tax
17(?) priceCatInclTax	(float)	Catalog price with tax
18 currency	(string)	Currency
19 unit	(string)	Metric unit
20 blocked	(int)	Sale lock (1-on, 0-off)
21 files	(int)	Does product have attach any files?(1-yes, 0-no)
22(?) hasDeposit	(int)	Does product have deposit? (1-yes, 0-no)
23(?) depositPriceExclTax	(float)	Deposit value without tax
24(?) depositPriceInclTax	(float)	Deposit value with tax
25(?) hasProductDeposit	(int)	Does product have a material deposit (1 - yes, 0 - no)
26(?) expectedDeliveryDate	(date)	[format: yyyy-mm-dd] Expected delivery date
27 isReplacement	(int)	Has replacement (1 - yes, 0 - no)
28 isPhoto	(int)	Has photo(1 - yes, 0 - no)
29(?) producer	(string)	Producer
30 gtuCode	(string)	GTU code
31 stockInfoList	(struct[])	Information about stock

1 stockInfo	(struct)	Information about stock
1 warehouseId	(string)	Warehouse id
2 viewOnly	(int)	Is ordering disabled (1-yes, 0-no)
3 inventory	(int)	Inventory in progress (1-yes, 0-no)
4 quantityShow	(string)	Stock available [include price list] (string, e.g. "> 5")
5 quantity	(float)	Stock available [include price list] (number)
6 quantityRealShow	(string)	Stock available [in warehouse] (string, e.g. "> 5")
7 quantityReal	(float)	Stock available [in warehouse] (number)
8(?) priceExclTax	(float)	Price without tax
9(?) priceInclTax	(float)	Price with tax
10(?) priceBeforePromotionExclTax	(float)	Price before promotion without tax
11(?) priceBeforePromotionInclTax	(float)	Price before promotion with tax
12(?) promotionPriceExclTax	(float)	Promotion price withou tax
13(?) promotionPriceInclTax	(float)	Promotion price with tax
14(?) promotionDiscount	(float)	Discount promotion
15 quantityMultiplier	(int)	Multiplier (0-off)
16(?) externalStockInfo	(struct)	Information about external stock
1 warehouseShow	(string)	Name of warehouse
2 quantityShow	(string)	Stock available (string, e.g. "> 5")
3 quantity	(float)	Stock available (number)
4(?) deliveryTimeMin	(int)	Minimal delivery time in hours
5(?) deliveryTimeShow	(string)	Delivery time (string, e.g. "1 - 2 days")
17(?) additionalStockInfo	(struct[])	Additional information about stock
1(?) sale	(int)	Sale active (1-yes, 0-no)
2 stockInfo ...		
32(?) photoList	(struct[])	List of product photos
1 photo	(struct)	Information about single photo
1 url	(string)	Photo URL
2 description	(string)	Photo description
3 isDefault	(int)	Default photo (1-yes, 0-no)
2 photo …		
33 subProductList	(struct[])	List of secondary products
1 subProductInfo	(struct)	Information about secondary product
1 id	(string)	Product id
2 reference	(string)	Product reference
3 name	(string)	Product name in user language
4 tax	(float)	Tax rate
5 unit	(string)	Unit
6 decimals	(int)	Decimal places (0-3)
7 quantityMultiplier	(int)	Multiplier (0-off)
8 main	(int)	1-main product 0-other
9 currency	(string)	Currency
10 size	(string)	Size
11 color	(string)	Color
12 subBarCodeList	(struct)	List of barcodes
1 subBarCode	(struct)	Information about single barcode
1 code	(string)	Barcode
2 quantity	(float)	Product quantity
3 default	(string)	Default barcode (1-yes, 0-no)
13 subBarCodeList ...		
2 subProductInfo ...		
2 productInfo ...		

Error codes:

ERR_PRODUCTLIST_OVERSIZED	Product list size exceeded (max: 200)

Comments:

* One of these is required: "id" or "reference", or both "tecidd" and "tecnum", or “motonet”.
* In case of sending TecDoc data or Motonet number - information about first fitting product is returned.
* Informations about products are returned in the same order as in request.
* If product is not found, not available for specified user-type, not found in any warehouse available for client or client doesn't have access to any warehouse, a blank struct <productInfo /> will be returned.

XML example:

```
Request:
<?xml version="1.0" encoding="utf-8" ?>
<getProductsInfo>
   <sessionId>GqOHVex1eHg9qVTeI5hCVXuGGMnFaH7GYYeJ2B_D</sessionId>
   <productList>
      <product>
         <reference>0-001-231-030</reference>
      </product>
      <product>
         <reference>SYMBOL123</reference>
      </product>
   </productList>
  <showExternalStockInfo>2</showExternalStockInfo>
  <showPhotos>1</showPhotos>
</getProductsInfo>

Response:
<?xml version="1.0" encoding="utf-8" ?>
<getProductsInfoResponse>
   <productInfoList>
      <productInfo>
         <id>003LFT</id>
         <type>0</type>
         <reference>0-001-231-030</reference>
         <name>Osłona przegubu /BOSCH/</name>
         <decimals>2</decimals>
         <tax>22</tax>
         <priceCatExclTax>52.49</priceCatExclTax>
         <priceCatInclTax>64.04</priceCatInclTax>
         <currency>zł</currency>
         <unit>szt</unit>
         <blocked>0</blocked>
         <files>0</files>
         <hasDeposit>0</hasDeposit>
         <color>pink</color>
         <size>XXL</size>
         <barcodeList>
            <barcode>
               <code>1234567</code>
               <quantity>2</quantity>
               <default>0</default>
            </barcode>
            <barcode>
               <code>2345678</code>
               <quantity>1</quantity>
               <default>1</default>
            </barcode>
         </barcodeList><stockInfoList>
            <stockInfo>
               <warehouseId>XX</warehouseId>
               <inventory>0</inventory>
               <quantityShow>&gt; 5.00</quantityShow>
               <quantity>6.00</quantity>
               <priceExclTax>55.57</priceExclTax>
               <priceInclTax>67.79</priceInclTax>
               <priceBeforePromotionExclTax>75.35</priceBeforePromotionExclTax>
               <priceBeforePromotionInclTax>96.22</priceBeforePromotionInclTax>
          <quantityMultiplier>0</quantityMultiplier>
             <externalStockInfo>
                   <warehouseShow>XXX</warehouseShow>
                   <quantityShow>2.00</quantityShow>
                   <quantity>2.00</quantity>
                   <deliveryTimeMin>24</deliveryTimeMin>
                   <deliveryTimeShow>1 day</deliveryTimeShow>
             </externalStockInfo>
        </stockInfo>
        <stockInfo>
           <warehouseId>CE</warehouseId>
           <inventory>0</inventory>
           <quantityShow>2.00</quantityShow>
           <quantity>2.00</quantity>
           <priceExclTax>5000.00</priceExclTax>
           <priceInclTax>6100.00</priceInclTax>
           <quantityMultiplier>0</quantityMultiplier>
        </stockInfo>
        <stockInfo>
```

```
          <warehouseId>OL</warehouseId>
          <inventory>0</inventory>
          <quantityShow>0.00</quantityShow>
          <quantity>0.00</quantity>
          <priceExclTax>1294.61</priceExclTax>
          <priceInclTax>1579.42</priceInclTax>
          <quantityMultiplier>0</quantityMultiplier>
       </stockInfo>
    </stockInfoList>
    <photoList>
       <photo>
          <url>http://upload.zdjecia.com/katalog/1/zdjecie1.jpg</url>
          <description>BOSCH Osłona przegubu, zdjęcie - przód</description>
          <isDefault>1</isDefault>
       </photo>
    </photoList>
    <additionalInfo>
       <sale>1</sale>
    </additionalInfo>
 </productInfo>
 <productInfo />
   </productInfoList>
      </getProductsInfoResponse>
```

JSON example:

```
Request:
 {
   "getProductsInfo": {
     "sessionId": "GqOHVex1eHg9qVTeI5hCVXuGGMnFaH7GYYeJ2B_D",
     "productList": {
       "product": [
         { "reference": "0-001-231-030" },
         { "reference": "SYMBOL123" }
       ]
     },
     "showExternalStockInfo": "2"
   }
 }

Response:
 {
   "getProductsInfoResponse": {
     "productInfoList": {
       "productInfo": {
         "id": "003LFT",
         "type": "0",
         "reference": "0-001-231-030",
         "name": "Osłona przegubu /BOSCH/",
         "decimals": "2",
         "tax": "22",
         "priceCatExclTax": "52.49",
         "priceCatInclTax": "64.04",
         "currency": "zł",
         "unit": "szt",
         "blocked": "0",
         "files": "0",
         "hasDeposit": "0",
         "color": "pink",
         "size": "XXL",
         "barcodeList": {
           "barcode": [
             {
               "code": "1234567",
               "quantity": "2",
               "default": "0"
             },
             {
               "code": "2345678",
               "quantity": "1",
               "default": "1"
             }
           ]
         },       "stockInfoList": {
           "stockInfo": [
             {
               "warehouseId": "XX",
               "inventory": "0",
               "quantityShow": "> 5.00",
               "quantity": "6.00",
               "priceExclTax": "55.57",
               "priceInclTax": "67.79",
               "priceBeforePromotionExclTax" : "75.35",
               "priceBeforePromotionInclTax" : "96.22",
               "quantityMultiplier": "0",
               "externalStock": {
                 "warehouseShow": "XXX",
                   "quantityShow": "2.00",
                 "quantity": "2.00",
                 "deliveryTimeMin": "24",
                 "deliveryTimeShow": "1 day"
               }
             },
             {
               "warehouseId": "CE",
               "inventory": "0",
               "quantityShow": "2.00",
               "quantity": "2.00",
               "priceExclTax": "5000.00",
               "priceInclTax": "6100.00",
               "quantityMultiplier": "0"
             },
             {
               "warehouseId": "OL",
               "inventory": "0",
               "quantityShow": "0.00",
               "quantity": "0.00",
                             "priceExclTax": "1294.61",
               "priceInclTax": "1579.42",
               "quantityMultiplier": "0"
             }
           ]
         },
         "additionalInfo": {
           "sale": "1"
         }
       }
     }
   }
 }
```

`getProductFileUrl`

Put file to download and return an URL adress

Parameters:

1 `sessionId`	`(string)`	Session id obtained with doLogin method
2 `fileId`	`(string)`	File id
3 `productId`	`(string)`	Product id

Return:

1 `url`	`(string)`	File url adress

Error codes:

ERR_WRONG_FILE_ID	Invalid file id
ERR_SERVER_NOT_CONFIGURED	Server is not configured correctly

XML example:

```
Request:
<?xml version="1.0" encoding="utf-8" ?>
<getProductFileUrl>>
   <fileId>00000013</fileId>
   <productId>0003B4</productId>
   <sessionId>YhQ4oWGiBJeiZ2LByY2Im7yViK3F27y0017943_D</sessionId>
</getProductFileUrl>

Response:
<?xml version="1.0" encoding="utf-8" ?>
<getProductFileUrlResponse>
   <url>http://produkty.com/katalog/1/produkt1.txt</url>
</getProductFileUrlResponse>
```

JSON:

```
Request:
{
   "getProductFileUrl": {
      "sessionId": "YhQ4oWGiBJeiZ2LByY2Im7yViK3F27y0017943_D",
      "fileId": "00000013",
      "productId": "0003B4"
      }
}

Response:
{
   "getProductFileUrlResponse": {
      "url": http://produkty.com/katalog/1/produkt1.txt
      }
}
```

getProductStockChanges

Get paged information about product stock changes

Parameters:

1 sessionId	(string)	Session id obtained with doLogin method
2 warehouseId	(string)	Warehouse id
3 fromDt	(string)	[yyyy-mm-dd hh:ii:ss] Starting date for stock changes analysis (*)
4 (?) pageSize	(int)	Number of results on a single page (max. 1000, def. 100) (*)
5 (?) pageNumber	(int)	Page number (def. 1) (*)

Return:

1 productStockList	(struct[])	Product stock list
1 productStock	(struct)	Information about product stock status
1 productId	(string)	Product id
2 changeDt	(string)	[yyyy-mm-dd hh:ii:ss] The most recent stock change date
3 quantity	(float)	Product quantity
4 quantityExt	(float)	External product quantity
2 productStock ...		

Error codes:

ERR_INVALID_WAREHOUSE	Invalid warehouse
ERR_INVALID_FROM_DATE	Invalid starting date
ERR_WAREHOUSE_ACCESS_DENIED	Warehouse access denied
ERR_RESULT_LIMIT	Result limit exceeded

Request example:

```
{
   "getProductStockChanges": {
      "sessionId": "vG1mNExIAPL7LXUs5dOVzyGeYeX7ra80024259_D",
      "warehouseId": "01",
      "fromDt": "2021-09-01 16:17:25"
   }
}
```

Response example:

```
{
   "getProductStockChangesResponse": {
      "productStockList": {
         "productStock": [
            {
               "changeDt": "2021-09-21 14:07:28",
               "productId": "0003AZ",
               "quantity": "1253",
               "quantityExt": "0"
            },
            {
               "changeDt": "2021-09-21 14:07:28",
               "productId": "0003B0",
               "quantity": "8468",
               "quantityExt": "0"
            }
         ]
      }
   }
}
```

doSearchProducts

Search products

Parameters:

1 sessionId	(string)	Session id obtained with doLogin method
2 searchString	(string)	Searched phrase or multiple phrases separated with space
3 searchMode	(struct[])	Search option
1 modeName	(string)	Available options: id, reference, name, tecdoc, treeid
2 modeName ...		

Return:

1 productInfoList	(struct[])	List of found products
1 productInfo	(struct)	Informations about found product
1 id	(string)	Product id
2 reference	(string)	Product reference
3 name	(string)	Product name in user language
4 source	(int)	Source: 1-id, 2-reference, 3-reference2, 4-original number, 5-supplier reference, 6-name, 7-tecdoc, 8-treeid
2 productInfo ...		

Error codes:

ERR_INVALID_WAREHOUSE	Invalid warehouse
ERR_WAREHOUSE_ACCESS_DENIED	No access to warehouse
ERR_INVALID_PRODUCT	Invalid product

Comments:

Function returns max. 50 products containing searching phrase.

XML example:

```
Request:
<?xml version="1.0" encoding="utf-8" ?>
<doSearchProducts>
   <sessionId>Gq0HVex1eHg9qVTeI5hCVXuGGMnFaH7GYYeJ2B_D</sessionId>
   <searchString>96575</searchString>
   <searchMode>
      <modeName>name</modeName>
 <modeName>reference</modeName>
   </searchMode>
</doSearchProducts>

Response:
<?xml version="1.0" encoding="utf-8" ?>
<doSearchProductsResponse>
   <productInfoList>
      <productInfo>
    <id>0AX451</id>
    <reference>96575</reference>
    <name>Osłona przegubu /BOSCH/</name>
    <source>2</source>
      </productInfo>
      <productInfo />
   </productInfoList>
</doSearchProductsResponse>
```

JSON example:

```
{
  "doSearchProducts": {
    "sessionId": "Gq0HVex1eHg9qVTeI5hCVXuGGMnFaH7GYYeJ2B_D",
    "searchString": "96575",
    "searchMode": {
      "modeName": [
        "name",
        "reference"
      ]
    }
  }
}

Response:
{
  "doSearchProductsResponse": {
    "productInfoList": {
      "productInfo": {
        "id": "0AX451",
        "reference": "96575",
        "name": "Osłona przegubu /BOSCH/",
        "source": "2"
      }
    }
  }
}
```

getReplacements

Get product replacements

Parameters:

`1 sessionId`	`(string)`	Session id obtained with doLogin method
`2 productId`	`(string)`	Product id
`3 (?) warehouseId`	`(string)`	Warehouse id

Return:

`1 replacementList`	`(struct[])`	List of replacements
`  1 replacementInfo`	`(struct)`	Information about replacement (product)
`    1 id`	`(string)`	Product id
`    2 reference`	`(string)`	Product reference
`    3 name`	`(string)`	Product name
`  2 replacementInfo ...`		

Error codes:

ERR_INVALID_WAREHOUSE	Invalid warehouse
ERR_WAREHOUSE_ACCESS_DENIED	No access to warehouse
ERR_INVALID_PRODUCT	Invalid product

Comments:

Replacements are sorted according to program settings

`doBuyNow`

Purchase method Buy Now by goods issue note.

Parameters:

1	sessionId	(string)	Session id obtained with doLogin method
2(?)	warehouseId	(string)	Warehouse id, which products will be ordered. (default: The default warehouse for the client)
3(?)	documentId	(string)	Document id, which order products. (default: Today unprintable document client)
4(?)	forceNewOrder	(int)	Force making a new order (1-yes, 0-no, default 0)
5	productOrderList	(struct[])	List of products to order
1	productOrder	(struct)	Information about product
1(?)*	id	(string)	Product id
2(?)*	reference	(string)	Product reference
3(?)*	tecidd	(string)	TecDoc supplier id
4(?)*	tecnum	(string)	TecDoc product id
5(?)*	ILkod	(string)	ILkod product id
6(?)*	motonet	(string)	Motonet number
7	quantity	(float)	Quantity to order
2	productOrder ...		

Return:

1	documentInfo	(struct)	Information about document, which ordered products.
1	id	(string)	Document id
2	number	(string)	Document number
2	productOrderInfoList	(struct[])	Information about ordered products
1	productOrderInfo	(struct)	Information about ordered product (if success)
1	id	(string)	Product id
2	reference	(string)	Product reference
3	ILkod	(string)	ILkod product id
4	name	(string)	Product name in user language
5	decimals	(int)	Decimal places (0-3)
6	tax	(float)	Tax rate in % ("-1" - exempt, "-2" - untaxed)
7	priceExclTax	(float)	Price without tax
8	priceInclTax	(float)	Price with tax
9	currency	(string)	Currency
10	unit	(string)	Unit
11	warehouseId	(string)	Warehouse id
12	quantity	(float)	Quantity ordered
2	productOrderInfo	(struct)	Information about ordered product (if error)
1	error	(string)	Error message
3	productOrderInfo ...		

Error codes:

ERR_METHOD_UNAVAILABLE	Method unavailable (user-type connects from wrong address / method does not exist)
ERR_NO_WAREHOUSE	No available warehouses to order
ERR_WAREHOUSE_UNAVAILABLE	In warehouse %% user can't make orders
ERR_METHOD_BLOCKED	Method is blocked
ERR_PRODUCTORDERLIST_OVERSIZED	Products list oversized (max: 50)
ERR_CANT_FORCE_NEW_ORDER	Can't force new order (orderId passed)
ERR_INVALID_DOCUMENT_ID	Invalid document id
ERR_ORDER_LOCKED	Document is locked by other user

Comments:

* One of these is required: "id" or "reference" or both "tecidd" and "tecnum" or “motonet”

* In case of sending TecDoc or Motonet data - information about first fitting product is returned.
* Informations about products are returned in the same order as in request.

COMPLAINTS

doCreateComplaint

Make a complaint

Parameters:

1 sessionId	(string)	Session id obtained with doLogin method
2 complaintInfo	(struct)	Information about complaint
1 srcDocumentId	(string)	Source document id
2 srcItemId	(string)	Document item id
3 productId	(string)	Product id
4 (?) dateProductDelivery	(date)	[format: yyyy-mm-dd] Date of product delivery
5 (?) serialNumber	(string)	Date of product delivery
6 (?)** defectDescription	(string)	Defect description
7 quantity	(float)	Quantity
8 (?) decisionIfYes	(string)	Decision if accepcted: "K"-money return, "W"-product replacement
9 (?) decisionIfNo	(string)	Decision if rejected: "Z"-recycling, "O"-product return
10 (?)*** confirmReComplaint	(int)	Confirmation of recomplaining the same product (1-yes, 0-no)
11 (?) registerNumber	(string)	Vehicle registration number
12 (?) engineNumber	(string)	Engine number
13 (?) vinNumber	(string)	VIN number
14 (?) engineCapacity	(string)	Engine Capacity
15 (?) installDate	(date)	[format: yyyy-mm-dd] Date of assembly parts
16 (?) counterStateInstall	(int)	Mileage counter before assembly
17 (?) workshopInstallName	(string)	Assembly - workshop name
18 (?) workshopInstallAddress	(string)	Assembly - workshop address
19 (?) uninstallDate	(date)	[format: yyyy-mm-dd] Date of disassembly parts
20 (?) counterStateUninstall	(int)	Mileage counter before disassembly
21 (?) workshopUninstallName	(string)	Disassembly - workshop name
22 (?) workshopUninstallAddress	(string)	Disassembly - workshop address
23 (?) productionYear	(string)	Year production vehicle
24 (?) brandVehicle	(string)	Mark vehicle
25 (?) **powerValue	(int)	Power - value
26 (?) **powerUnit	(int)	Power - unit
27 (?) **installedBy	(int)	Installed by (0-workshop, 1-user)
28 (?) **uninstalledBy	(int)	Uninstalled był (0-workshop, 1-user)
29 (?) **faultFoundBy	(int)	Fault found by (0-workshop, 1-user)
30 (?) **faultWhen	(int)	When a fault was found (0-before assembly, 1-during assembly, 2-during exploitation, 3-after the collision)
31 (?) **refuseAddCosts	(int)	The complainant resigns from additional costs (0-no, 1-yes)
32 (?) compensationValue	(int)	Compensation value
33 (?) compensationCurrency	(string)	Compensation currency

Return value:

1 complaintId	(string)	Complaint id

Error codes:

ERR_COMPLAINTS_INACTIVE	Complaints are not active
ERR_WRONG_DOCUMENT_ID	Invalid document id
ERR_WRONG_ITEM_ID	Invalid document item id
ERR_WRONG_PRODUCT_ID	Invalid product id
ERR_INVALID_QUANTITY	Invalid quantity

`ERR_QUANTITY_TOO_MUCH`	Too much quantity to complain (max: %%)
`ERR_PRODUCT_ALREADY_COMPLAINED`	Product was already complained
`ERR_DEFECT_DESCRIPTION_INACTIVE`	Defect description is not available
`ERR_DEFECT_DESCRIPTION_REQUIRED`	Defect description is required
`ERR_INVALID_DECISION_IF_YES`	Illegal decision if accepted
`ERR_INVALID_DECISION_IF_NO`	Illegal decision if rejected
`ERR_CAR_DATA_INACTIVE`	Edition of vehicle data is not available

Comments:

** This functionality can be limited

*** If the product was already complained, an error code `ERR_PRODUCT_ALREADY_COMPLAINED` will be returned, unless a parameter `confirmReComplaint` = 1 is sent, then a re-complaint will be accepted.

getMyComplaints

Get list of complaints of logged user

Parameters:

`1 sessionId`	`(string)`	Session id obtained with doLogin method
`2 (?)* documentIdList`	`(string[])`	List of documents to return
`   1 id`	`(string)`	Document id
`      2 id ...`		
`3 (?)* state`	`(string)`	Complaint state (0-closed, 1-opened, 2-opened by customer), default 1
`4 (?) filterOptions`	`(struct)`	Document filters
`   1 (?) dateFrom`	`(date)`	[format: yyyy-mm-dd] Begin of date range. By default, documents from last 7 days are returned (if parameter documentIdList was NOT given) or from any range (if parameter documentIdList was given)
`   2 (?) dateTo`	`(date)`	[format: yyyy-mm-dd] End of date range (default: today)
`5 (?) sortOptions`	`(struct)`	Sorting options
`   1 (?) sortType`	`(int)`	Sort type (1-by date of create (default))
`   2 (?) sortOrder`	`(int)`	Sort order (1-ascending, 2-descending (default))
`6 (?) pageSize`	`(int)`	Number of documents on page (min: 1, max: 50, default: 10)
`7 (?) pageNumber`	`(int)`	Page number (default: 1)

Return:

`1 complaintList`	`(struct[])`	List of complaints
`   1 complaint`	`(struct)`	Information about complaint
`      1 id`	`(string)`	Complaint id
`      2 number`	`(string)`	Complaint number
`      3 state`	`(string)`	Complaint state (0-closed, 1-opened, 2-opened by customer)
`      4 dateCreate`	`(date)`	[format: yyyy-mm-dd] Date of create
`      5 dateClose`	`(date)`	[format: yyyy-mm-dd] Date of decision
`      6 decision`	`(int)`	Decision id
`      7 decisionText`	`(string)`	Description of the decision
`      8 decisionType`	`(int)`	Decision type: 0-none, 1-accepted, 2- rejected
`      9 status`	`(string)`	Complaint status
`      10 statusText`	`(string)`	Complaint status description
`      11 statusDate`	`(date)`	[format: yyyy-mm-dd] Date of status change
`      12 productId`	`(string)`	Product id
`      13 productName`	`(string)`	Product name
`      14 productReference`	`(string)`	Product reference
`      15 quantity`	`(int)`	Quantity complained
`      16 priceNet`	`(int)`	Price net
`      17 valueNet`	`(int)`	Value net
`      18 srcDocumentType`	`(string)`	Source document type
`      19 srcDocumentNumber`	`(string)`	Source document number
`      20 srcDocumentDate`	`(date)`	[format: yyyy-mm-dd] Date of sale
`      21 srcDocumentId`	`(string)`	Source document id
`      22 defectDescription`	`(string)`	Defect description
`      23 serialNumber`	`(string)`	Serial number
`      24 decisionIfYes`	`(string)`	Decision if accepted (id)
`      25 decisionIfYesText`	`(string)`	Decision if accepted (description)
`      26 decisionIfNo`	`(string)`	Decision if rejected (id)
`      27 decisionIfNoText`	`(string)`	Decision if rejected (description)
`      28 decisionComment`	`(string)`	Description of the decision complaint
`      29 comment`	`(string)`	Comments
`      30 storeExt`	`(string)`	Warehouse id complaint

```
    31 currency                (string)        Currency
    32 productionYear          (string)        Year production vehicle
    33 brandVehicle            (string)        Mark vehicle
    34 registerNumber          (string)        Vehicle registration numbe
    35 engineNumber            (string)        Engine number
    36 vinNumber               (string)        VIN number
    37 engineCapacity          (string)        Engine Capacity
    38 installDate             (date)          [format: yyyy-mm-dd] Date of assembly parts
    39 counterStateInstall     (int)           Mileage counter before assembly
    40 workshopInstallName     (string)        Assembly - workshop name
    41 workshopInstallAddress  (string)        Assembly - workshop address
    42 uninstallDate           (date)          [format: yyyy-mm-dd] Date of disassembly parts
    43 counterStateUninstall   (int)           Mileage counter before disassembly
    44 workshopUninstallName   (string)        Disassembly - workshop name
    45 workshopUninstallAddress (string)       Disassembly - workshop address
 2 complaint …
```

Error codes:

ERR_COMPLAINTS_INACTIVE	Complaints are not active
ERR_WRONG_COMPLAINTS_STATE	Invalid complaint state

getComplaintQuantityAvailable

Get quantity can be complained for document item.

Parameters:

1 sessionId	(string)	Session id obtained with doLogin method
2 complaintInfo	(struct)	Information about complaint
1 srcDocumentId	(string)	Complaint id
2 srcItemId	(string)	Complaint number
3 productId	(string)	Product id

Return:

1 quantity	(float)	Quantity can be complained

Error codes:

ERR_COMPLAINTS_INACTIVE	Complaints are not active
ERR_WRONG_DOCUMENT_ID	Wrong document id
ERR_WRONG_ITEM_ID	Wrong item id
ERR_WRONG_PRODUCT_ID	Wrong product id

getInvoicesComplaintProduct

Get list of sales documents by the specified product and complaint date range.

Parameters:

1 sessionId	(string)	Session id obtained with doLogin method
2 productId	(string)	Product Id
3 dateFrom	(date)	[format: yyyy-mm-dd] Begin of date range of documents returned.
4(?) dateTo	(date)	[format: yyyy-mm-dd] End of date range of documents returned, (default: today)
5(?) warehouseId	(string)	Warehouse id, which documents will be returned. (default: all)

Return value:

```
1 invoiceList                (struct[])      List of documents
   1 invoice                 (struct)        Information about document
      1 id                   (string)        Document id
      2 number               (string)        Document number
      3 date                 (date)          [format: yyyy-mm-dd] Document date
      4 warehouseId          (string)        Warehouse id
      5 quantity             (float)         Quantity
      6 quantityAvailable    (float)         Quantity possible to return (all)
      7 decimals             (int)           Decimal places (0-3)
      8 itemList             (struct[])      List of items
         1 item              (struct)        Item with product
            1 itemId         (string)        Item id
            2 quantity       (float)         Quantity
            3 quantityAvailable (float)      Quantity possible to return
         2 item …
   2 invoice …
```

Error codes:

ERR_INVALID_PRODUCT	Invalid product id
ERR_INVALID_DATEFROM	Invalid begin of date range
ERR_INVALID_DATETO	Invalid end of date range
ERR_INVALID_WAREHOUSE	Invalid warehouse id

doComplaintPrint

Generate PDF printout of complaint

Parameters:

1	sessionId	(string)	Session id obtained with doLogin method
2	complaintId	(string)	Complaint Id

Return:

1	pdfData	(string)	Complaint printout (PDF coded Base64)

Error codes:

ERR_COMPLAINTS_INACTIVE	Complaint module is not available
ERR_WRONG_COMPLAINT_ID	Invalid complaint id

RESERVATION

doOrderProducts

Make reservation for products

Parameters:

1 sessionId	(string)	Session id obtained with doLogin method
2(?) warehouseId	(string)	Warehouse id in which reservation will be made (if parameter is not given, then reservation will be made in default warehouse)
3(?) orderId	(string)	Order id in which products will be added (if parameter is not given, products will be added to newest opened order or (if not found) a new order will be created)
4(?) dateDelivery	(date)	Date of order realization [format: yyyy-mm-dd]
5(?) forceNewOrder	(int)	Force making a new order (1-yes, 0-no, default 0)
6(?) onlyFoundItems	(int)	Should the error be returned if any of the items is not found (1-yes, 0-no, default 0) (only if forceNewOrder = 1)
7(?) newOrderInfo	(struct)	Information about new order (only if `forceNewOrder` = 1)
1(?) externalId	(string)	Order external id
2(?) comment	(string)	Comment to order
3(?) srcDocId	(string)	Source document id
4**(?) incotermsCode	(string)	Incoterms code
5**(?) additionalInfo	(string)	Additional information
6(?) isReward	(int)	Bring out reward marker (1 – yes, 0 – no (default))
7(?) packMergeKey	(int)	The key used to merge orders into a single package (if empty, the order will be packed separately)
8(?) retShelfMerge	(int)	Specifies the order in which products are taken out of return shelves (0 – normal, 1 – first, 2 – last)
9(?) deliveryAddress**	(struct)	Delivery address
1 name	(string)	Name and surname
2 company	(string)	Company name
3 street	(string)	Street, street number
4 postcode	(string)	Post code
5 city	(string)	City
6 country	(string)	Country
7 phone	(string)	Phone
8 email	(string)	E-mail
10(?) additionalDeliveryAddress**	(struct)	Delivery address 2
1 name	(string)	Name and surname
2 company	(string)	Company name
3 street	(string)	Street, street number
4 postcode	(string)	Post code
5 city	(string)	City
6 country	(string)	Country
7 phone	(string)	Phone
8 email	(string)	E-mail
11(?) pickupPointAddress	(struct)	Pick up point address
1 name	(string)	Name and surname
2 company	(string)	Company name
3 street	(string)	Street, street number
4 postcode	(string)	Post code
5 city	(string)	City
6 country	(string)	Country
7 phone	(string)	Phone
8 email	(string)	E-mail

9 pickupPointId	(string)	DHLD - Number of the packstation
10 packstationPostNumber	(string)	DHLD - Post Nummer of the postfiliale
12(?) payment	(string)	Payment information
1 type	(int)	Payment type (2-cash, 3-cash in term, 5-bank wire, 6-cash/bank wire, 7-COD, 8-credit card)
2(?) days	(int)	Payment term in days (if not given, default value will be inserted)
13(?) routeId	(string)	Route id
15**(?) packageProductId	(string)	Product Id- package
16**(?) deliveryOnSaturday	(int)	Delivery on Saturday (1=yes, 0=no, default 0)
17**(?) sendOnSaturday	(int)	Send on Saturday (1=yes, 0=no, default 0)
18 (?) shipmentType	(string)	Shipment type
19 (?) handoverToDD	(int)	Hand over to a goods issue note (1 – yes, 0 – no, by default the parameter is taken from the customer settings)
20 (?) superSupplier	(int)	Is the order to the super supplier (1 – yes, 0 – no (default))
21 (?) customerDestinationWarehouseId	(string)	For warehouse orders, the symbol of the warehouse where the order was created
22 (?) isCustomerSupply	(int)	For warehouse orders, is is the supply order
8 productOrderList	(struct[])	List of products to order
1 productOrder	(struct)	Information about product
1*(?) id	(string)	Product id
2*(?) reference	(string)	Product reference
3*(?) tecidd	(string)	TecDoc supplier id
4*(?) tecnum	(string)	TecDoc product id
5*(?) ILkod	(string)	ILkod product id
6*(?) motonet	(string)	Motonet number
7 quantity	(float)	Quantity to order
8(?) requireStockBlock	(int)	Information whether product should be ordered only if user can block quantity in warehouse (1-yes, 0-no, default 0)
9(?) finalDocumentType	(int)	Type of final document, "conversion direction" (0-default, 1-invoice, 2-bill, default 0)
10(?) customerReference	(string)	Customer reference
11(?) packageType	(int)	Package type (0-self, 1-client, 2-none, default 0)
12(?) description	(string)	Item description
13(?) dataExpire	(date)	[format: yyyy-mm-dd] Minimum expiration date
14(?) externalItemId	(string)	External item id
2 productOrder ...		

Return:

1 orderInfo	(struct)	Information about order
1 id	(string)	Order id
2 number	(string)	Order number
2 productOrderInfoList	(struct[])	Information about ordered products
1 productOrderInfo	(struct)	Information about ordered product (if success)
1 id	(string	Product id
2 reference	(string)	Product reference
3 ILkod	(string)	ILkod product id
4 motonet	(string)	Motonet number
5 name	(string)	Product name in user language
6 decimals	(int)	Decimal places (0-3)
7 tax	(float)	Tax rate
8 priceExclTax	(float)	Price without tax
9 priceInclTax	(float)	Price with tax
10 currency	(string)	Currency
11 unit	(string)	Unit
12 warehouseId	(string)	Warehouse id

```
      13 quantity              (float)          Quantity ordered
      14 quantityBlocked       (float)          Quantity blocked in warehouse
      15 quantityNotBlocked    (float)          Quantity not blocked in warehouse
   2 productOrderInfo          (struct)         Information about ordered product (if error)
      1 error                  (string)         Error message
   3 productOrderInfo ...
```

Error codes:

ERR_NO_WAREHOUSE	No available warehouses
ERR_WAREHOUSE_UNAVAILABLE	In warehouse %% user can't make orders
ERR_METHOD_BLOCKED	Method is blocked
ERR_PRODUCTORDERLIST_OVERSIZED	Products list oversized (max: 200)
ERR_CANT_FORCE_NEW_ORDER	Can't force new order (orderId passed)
ERR_COMMENT_TOO_LONG	Comment too long (max 30 chars)
ERR_COURIER_INFO_EDIT_INACTIVE	Editing parameters courier unavailable
ERR_EXTERNALID_TOO_LONG	External order number is too long (max 47 chars)
ERR_EXTERNALID_ALREADY_EXISTS	External order number already exists
ERR_DATEDELIVERY_WRONG_FORMAT	Wrong order delivery date type
ERR_DELIVERY_ADDRESS_INACTIVE	Delivery address support unavailable
ERR_DELIVERY_ADDRESS_EMPTY_NAME_AND_COMPANY	Empty delivery address
ERR_ADDITIONAL_DELIVERY_ADDRESS_EMPTY_NAME_AND_COMPANY	Empty additional delivery address
ERR_INCOTERMS_INACTIVE	INCOTERMS code support unavailable
ERR_INCOTERMS_INVALID_CODE	Wrong INCOTERMS code
ERR_ADDITIONAL_INFO_INACTIVE	Additional information about product unavailable
ERR_PACKAGE_PRODUCT_INACTIVE	Product package unavailable
ERR_PACKAGE_PRODUCT_ID_INVALID	Wrong package product id
ERR_PAYMENT_EDIT_NOT_AVAILABLE	Editing payment type unavailable
ERR_PAYMENT_INVALID_TYPE	Wrong payment type
ERR_PAYMENT_TYPE_NOT_AVAILABLE	Selected payment type unavailable
ERR_ROUTE_INVALID	Selected route is invalid or isn't available
ERR_INVALID_CUSTOMER	Invalid customer
ERR_ITEM_NOT_FOUND	At least one product has not been found
ERR_INVALID_HANDOVERTODD	Handing over to goods issue note is not available
ERR_LOGM_SERVICE_TYPE_INVALID	Invalid logistic customer type
ERR_INVALID_FINAL_DOCUMENT_TYPE	Invalid final document type

Comments:

* One of these is required: "id", or "reference", or both "tecidd" and "tecnum", or “motonet”.
* In case of sending TecDoc or Motonet data - information about first fitting product is returned.
* Informations about products are returned in the same order as in request.
** This functionality can be limited

XML example:

```
<?xml version="1.0" encoding="utf-8" ?>
<doOrderProducts>
   <sessionId>2BZY8zSxODOOUmMOylFKSZjMsgDG564YxTMvju_D</sessionId>
   <forceNewOrder>1</forceNewOrder>
   <newOrderInfo>
      <payment>
         <type>5</type>
         <days>8</days>
      </payment>
   </newOrderInfo>
   <productOrderList>
      <productOrder>
         <reference>0-001-231-030</reference>
         <quantity>2</quantity>
      </productOrder>
      <productOrder>
         <reference>0-001-268-320</reference>
         <quantity>1</quantity>
      </productOrder>
      <productOrder>
         <reference>SYMBOL123</reference>
         <quantity>2</quantity>
      </productOrder>
   </productOrderList>
</doOrderProducts>

Response:
<?xml version="1.0" encoding="utf-8" ?>
<doOrderProductsResponse>
   <orderInfo>
      <id>00001269</id>
      <number>63</number>
   </orderInfo>
   <productOrderInfoList>
      <productOrderInfo>
         <id>003LFT</id>
         <reference>0-001-231-030</reference>
         <name>Osłona przegubu /BOSCH/</name>
         <decimals>2</decimals>
         <tax>22</tax>
         <warehouseId>XX</warehouseId>
         <unit>szt</unit>
         <priceExclTax>55.57</priceExclTax>
         <priceInclTax>67.79</priceInclTax>
         <currency>zł</currency>
         <quantity>2</quantity>
         <quantityBlocked>2</quantityBlocked>
         <quantityNotBlocked>0</quantityNotBlocked>
      </productOrderInfo>
      <productOrderInfo>
         <id>003LGF</id>
         <reference>0-001-268-320</reference>
         <name>Osłona przegubu wew. /BOSCH/</name>
         <decimals>0</decimals>
         <tax>22</tax>
         <warehouseId>XX</warehouseId>
         <unit>szt</unit>
         <priceExclTax>6111.11</priceExclTax>
         <priceInclTax>7455.56</priceInclTax>
         <currency>zł</currency>
         <quantityBlocked>0</quantityBlocked>
         <quantityNotBlocked>1</quantityNotBlocked>
      </productOrderInfo>
      <productOrderInfo>
         <error>brak kartoteki w magazynie</error>
      </productOrderInfo>
   </productOrderInfoList>
</doOrderProductsResponse>
```

JSON example:

```
{
  "doOrderProducts": {
    "sessionId": "2BZY8zSx0D0OUmM0ylFKSZjMsgDG564YxTMvju_D",
    "productOrderList": {
      "productOrder": [
        {
          "reference": "0-001-231-030",
          "quantity": "2"
        },
        {
          "reference": "0-001-268-320",
          "quantity": "1"
        },
        {
          "reference": "SYMBOL123",
          "quantity": "2"
        }
     ]
    }
  }
}

Response:
{ "doOrderProductsResponse": {
    "orderInfo": {
      "id": "00001269",
      "number": "63"
    },
    "productOrderInfoList": {
      "productOrderInfo": [
         {
          "id": "003LFT",
          "reference": "0-001-231-030",
          "name": "Osłona przegubu /BOSCH/",
          "decimals": "2",
          "tax": "22",
          "warehouseId": "XX",
          "unit": "szt",
          "priceExclTax": "55.57",
          "priceInclTax": "67.79",
          "currency": "zł",
          "quantity": "2",
          "quantityBlocked": "2",
          "quantityNotBlocked": "0"
         },
         {
          "id": "003LGF",
          "reference": "0-001-268-320",
          "name": "Osłona przegubu wew. /BOSCH/",
          "decimals": "0",
          "tax": "22",
          "warehouseId": "XX",
          "unit": "szt",
          "priceExclTax": "6111.11",
          "priceInclTax": "7455.56",
          "currency": "zł",
          "quantity": "1",
          "quantityBlocked": "0",
          "quantityNotBlocked": "1"
         ,
        { "error": "brak kartoteki w magazynie" }
      ]
    }
  }
}
```

`getMyOrders`

Get order list for logged user

Parameters:

`1 sessionId`	`(string)`	Session id obtained with doLogin method
`2(?) orderIdList`	`(string[])`	List of orders
`   1 id`	`(string)`	Order id
`   2 id …`		
`3(?) filterOptions`	`(struct)`	Orders filters
`   1(?) externalId`	`(string)`	The customer order number
`   2(?) isReward`	`(int)`	Filtering orders by tags “bring out reward” (1 – tag is set to yes, 0 – tag is set to no, no default filter)
`4(?) getItems`	`(int)`	Return information about items (1-yes, 0-no; default 0)
`5(?) getLoyaltyInfo`	`(int)`	Return data of loyalty system (1 – yes, 0 – no (default))

Return:

`1 orderList`	`(struct[])`	Order list
`   1 order`	`(struct)`	Information about order
`      1 id`	`(string)`	Document id
`      2 number`	`(string)`	Document number
`      3 justNumber`	`(string)`	Document number without postscript
`      4 externalId`	`(string)`	External order number
`      5 type`	`(string)`	Document type: WZ – invoice, DD – goods issue note, ZW – invoice correction
`      6 warehouseId`	`(string)`	Warehouse id
`      7 date`	`(date)`	[format: yyyy-mm-dd] Document date
`      8 dateExpire`	`(date)`	[format: yyyy-mm-dd] Date of expire
`      9 dateRealization`	`(date)`	[format: yyyy-mm-dd] Date of realization
`      10 isClosed`	`(int)`	Is order closed (1-yes, 0-no)
`      11 additionalInfo`	`(string)`	Additional information for the document
`      12 dateAddInfo`	`(date)`	Date of editing additional information to the document
`      13 recipientId`	`(string)`	Receiver id
`      14 recipientName`	`(string)`	Receiver name
`      15 recipientShortName`	`(string)`	Abbreviated receiver name
`      16 isDoReturnNow`	`(int)`	Is document created with the doReturnNow method (1 – yes, 0 - no)
`      17 comment`	`(string)`	Comment
`      18 currency`	`(string)`	Currency
`      19 localCurrency`	`(string)`	Local currency
`      20 currencyRate`	`(float)`	Currency rate
`      21 exportType`	`(int)`	Type of bill (0-country, 1-export, 2-country in currency)
`      22 isTaxLC`	`(int)`	Is tax in local currency (1) or document currency (0)
`      23 isValueLC`	`(int)`	Is value in local currency (1) or document currency (0)
`      24 valueExclTaxLC`	`(float)`	Price without tax in local currency
`      25 valueInclTaxLC`	`(float)`	Price with tax in local currency
`      26 valueTaxLC`	`(float)`	Tax amount in local currency
`      27 valueExclTax`	`(float)`	Value without tax
`      28 valueInclTax`	`(float)`	Value with tax
`      29 valueTax`	`(float)`	Tax value
`      30 isEU`	`(int)`	EU document (1-yes, 0-no)
`      31 isReExport`	`(int)`	Is document re-exported (1-yes, 0-no)
`      32 paymentType`	`(int)`	Payment type: 1-none, 2-cash, 3-cash in term, 4-check, 5-bank wire, 6-cash/bank wire, 7-COD, 8-credit card, 9-installment
`      33 paymentDays`	`(float)`	Payment term (days)
`      34 isPrinted`	`(int)`	Was the document printed (1-yes, 0-no)
`      35 wmsStatus`	`(int)`	Issue status of the document (0 – not issued, 1 – issue due, 2 – issue complete)

36 paymentStatus	(int)	Payment status (0 – unkown, 1 – payed, 2 – not payed, 3 – not payed and overdue)
30 isEU	(int)	EU document (1-yes, 0-no)
37 loyaltyPoints	(int)	Number of points earned in loyalty system (only if getLoyaltyInfo = 1)
38 loyaltyRewardPoints	(int)	Number of points that will be deducted from client account after “bring out reward” (return only if getLoyaltyInfo = 1)
39 isDeliveryConfirmed	(int)	Is delivery confirmed (1 – yes, 0 – no)
40 isReward	(int)	Content of tag “bring out reward” (0 – no, 1 – yes)
41 routeId	(string)	Route id
42 isFromDeficiencies	(int)	Is order created because of deficiencies (0 – no, 1 – yes)
43 isExternalOrderInProgress	(int)	Is external order currently in progress (0 – no, 1 - yes)
44 itemsCount	(int)	Number of items in order
45 items	(struct[])	List of order items (only if `getItems` = 1)
1 item	(struct)	Information about document item
1 id	(string)	Item id
2 ordinal	(int)	Item position in document
3 productId	(string)	Product id
4 productType	(int)	Product type (0-article, 1-service)
5 productReference	(string)	Product reference
6 productName	(string)	Name of product in user language
7 (?) motonet	(string)	Motonet number
8 decimals	(int)	Decimal places (0-3)
9 tax	(float)	Tax rate
10 quantity	(float)	Quantity ordered
11 quantityBlocked	(float)	Quantity blocked in warehouse
12 isExternal	(int)	Is item ordered in external warehouse (1/0)
13 externalInfo	(string)	Information about delivery time
14 isDeposit	(int)	Is item a deposit (1/0)
15 unit	(string)	Unit
16 priceExclTax	(float)	Price without tax
17 valueExclTax	(float)	Value without tax
18 priceInclTax	(float)	Price with tax
19 valueInclTax	(float)	Value with tax
20 valueTax	(float)	Tax value
21 priceExclTaxLC	(float)	Price without tax in local currency
22 valueExclTaxLC	(float)	Value without tax in local currency
23 priceInclTaxLC	(float)	Price with tax in local currency
24 valueInclTaxLC	(float)	Value with tax in local currency
25 valueTaxLC	(float)	Tax value in local currency
26 currency	(string)	Currency
27 discountPercent	(float)	Discount %
28 startingPriceExclTax	(float)	Price without tax before discount
29 startingPriceInclTax	(float)	Price with tax before discount
30 (?) priceCatExclTax	(float)	Catalogue price without tax in document currency
31 (?) priceCatInclTax	(float)	Catalogue price with tax in document currency
32 packageType	(int)	Package type: 0 – original package, 1 – customer’s package, 2 – none (only if package types are enabled in the system)
33 customerReference	(string)	Customer reference (only if enabled in the system)
34 finalDocumentType	(int)	The final type of the document (0 – default, 1 – invoice, 2 – receipt)
35 loyaltyRewardPoints	(int)	Number of points that will be deducted from client account after “bring out reward” (return only if getLoyaltyInfo = 1)
2 item ...		
2 order ...		

2 ordersCount	(int)	Number of orders (can be approximate)

Error codes:

none

XML example:

```
<?xml version="1.0" encoding="utf-8" ?>
<getMyOrders>
   <sessionId>V5etQUyinDnHosbSzjy7iLdj3Us4GaEr5ILMt9_D</sessionId>
   <getItems>1</getItems>
</getMyOrders>

Response:
<?xml version="1.0" encoding="utf-8" ?>
<getMyOrdersResponse>
   <orderList>
      <order>
         <id>01520531</id>
         <number>RE 11426</number>
         <warehouseId>CE</warehouseId>
         <date>2013-01-01</date>
         <dateExpire>2013-01-01</dateExpire>
         <dateRealization>2013-01-07</dateRealization>
         <isClosed>0</isClosed>
         <comment>testowe zamówienie</comment>
         <currency>zł</currency>
         <localCurrency>zł</localCurrency>
         <currencyRate>0</currencyRate>
         <exportType>0</exportType>
         <valueExclTax>100.00</valueExclTax>
         <valueInclTax>123.00</valueInclTax>
         <valueTax>23.00</valueTax>
         <itemsCount>1</itemsCount>
         <items>
            <item>
               <id>R-3975485</id>
               <ordinal>1</ordinal>
               <productId>105275</productId>
               <productType>0</productType>
               <productReference>123-456-789</productReference>
               <productName>Filtr kabiny VAG</productName>
               <decimals>0</decimals>
               <tax>23</tax>
               <quantity>2.000</quantity>
               <quantityBlocked>1.000</quantityBlocked>
               <isExternal>0</isExternal>
               <externalInfo></externalInfo>
               <isDeposit>0</isDeposit>
               <unit>szt</unit>
               <priceExclTax>50.00</priceExclTax>
               <priceInclTax>61.50</priceInclTax>
               <valueExclTax>100.00</valueExclTax>
               <valueInclTax>123.00</valueInclTax>
               <valueTax>23.00</valueTax>
               <currency>zł</currency><discountPercent>50.0</discountPercent>
               <startingPriceExclTax>100.00</startingPriceExclTax>
               <startingPriceInclTax>123.00</startingPriceInclTax>
            </item>
         </items>
      </order>
```

```
    </orderList>
    <ordersCount>1</ordersCount>
</getMyOrdersResponse>
```

JSON example:

```
{
  "getMyOrders": {
    "sessionId": "V5etQUyinDnHosbSzjy7iLdj3Us4GaEr5ILMt9_D",
    "getItems": "1"
  }
}

Response:
{
  "getMyOrdersResponse": {
    "orderList": {
      "order": {
        "id": "01520531",
        "number": "RE 11426",
        "warehouseId": "CE",
        "date": "2013-01-01",
        "dateExpire": "2013-01-01",
        "dateRealization": "2013-01-07",
        "isClosed": "0",
        "comment": "testowe zamówienie",
        "currency": "zł",
        "localCurrency": "zł",
        "currencyRate": "0",
        "exportType": "0",
        "valueExclTax": "100.00",
        "valueInclTax": "123.00",
        "valueTax": "23.00",
        "itemsCount": "1",
        "items": {
          "item": {
            "id": "R-3975485",
            "ordinal": "1",
            "productId": "105275",
            "productType": "0",
            "productReference": "123-456-789",
            "productName": "Filtr kabiny VAG",
            "decimals": "0",
            "tax": "23",
            "quantity": "2.000",
            "quantityBlocked": "1.000",
            "isExternal": "0",
            "isDeposit": "0",
            "unit": "szt",
            "priceExclTax": "50.00",
            "priceInclTax": "61.50",
            "valueExclTax": "100.00",
            "valueInclTax": "123.00",
            "valueTax": "23.00",
            "currency": "zł",
            "discountPercent": "50.0",
            "startingPriceExclTax": "100.00",
            "startingPriceInclTax": "123.00"
          }
        }
      }
    },
    "ordersCount": "1"
  }
}
```

`getOrderStockInfo`

Get information about availability of order items

Parameters:

1 sessionId	(string)	Session id obtained with doLogin method
2 orderId	(string)	Order id

Return value:

1 orderId	(string)	Order id
2 itemsCount	(int)	Number of items
3 items	(struct[])	List of items
1 item	(struct)	Information about item
1 id	(string)	Item id
2 ordinal	(int)	Item position in document
3 productId	(string)	Product id
4 productType	(int)	Product type (0-article, 1-service)
5 productReference	(string)	Product reference
6 productName	(string)	Product name in user language
7 decimals	(int)	Decimal places (0-3)
8 quantity	(float)	Quantity ordered
9 quantityInStock	(float)	Quantity available
10 stockStatus	(int)	Availability status (0-in warehouse, 1-partial external availability, 2-full external availability, "-1" product external unavailable)
11 stockInfo	(string)	Text info about availability
12 unit	(string)	Unit
2 item ...		

Error codes:

ERR_WRONG_ORDER_ID	Wrong order id
ERR_ORDER_ALREADY_CLOSED	Order is already closed
ERR_ORDER_LOCKED	Order is locked by other user

XML example:

```
<?xml version="1.0" encoding="utf-8" ?>
<getOrderStockInfo>
   <sessionId>l5LADxWUF9w4RFXoj7rMsOdiGVTApt3AoE1S1O_D</sessionId>
   <orderId>01512768</orderId>
</getOrderStockInfo>

Response:
<?xml version="1.0" encoding="utf-8" ?>
<getOrderStockInfoResponse>
   <orderId>01512768</orderId>
   <itemsCount>10</itemsCount>
   <items>
      <item>
         <id>R-3899087</id>
         <ordinal>1</ordinal>
         <productId> 40458</productId>
         <productType>0</productType>
         <productReference>11.5855</productReference>
         <productName>Linka ham.OPEL ASTRA /COFLE/ kombi</productName>
         <decimals>0</decimals>
         <quantity>3</quantity>
         <quantityInStock>3</quantityInStock>
         <stockStatus>0</stockStatus>
         <stockInfo>w magazynie</stockInfo>
         <unit>szt</unit>
      </item>
      <item>
         <id>R-3899089</id>
         <ordinal>2</ordinal>
         <productId>000UAN</productId>
         <productType>0</productType>
         <productReference>MD9044</productReference>
         <productName>
            Zawór zwrotny paliwa 6mm - uniwersalny
         </productName>
         <decimals>0</decimals>
         <quantity>3</quantity>
         <quantityInStock>0</quantityInStock>
         <stockStatus>2</stockStatus>
         <stockInfo>Pełna (od 2 dni)</stockInfo>
         <unit>szt</unit>
      </item>
      <item>
         <id>R-3899090</id>
         <ordinal>3</ordinal>
         <productId>0009VJ</productId>
         <productType>0</productType>
         <productReference>54624 GL</productReference>
         <productName>
            Uszcz.miski olej.VAG 1.6-2.0 93- /GLASER/
         </productName>
         <decimals>0</decimals>
         <quantity>4</quantity>
         <quantityInStock>0</quantityInStock>
         <stockStatus>1</stockStatus>
         <stockInfo>1 szt (1 - 2 dni)</stockInfo>
         <unit>szt</unit>
      </item>
      <item>
         <id>R-3899091</id>
         <ordinal>4</ordinal>
         <productId>141129</productId>
         <productType>0</productType>
         <productReference>11.721.800</productReference>
         <productName>
            Świeca żarowa HONDA,RENAULT,ROVER 2.2DT /ISKRA/
         </productName>
         <decimals>0</decimals>
         <quantity>4</quantity>
         <quantityInStock>0</quantityInStock>
         <stockStatus>-1</stockStatus>
         <stockInfo>towar niedostępny</stockInfo>
         <unit>szt</unit>
      </item>
   </items>
</getOrderStockInfoResponse>
```

JSON example:

```
{
  "getOrderStockInfo": {
    "sessionId": "l5LADxWUF9w4RFXoj7rMsOdiGVTApt3AoE1S1O_D",
    "orderId": "01512768"
  }
}

Response:
{
  "getOrderStockInfoResponse": {
    "orderId": "01512768",
    "itemsCount": "10",
    "items": {
      "item": [
        {
          "id": "R-3899087",
          "ordinal": "1",
          "productId": " 40458",
          "productType": "0",
          "productReference": "11.5855",
          "productName": "Linka ham.OPEL ASTRA /COFLE/ kombi",
          "decimals": "0",
          "quantity": "3",
          "quantityInStock": "3",
          "stockStatus": "0",
          "stockInfo": "w magazynie",
          "unit": "szt"
        },
        {
          "id": "R-3899089",
          "ordinal": "2",
          "productId": "000UAN",
          "productType": "0",
          "productReference": "MD9044",
          "productName": "Zawór zwrotny paliwa 6mm - uniwersalny",
          "decimals": "0",
          "quantity": "3",
          "quantityInStock": "0",
          "stockStatus": "2",
          "stockInfo": "Pełna (od 2 dni)",
          "unit": "szt"
        },
        {
          "id": "R-3899090",
          "ordinal": "3",
          "productId": "0009VJ",
          "productType": "0",
          "productReference": "54624 GL",
          "productName": "Uszcz.miski olej.VAG 1.6-2.0 93- /GLASER/",
          "decimals": "0",
          "quantity": "4",
          "quantityInStock": "0",
          "stockStatus": "1",
          "stockInfo": "1 szt (1 - 2 dni)",
          "unit": "szt"
        },
        {
          "id": "R-3899091",
          "ordinal": "4",
          "productId": "141129",
          "productType": "0",
          "productReference": "11.721.800",
          "productName": "Świeca żarowa HONDA,RENAULT,ROVER 2.2DT /ISKRA/",
          "decimals": "0",
          "quantity": "4",
          "quantityInStock": "0",
          "stockStatus": "-1",
          "stockInfo": "towar niedostępny",
          "unit": "szt"
        }
      ]
    }
  }
}
```

doOrderClose

Close order

Parameters:

1 sessionId	(string)	Session id obtained with doLogin method
2 orderId	(string)	Order id
3 (?) collectionInPerson	(int)	Set delivery method to “personal collection”
4 (?) routeId	(string)	Identyfikator trasy

Return value:

1 orderId	(string)	Order id

Error codes:

ERR_WRONG_ORDER_ID	Wrong order id
ERR_ORDER_ALREADY_CLOSED	Order is already closed
ERR_ORDER_LOCKED	Order is locked by other user
ERR_CANT_CLOSE_ORDER	Closing failed
ERR_COLLECTION_IN_PERSON_UNAVAILABLE	Collection in person is not available
ERR_CIP_ROUTE	Only one of the two fields may be used at the same time: collectionInPerson, routeId.
ERR_ROUTE_UNAVAILABLE	Choosing route is unavailable.
ERR_WRONG_ROUTE_ID	Incorrent route.
ERR_BLOCKED_CONFIRM_RECEIPT_DELIVERY	Delivery confirmation is required.
ERR_SALE_BLOCK	Order can't be closed. Sale locked
ERR_ISSUE_BLOCK	Order can't be closed. Vouchers locked
ERR_ISSUES_OVERDUE	Order can't be closed. Vouchers term exceeded
ERR_PAYMENT_OVERDUE	Order can't be closed. Payment term exceeded
ERR_CREDIT_LIMIT	Order can't be closed. Credit limit exceeded

XML example:

```
<?xml version="1.0" encoding="utf-8" ?>
      <doOrderClose>
         <sessionId>V5etQUyinDnHosbSzjy7iLdj3Us4GaEr5ILMt9_D</sessionId>
         <orderId>01520531</orderId>
      </doOrderClose>

   Response:
      <?xml version="1.0" encoding="utf-8" ?>
      <doOrderCloseResponse>
         <orderId>01520531</orderId>
      </doOrderCloseResponse>
```

JSON example:

```
{
  "doOrderClose": {
    "sessionId": "V5etQUyinDnHosbSzjy7iLdj3Us4GaEr5ILMt9_D",
    "orderId": "01520531",
    "collectionInPerson": "1"
  }
}

Response:
{
  "doOrderCloseResponse": { "orderId": "01520531" }
}
```

`doOrderItemDelete`

Delete order item

Parameters:

1 sessionId	(string)	Session id obtained with doLogin method
2 orderId	(string)	Order id
3 itemId	(string)	Order item id

Return:

1 itemId	(string)	Deleted item id
2 orderDeleted	(int)	Blank order deleted (1-yes, 0-no)

Error code:

ERR_WRONG_ORDER_ID	Wrong order id
ERR_ORDER_CLOSED	Order is closed
ERR_ORDER_LOCKED	Order is locked by other user
ERR_WRONG_ITEM_ID	Wrong item id
ERR_ITEM_IS_EXTERNAL	Can't delete external ordered item
ERR_ITEM_IS_DEPOSIT	Can't delete deposit
ERR_CANT_DELETE_ITEM	Deleting failed
ERR_SALE_BLOCK	Order can't be closed. Sale locked
ERR_ISSUE_BLOCK	Order can't be closed. Vouchers locked
ERR_ISSUES_OVERDUE	Order can't be closed. Vouchers term exceeded
ERR_PAYMENT_OVERDUE	Order can't be closed. Payment term exceeded
ERR_CREDIT_LIMIT	Order can't be closed. Credit limit exceeded

XML example:

```
<?xml version="1.0" encoding="utf-8" ?>
<doOrderItemDelete>
   <sessionId>V5etQUyinDnHosbSzjy7iLdj3Us4GaEr5ILMt9_D</sessionId>
   <orderId>01520531</orderId>
   <itemId>R-3975485</itemId>
</doOrderItemDelete>

Response:
<?xml version="1.0" encoding="utf-8" ?>
<doOrderItemDeleteResponse>
   <itemId>R-3975485</itemId>
   <orderDeleted>0</orderDeleted>
</doOrderItemDeleteResponse>
```

JSON example:

```
{
  "doOrderItemDelete": {
    "sessionId": "V5etQUyinDnHosbSzjy7iLdj3Us4GaEr5ILMt9_D",
    "orderId": "01520531",
    "itemId": "R-3975485"
  }
}

Response:
{
  "doOrderItemDeleteResponse": {
    "itemId": "R-3975485",
    "orderDeleted": "0"
  }
}
```

doOrderItemEdit

Edit order item

Parameters:

1 sessionId	(string)	Session id obtained with doLogin method
2 orderId	(string)	Order id
3 itemId	(string)	Order item id
4 itemInfo	(struct)	Item information
1(?) quantity	(float)	Quantity ordered
2(?) finalDocumentType	(int)	Final document type, "convert direction" (0-default, 1-invoice, 2-bill)
3(?) customerReference	(string)	Customer reference
4(?) packageType	(int)	Package type (0-own, 1-client, 2-none, default 0)
5(?) description	(string)	Item description

Return:

1 item	(struct)	Item information
1 id	(string)	Item id
2 ordinal	(int)	Item position in document
3 productId	(string)	Product id
4 productType	(int)	Product type (0-article, 1-service)
5 productReference	(string)	Product reference
6 productName	(string)	Product name in user language
7 decimals	(int)	Decimal places (0-3)
8 tax	(float)	Tax rate
9 quantity	(float)	Quantity ordered
10 quantityBlocked	(float)	Quantity blocked in warehouse
11 isExternal	(int)	Is item ordered in external warehouse (1/0)
12 externalInfo	(string)	Information about delivery time
13 isDeposit	(int)	Is item a deposit (1/0)
14 unit	(string)	Unit
15 priceExclTax	(float)	Price without tax
16 valueExclTax	(float)	Value without tax
17 priceInclTax	(float)	Price with tax
18 valueInclTax	(float)	Value with tax
19 valueTax	(float)	Tax value
20 priceExclTaxLC	(float)	Price without tax in local currency
21 valueExclTaxLC	(float)	Value without tax in local currency
22 priceInclTaxLC	(float)	Price with tax in local currency
23 valueInclTaxLC	(float)	Value with tax in local currency
24 valueTaxLC	(float)	Tax value in local currency
25 currency	(string)	Currency
26 discountPercent	(float)	Discount %
27 startingPriceExclTax	(float)	Price without tax before discount
28 startingPriceInclTax	(float)	Price with tax before discount
29 (?) priceCatExclTax	(float)	Catalogue price without tax in document currency
30 (?) priceCatInclTax	(float)	Catalogue price with tax in document currency

Error codes:

ERR_WRONG_ORDER_ID	Wrong order id
ERR_ORDER_CLOSED	Order is already closed
ERR_ORDER_LOCKED	Order is locked by other user
ERR_WRONG_ITEM_ID	Wrong item id
ERR_ITEM_IS_EXTERNAL	Can't edit external ordered item
ERR_ITEM_HAS_DEPOSIT	Can’t edit item with deposit
ERR_ITEM_IS_DEPOSIT	Can't edit deposit

`ERR_WRONG_QUANTITY`	Wrong quantity of ordered items
`ERR_CANT_EDIT_ITEM`	Edit failed
`ERR_SALE_BLOCK`	Order can't be closed. Sale locked
`ERR_ISSUE_BLOCK`	Order can't be closed. Vouchers locked
`ERR_ISSUES_OVERDUE`	Order can't be closed. Vouchers term exceeded
`ERR_PAYMENT_OVERDUE`	Order can't be closed. Payment term exceeded
`ERR_CREDIT_LIMIT`	Order can't be closed. Credit limit exceeded

XML example:

```
<?xml version="1.0" encoding="utf-8" ?>
<doOrderItemEdit>
   <sessionId>V5etQUyinDnHosbSzjy7iLdj3Us4GaEr5ILMt9_D</sessionId>
   <orderId>01520531</orderId>
   <itemId>R-3975485</itemId>
   <itemInfo>
      <quantity>2</quantity>
   </itemInfo>
</doOrderItemEdit>

Rsponse:
<?xml version="1.0" encoding="utf-8" ?>
<doOrderItemEditResponse>
   <item>
      <id>R-3975485</id>
      <ordinal>1</ordinal>
      <productId>105275</productId>
      <productType>0</productType>
      <productReference>123-456-789</productReference>
      <productName>Filtr kabiny VAG</productName>
      <decimals>0</decimals>
      <tax>23</tax>
      <quantity>2.000</quantity>
      <quantityBlocked>1.000</quantityBlocked>
      <isExternal>0</isExternal>
      <externalInfo></externalInfo>
      <isDeposit>0</isDeposit>
      <unit>szt</unit>
      <priceExclTax>50.00</priceExclTax>
      <priceInclTax>61.50</priceInclTax>
      <valueExclTax>100.00</valueExclTax>
      <valueInclTax>123.00</valueInclTax>
      <valueTax>23.00</valueTax>
      <currency>zł</currency>
      <discountPercent>50.0</discountPercent>
      <startingPriceExclTax>100.00</startingPriceExclTax>
      <startingPriceInclTax>123.00</startingPriceInclTax>
   </item>
</doOrderItemEditResponse>
```

JSON example:

```
{
  "doOrderItemEdit": {
    "sessionId": "V5etQUyinDnHosbSzjy7iLdj3Us4GaEr5ILMt9_D",
    "orderId": "01520531",
    "itemId": "R-3975485",
    "itemInfo": { "quantity": "2" }
  }
}

Rsponse:
{
  "doOrderItemEditResponse": {
    "item": {
      "id": "R-3975485",
      "ordinal": "1",
      "productId": "105275",
      "productType": "0",
      "productReference": "123-456-789",
      "productName": "Filtr kabiny VAG",
      "decimals": "0",
      "tax": "23",
      "quantity": "2.000",
      "quantityBlocked": "1.000",
      "isExternal": "0",
      "isDeposit": "0",
      "unit": "szt",
      "priceExclTax": "50.00",
      "priceInclTax": "61.50",
      "valueExclTax": "100.00",
      "valueInclTax": "123.00",
      "valueTax": "23.00",
      "currency": "zł",
      "discountPercent": "50.0",
      "startingPriceExclTax": "100.00",
      "startingPriceInclTax": "123.00"
    }
  }
}
```

doOrderEdit

Edit order parameters

Parameters:

1 sessionId	(string)	Session id obtained with doLogin method
2 orderId	(string)	Order id
3 orderInfo	(struct)	Information about order
1(?) comment	(string)	Comment
2(?) dateDelivery	(date)	Date of realization [format: yyyy-mm-dd]
3(?) additionalInfo	(string)	Additional information for the documentu

Return:

1 orderId	(string)	Order id

Error codes:

ERR_WRONG_ORDER_ID	Wrong order id
ERR_ORDER_ALREADY_CLOSED	Order is already closed
ERR_ORDER_LOCKED	Order is locked by other user
ERR_COMMENT_TOO_LONG	Comment is too long (max 30 chars)
ERR_CANT_CHANGE_ORDER	Order save failed
ERR_SALE_BLOCK	Order can't be closed. Sale locked
ERR_ISSUE_BLOCK	Order can't be closed. Vouchers locked
ERR_ISSUES_OVERDUE	Order can't be closed. Vouchers term exceeded
ERR_PAYMENT_OVERDUE	Order can't be closed. Payment term exceeded
ERR_CREDIT_LIMIT	Order can't be closed. Credit limit exceeded
ERR_ADDITIONAL_INFO_INACTIVE	Additional information about product unavailable

XML example:

```
<?xml version="1.0" encoding="utf-8" ?>
<doOrderEdit>
   <sessionId>V5etQUyinDnHosbSzjy7iLdj3Us4GaEr5ILMt9_D</sessionId>
   <orderId>01520531</orderId>
   <orderInfo>
      <comment>komentarz</comment>
   </orderInfo>
</doOrderEdit>

Response:
<?xml version="1.0" encoding="utf-8" ?>
<doOrderEditResponse>
   <orderId>01520531</orderId>
</doOrderEditResponse>
```

JSON example:

```
{
  "doOrderEdit": {
    "sessionId": "V5etQUyinDnHosbSzjy7iLdj3Us4GaEr5ILMt9_D",
    "orderId": "01520531",
    "orderInfo": { "comment": "komentarz" }
  }
}

Response:
{
  "doOrderEditResponse": { "orderId": "01520531" }
}
```

doOrderDelete

Delete order

Parameters:

1 sessionId	(string)	Session id obtained with doLogin method
2 orderId	(string)	Order id
3(?) deleteItems	(int)	Delete all items from odrder (1-yes, 0-no, default 0)

Return:

1 orderId	(string)	Order id

Error code:

ERR_WRONG_ORDER_ID	Wrong order id
ERR_METHOD_BLOCKED	Method blocked
ERR_ORDER_ALREADY_CLOSED	Order is already closed
ERR_ORDER_LOCKED	Order is locked by other user
ERR_ORDER_STILL_HAS_ITEMS	Order still has some items
ERR_CANT_DELETE_ORDER	Order deleting error

XML example:

```
<?xml version="1.0" encoding="utf-8" ?>
<doOrderDelete>
   <sessionId>V5etQUyinDnHosbSzjy7iLdj3Us4GaEr5ILMt9_D</sessionId>
   <orderId>01520531</orderId>
</doOrderDelete>

Response:
<?xml version="1.0" encoding="utf-8" ?>
<doOrderDeleteResponse>
   <orderId>01520531</orderId>
</doOrderDeleteResponse>
```

JSON example:

```
{
  "doOrderDelete": {
    "sessionId": "V5etQUyinDnHosbSzjy7iLdj3Us4GaEr5ILMt9_D",
    "orderId": "01520531"
  }
}

Response:
{
  "doOrderDeleteResponse": { "orderId": "01520531" }
}
```

doOrderItemsMove

Move items between orders

Parameters:

1 sessionId	(string)	Session id obtained with doLogin method
2 srcOrderId	(string)	Source order id
3 srcItemList	(struct[])	List of items to move
1 srcItem	(struct)	Item to move
1 itemId	(string)	Item id
2 quantity	(float)	Quantity to move
2 srcItem ...		
4(?) destOrderId	(string)	Destination order id
5 destOrderInfo	(struct)	Information about destination order
1(?) dateDelivery	(date)	Date of realization [format: yyyy-mm-dd]

Return value:

1 status	(int)	1 - success, 0 – failed
2(*) destOrderId	(string)	Destination order id

Error codes:

ERR_WRONG_ORDER_ID	Wrong source order id
ERR_WRONG_ORDERS	Wrong orders
ERR_WRONG_DEST_ORDER_ID	Wrong destination order id
ERR_WRONG_ITEM_ID	Wrong item id
ERR_ORDER_ALREADY_CLOSED	Order is already closed
ERR_ORDER_LOCKED	Order is locked by other user
ERR_DATEDELIVERY_WRONG_FORMAT	Wrong delivery date format
ERR_ORDER_CREATE	Error during creating new order

Uwagi:
* Information about destination order id will be returned if moving position will end succesfull.

XML example:

```
query:
   <?xml version="1.0" encoding="utf-8" ?>
   <doOrderItemsMove>
      <sessionId>V5etQUyinDnHosbSzjy7iLdj3Us4GaEr5ILMt9_D</sessionId>
      <srcOrderId>01520531</srcOrderId>
      <destOrderId>01520532</destOrderId>
   </doOrderItemsMove>

response:
  <?xml version="1.0" encoding="utf-8" ?>
  <doOrderItemsMoveResponse>
      <status>1</status>
  </doOrderItemsMoveResponse>
```

DOCUMENTS

`getMyInvoices`

Get list of sale documents of logged user

Parameters:

`1 sessionId`	`(string)`	Session id obtained with doLogin method
`2 finalDocuments`	`(int)`	Type of returned documents (0-vouchers, 1-invoices, 2-vouchers and invoices, 3-invoices corrections; default 2) ignored if given `documentIdList`
`3(?) documentIdList`	`(string[])`	List of documents to return
`   1 id`	`(string)`	Document id
`   2 id ...`		
`4(?) filterOptions`	`(struct)`	Document filters
`   1(?) dateFrom`	`(date)`	[format: yyyy-mm-dd] Begin of date range. By default, documents from last 7 days are returned (if parameter `documentIdList` was NOT given) or from any range (if parameter `documentIdList` was given)
`   2(?) dateTo`	`(date)`	[format: yyyy-mm-dd] End of date range (default: today)
`   3(?) paymentStatus`	`(string)`	Payment status (0-all, 1-only payed, 2-only not payed, 3-only not payed, out of date)
`   4(?) closeStatus`	`(string)`	Document status (0-all, 1-not confirmed, 2-confimed)
`   5(?) srcDocId`	`(string)`	Id source documet
`   6(?) externalId`	`(string)`	The customer order number
`   7(?) isDoReturnNow`	`(int)`	Return only documents which were created by doReturnNow (1-yes, 0-no)
`   8(?) deliveryConfirmationStatus`	`(int)`	Delivery confirmation status (1-confirmed, 2-unconfirmed)
`   9(?) wmsStatus`	`(int)`	Filter by document (0 – nohow, 1 – pass to delivery, 2 – to deliver)
`   10(?) isReward`	`(int)`	Filtering orders by tags “bring out reward” (1 – tag is set to yes, 0 – tag is set to no, no default filter)
`   11(?) productIdList`	`(string[])`	List of products of which at least one needs to be in the document
`      1 id`	`(string)`	Product id
`      2 id ...`		
`5(?) sortOptions`	`(struct)`	Sorting options
`   1(?) sortType`	`(int)`	Sort type (1-by date of create (default))
`   2(?) sortOrder`	`(int)`	Sort order (1-ascending, 2-descending (default))
`6(?) getItems`	`(int)`	Return information about items (1-yes, 0-no; default 0)
`7(?) getLoyaltyInfo`	`(int)`	Return data of loyalty system (1 - yes, 0 - no; default 0)
`8(?) getTrackingNrList`	`(int)`	Return data about list? (1 – return trackingNrList, 0 – don’t return)
`9(?) pageSize`	`(int)`	Number of documents on page (min: 1, max: 50, default 10)
`10(?) pageNumber`	`(int)`	Page number (default: 1)

Return value:

`1 invoiceList`	`(struct[])`	List of documents
`   1 invoice`	`(struct)`	Information about document
`      1 id`	`(string)`	Document id
`      2 number`	`(string)`	Document number
`      3 justNumber`	`(string)`	Document number without postscript
`      4 externalId`	`(string)`	Number of external order

5 type	(string)	Document type: WZ-invoice, DD-goods issue note, ZW-invoice correction
6 warehouseId	(string)	Warehouse id
7 date	(date)	[format: yyyy-mm-dd] Document date
8 isClosed	(int)	Document status (1-confirmed, 0- not confirmed)
9 recipientId	(string)	Recipient id
10 recipientName	(string)	Recipient name
11 recipientShortName	(string)	Short name of recipient
12 dateSale	(date)	[format: yyyy-mm-dd] Sale date
13 isFinalDocument	(int)	Type of document: 1-invoice, 0-voucher
14 isDoReturnNow	(int)	Return only documents which were created by doReturnNow (1-yes, 0-no)
15 comment	(string)	Comment
16 currency	(string)	Currency
17 localCurrency	(string)	Local currency
18 currencyRate	(float)	Currency rate
19 exportType	(int)	Invoice type: 0 - country, 1 - export, 2 – country in currency
20 isTaxLC	(int)	Is tax in local currency (1) or document currency (0)
21 IsValueLC	(int)	Is value in local currency (1) or document currency (0)
22 valueExclTaxLC	(float)	Value without tax in local currency
23 valueInclTaxLC	(float)	Value with tax in local currency
24 valueTaxLC	(float)	Tax value in local currency
25 valueExclTax	(float)	Value without tax
26 valueInclTax	(float)	Value with tax
27 valueTax	(float)	Tax value
28 isEU	(int)	EU document (1-yes, 0-no)
29 isReExport	(int)	Is document re-exported (1-yes, 0-no)
30 paymentType	(string)	Payment type: 1-none, 2-cash, 3-cash in term, 4-check, 5-bank wire, 6-cash/bank wire, 7-COD, 8-credit card, 9-installment
31 paymentDays	(int)	Payment term (in days)
32 isEDocument	(int)	Electronic version of document exist (1-yes, 0-no)
33 eDocumentStatus	(int)	Electronic version status: (0-no information, 1-created, 2-sent)
34 isPrinted	(int)	Printed document (1 – yes, 0 – no)
35 wmsStatus	(int)	Document WMS status (0-none, 1-given to release, 2-released)
36 paymentStatus	(int)	Payment status (0-unknown status, 1-settled, 2-not settled, 3-not settled, out of date)
37 loyaltyPoints	(int)	Number of points earned in loyalty system (only if getLoyaltyInfo = 1)
38 loyaltyRewardPoints	(int)	Number of points that will be decreased after given reward (returned only if getLoyaltyInfo = 1)
39 isDeliveryConfirmed	(int)	Is delivery confirmed (1 - yes, 0 - no)
40 isReward	(int)	Content of tag “bring out reward” (0 – no, 1 - yes)
41 routeId	(string)	Route id
42 trackingNrList	(struct[])	List of tracking numbers
1 item	(string)	Tracking number
2 item ...		
43 itemsCount	(int)	Number of items in document
44 items	(struct[])	List of document items (only if `getItems` = 1)
1 item	(struct)	Information about item
1 id	(string)	Item id
2 ordinal	(int)	Item position in document
3 productId	(string)	Product id
4 productType	(int)	Product type (0-article, 1-service)
5 productReference	(string)	Product reference

```
            6 productName        (string)          Product name in user language
            7 country            (string)          Product country
            8 decimals           (int)             Decimal places (0-3)
            9 tax                (float)           Tax rate
            10 quantity          (float)           Quantity (for invoice corrections: quantity after
                                                   correction)
            11 unit              (string)          Unit
            12 priceExclTax      (float)           Price without tax
            13 valueExclTax      (float)           Value without tax
            14 priceInclTax      (float)           Price with tax
            15 valueInclTax      (float)           Value with tax
            16 valueTax          (float)           Tax value
            17 priceExclTaxLC    (float)           Price without tax in local currency
            18 valueExclTaxLC    (float)           Value without tax in local currency
            19 priceInclTaxLC    (float)           Price with tax in local currency
            20 valueInclTaxLC    (float)           Value with tax in local currency
            21 valueTaxLC        (float)           Tax value in local currency
            22 currency          (string)          Currency
            23 discountPercent   (float)           Discount %
            24 startingPriceExclTax (float)        Price without tax before discount
            25 startingPriceInclTax (float)        Price with tax before discount
            26 (?) priceCatExclTax  (float)        Catalogue price without tax in document currency
            27 (?) priceCatInclTax  (float)        Catalogue price with tax in document currency
            28 loyaltyPoints     (int)             Number of points earned in loyalty system
                                                   (only if getLoyaltyInfo = 1)
            29 loyaltyRewardPoints  (int)          Number of points that will be decreased after given
                                                   reward (returned only if getLoyaltyInfo = 1)
            30 weight_net        (float)           Net weight
            31 weight_gross      (float)           Gross weight
            32 customs_code      (string)          Customs code
            33 (?) correctedItem (struct)          Information about corrected item (for invoice
                                                   corrections)
               1 quantity        (float)           Quantity before correction
               2 priceExclTax    (float)           Price without tax
               3 valueExclTax    (float)           Value without tax
               4 priceInclTax    (float)           Price with tax
               5 valueInclTax    (float)           Value with tax
               6 valueTax        (float)           Tax value
               7 priceExclTaxLC  (float)           Price without tax in local currency
               8 valueExclTaxLC  (float)           Value without tax in local currency
               9 priceInclTaxLC  (float)           Price with tax in local currency
               10 valueInclTaxLC (float)           Value with tax in local currency
               11 valueTaxLC     (float)           Tax value in local currency
               12 discountPercent (float)          Discount %
               13 startingPriceExclTax (float)     Price without tax before discount
               14 startingPriceInclTax (float)     Price with tax before discount
            34** pieceList       (struct[])        List information about piece
               1 piece           (struct)          Piece information
                  1 imei         (string)          IMEI number
                  2 sn           (string)          Serial number
               2 piece ...
         2 item …
      2 invoice ...
   2 invoicesCount               (int)             Number of documents that meets the criteria
                                                   value can be approximated
```

Error codes:

```
      ERR_WRONG_FINALDOCUMENTS_TYPE                Wrong document type
```

`ERR_NOT_ALLOWED_FINALDOCUMENTS_TYPE`	Forbidden document type

Comments:

Metod returns documents from 7 days, unless `dateFrom`, `dateTo` (dates range) or `documentIdList` (list of documents IDs) is passed.

** Information about piece returns only for products with specified parameters.

XML example

```
<?xml version="1.0" encoding="utf-8" ?>
<getMyInvoices>
   <sessionId>
      2BZY8zSxODOOUmMOylFKSZjMsgDG564YxTMvju_D
   </sessionId>
   <documentIdList>
      <id>00001277</id>
      <id>23456789</id>
   </documentIdList>
   <filterOptions>
      <dateFrom>2013-03-01</dateFrom>
      <dateTo>2013-03-07</dateTo>
      <webCatUserId>012932</webCatUserId>
   </filterOptions>
   <sortOptions>
      <sortType>1</sortType>
      <sortOrder>1</sortOrder>
   </sortOptions>
   <pageSize>50</pageSize>
   <pageNumber>1</pageNumber>
   <getItems>1</getItems>
</getMyInvoices>

Response:
<?xml version="1.0" encoding="utf-8" ?>
<getMyInvoicesResponse>
   <invoiceList>
      <invoice>
         <id>00001277</id>
         <number>FV/1/01/CE/2009</number>
         <warehouseId>CE</warehouseId>
         <dateIssue>2013-03-04</dateIssue>
         <dateSale>2013-03-04</dateSale>
         <isFinalDocument>1</isFinalDocument>
         <comment />
         <currency>zł</currency>
         <localCurrency>zł</localCurrency>
         <currencyRate>0</currencyRate>
         <exportType>0</exportType>
         <valueExclTax>1040.44</valueExclTax>
         <valueInclTax>1269.34</valueInclTax>
         <valueTax>228.90</valueTax>
         <isEDocument>0</isEDocument>
```

```
         <itemsCount>2</itemsCount>
         <items>
            <item>
               <id>R-8239</id>
               <ordinal>1</ordinal>
               <productId>003WUL</productId>
               <productType>0</productType>
               <productReference>0-092-S40-280</productReference>
               <productName>Akumulator 95AH/830A /BOSCH </productName>
               <decimals>0</decimals>
               <tax>22</tax>
               <quantity>1.000</quantity>
               <unit>szt</unit>
               <priceExclTax>819.67</priceExclTax>
               <valueExclTax>819.67</valueExclTax>
               <priceInclTax>1000.00</priceInclTax>
               <valueInclTax>1000.00</valueInclTax>
               <valueTax>180.33</valueTax>
               <currency>zł</currency>
               <discountPercent>0.0</discountPercent>
               <startingPriceExclTax>819.67</startingPriceExclTax>
               <startingPriceInclTax>1000.00</startingPriceInclTax>
            </item>
            <item>
                           ...
            </item>
         </items>
      </invoice>
      <invoice>
      ...
      </invoice>
   </invoiceList>
</getMyInvoicesResponse>
```

JSONexample:

```
{
  "getMyInvoices": {
    "sessionId": "2BZY8zSxODOOUmMOylFKSZjMsgDG564YxTMvju_D",
    "documentIdList": {
      "id": [
        "00001277",
        "23456789"
      ]
    },
    "filterOptions": {
      "dateFrom": "2013-03-01",
      "dateTo": "2013-03-07",
      "webCatUserId": "012932"
    },
    "sortOptions": {
      "sortType": "1",
      "sortOrder": "1"
    },
    "pageSize": "50",
    "pageNumber": "1",
    "getItems": "1"
  }
}

Response:
{
  "getMyInvoicesResponse": {
    "invoiceList": {
      "invoice": [
        {
          "id": "00001277",
          "number": "FV/1/01/CE/2009",
          "warehouseId": "CE",
          "dateIssue": "2013-03-04",
          "dateSale": "2013-03-04",
          "isFinalDocument": "1",
          "currency": "zł",
          "localCurrency": "zł",
          "currencyRate": "0",
          "exportType": "0",
          "valueExclTax": "1040.44",
          "valueInclTax": "1269.34",
          "valueTax": "228.90",
          "isEDocument": "0",
          "itemsCount": "2",
          "items": {
            "item": [
              {
                "id": "R-8239",
                "ordinal": "1",
                "productId": "003WUL",
                "productType": "0",
                "productReference": "0-092-S40-280",
                "productName": "Akumulator 95AH/830A /BOSCH ",
                "decimals": "0",
                "tax": "22",
                "quantity": "1.000",
                "unit": "szt",
                "priceExclTax": "819.67",
                "valueExclTax": "819.67",
                "priceInclTax": "1000.00",
                "valueInclTax": "1000.00",
                 "valueTax": "180.33",
                "currency": "zł",
                "discountPercent": "0.0",
                "startingPriceExclTax": "819.67",
                "startingPriceInclTax": "1000.00"
              },
              item2...
            ]
          }
        },
        invoice2...
      ]
    }
  }
}
```

getEDocument

Get digital version of document (PDF)

Parameters:

1 sessionId	(string)	Session id obtained with doLogin method
2 documentId	(string)	Document id

Return:

1 eDocument	(string)	PDF data of document (base64-encoded)

Error codes:

ERR_WRONG_DOCUMENT_ID	Wrong document id
ERR_NO_EDOCUMENT	Digital version of document is not available

XML example:

```
   <?xml version="1.0" encoding="utf-8" ?>
   <getEDocument>
      <sessionId>2BZY8zSx0DOOUmMOylFKSZjMsgDG564YxTMvju_D</sessionId>
      <documentId>00001277</documentId>
   </getEDocument>

Response:
   <?xml version="1.0" encoding="utf-8" ?>
   <getEDocumentResponse>
      <eDocument>PDF zakodowany base64</eDocument>
   </getEDocumentResponse>
```

doDocumentPrint

Generate PDF printout of document

Parameters:

1 sessionId	(string)	Session id obtained with doLogin method
2 documentId	(string)	Document id

Return:

1 pdfData	(string)	Printout of document

Error codes:

ERR_WRONG_DOCUMENT_ID	Wrong document id
ERR_DOCUMENTPRINT_INACTIVE	Printout is unavailable
ERR_EDOCUMENT_EXISTS	This document exists in electronic form

doSetDeliveryConfirmation

Delivery confirmation for WZ document

Parametry:

1 sessionId	(string)	Session id obtained with doLogin method
2 documentId	(string)	Document id

Wartość zwracana:

1 status	(int)	1 – confirmed documents

Kody błędów:

ERR_WRONG_DOCUMENT_ID	Wrong document id
ERR_DOCUMENT_ALREADY_CONFIRMED	Document already confirmed

ROUTES

getMyRoutes

Get list of users routes

Parameters:

1 sessionId	(string)	Session id obtained with doLogin method

Return:

1 warehouseList	(struct[])	List of routes for available warehouses
1 warehouse	(struct)	List of routes in a single warehouse
1 warehouseId	(string)	Warehouse id
2 routeList	(struct[])	List of routes
1 route	(struct)	Information about route
1 id	(string)	Route id
2 description	(string)	Route description
3 isDefault	(int)	Default route in warehouse (deprecated, always returns: 0)
2 route ...		
3 warehouseId ...		
2 warehouse ...		

Error codes:

none

XML example:

```
<getMyRoutes>
   <sessionId>V5etQUyinDnHosbSzjy7iLdj3Us4GaEr5ILMt9_D</sessionId>
</getMyRoutes>

Response:

<?xml version="1.0" encoding="utf-8" ?>
<getMyRoutesResponse>
   <warehouseList>
      <warehouse>
         <warehouseId>01</warehouseId>
         <routeList>
            <route>
               <id>314A</id>
               <description>trasa pi  ( 12:00 - 13:00 ) Def.</description>
               <isDefault>1</isDefault>
            </route>
            <route>
               <id>111A</id>
               <description>Odbiór osobisty</description>
               <isDefault>0</isDefault>
            </route>
         </routeList>
      </warehouse>
   </warehouseList>
</getMyRoutesResponse>
```

JSON example:

```
{
  "getMyRoutes": { "sessionId": "V5etQUyinDnHosbSzjy7iLdj3Us4GaEr5ILMt9_D" }
}

Response:

{
  "getMyRoutesResponse": {
    "warehouseList": {
      "warehouse": {
        "warehouseId": "01",
        "routeList": {
          "route": [
            {
              "id": "314A",
              "description": "trasa pi ( 12:00 - 13:00 ) Def.",
              "isDefault": "1"
            },
            {
              "id": "111A",
              "description": "Odbiór osobisty",
              "isDefault": "0"
            }
          ]
        }
      }
    }
  }
}
```

LOGISTICS

getMyPackages

Get list of users packages

Parameters:

`1 sessionId`	`(string)`	Session id obtained with doLogin method
`2(?) packageIdList`	`(string[])`	List of packages to return
`   1 id`	`(string)`	Package Id
`   2 id ...`		
`3(?) filterOptions`	`(struct)`	Packages filter
`   1(?) dateFrom`	`(date)`	[format: yyyy-mm-dd] Begin of date range. By default, packages from last 7 days are returned (if parameter `packageIdList` was not given) or from any range (if parameter `packageIdList` was given)
`   2(?) dateTo`	`(date)`	[format: yyyy-mm-dd] End of date range (default:today)
`   3(?) documentIdList`	`(string[])`	Documents Id related with packages
`      1 id`	`(string)`	Document Id
`      2 id ...`		
`   4(?) routeId`	`(string)`	Route Id
`4(?) sortOptions`	`(struct)`	Sorting options
`   1(?) sortType`	`(int)`	Sort type (1 - by date of create (default))
`   2(?) sortOrder`	`(int)`	Sort order (1-ascending, 2-descending (default))
`5(?) pageSize`	`(int)`	Number of packages on page (min: 1, max: 50, default 10)
`6(?) pageNumber`	`(int)`	Page number(default: 1)
`7(?) getItems`	`(int)`	Return information about items (default: 0 = No, 1 = Yes)
`8(?) getTrackingUrl`	`(int)`	Return tracking urls for packages (default: 0 = No, 1 = Yes)

Return:

`1 packageList`	`(struct[])`	List of packages
`   1 package`	`(struct)`	Package information
`      1 id`	`(string)`	Package Id
`      2 date`	`(date)`	[format: yyyy-mm-dd] Data of create
`      3 routeId`	`(string)`	Route id
`      4 srcDocId`	`(string)`	Package's number document. If it's empty, package contain items from a many documents
`      5 weight`	`(int)`	Package weight in kg
`      6 x`	`(int)`	Size 1
`      7 y`	`(int)`	Size 2
`      8 z`	`(int)`	Size 3
`      9 isClosed`	`(int)`	Is package closed
`      10 urlTracking`	`(string)`	URL to the package history in TMS
`      11 items`	`(struct[])`	List of package item (only if `getItems` = 1)
`         1 item`	`(struct)`	Item
`            1 productId`	`(string)`	Product Id
`            2 Ilkod`	`(string)`	Product ILkod
`            3 itemSrcDocId`	`(string)`	Source document Id
`            4 quantity`	`(int)`	Quantity
`      2 item ...`		
`   2 package ...`		

Error codes:

ERR_INVALID_WAREHOUSE	Invalid warehouse
ERR_WAREHOUSE_ACCESS_DENIED	No access to the selected warehouse

doLogmCreatePackage

New fast sales method – create package based on the contents of another package and generate the PDF label

Parameters:

1 sessionId	(string)	Session id obtained with doLogin method
2 packMergeKey	(string)	Key used for merging the packages
3 externalPackageId	(string)	Id of the external (source) package
4 (?) x	(int)	dimension 1
5 (?) y	(int)	dimension 2
6 (?) z	(int)	dimension 3
7 (?) weight	(int)	Package weight in KG
8 packageItemList	(struct[])	Contents of the package
1 packageItem	(struct)	Package item
1 (?) srcDocId	(string)	Id of the source order document
2 ILkod	(string)	Product ILkod
3 quantity	(int)	Product quantity
4 buyPrice	(float)	Product buy price
5 hasDeposit	(int)	Does product have a deposit (1 – yes, 0 – no)
6 (?) depositBuyPrice	(float)	Deposit buy price
7 (?) depositIlkod	(string)	Deposit ILkod
2 packageItem …		
9 collectivePackageInfo	(struct)	Collective Package – information
1 collectivePackageId	(string)	Collective Package ID
2 collectiveProductILkod	(string)	ILkod of product related to Package
3 collectivePackageQuantity	(int)	Used quantity
4 collectivePackagePriceNetto	(float)	Unit price netto

Return:

1 (?) labelPdfData	(string)	System package label in PDF format encoded in BASE64
2 (?) specificationPdfData	(string)	Package specification in PDF format encoded in BASE64
3 (?) courierLabelList	(struct[])	List of courier labels
1 type	(string)	Labels format (pdf/zpl)
2 data	(string)	Courier label encoded in BASE64
4 (?) dropshippingLabelList	(struct[])	List of dropshipping labels
1 type	(string)	Labels format (only pdf)
2 data	(string)	Dropshipping label encoded in BASE64

Error codes:

ERR_METHOD_BLOCKED	Method is blocked
ERR_UNKNOWN_ILKOD	Unknown product ILkod
ERR_INVALID_PACKMERGEKEY	Invalid pack merge key
ERR_INVALID_PRODUCT	Invalid product
ERR_INVALID_QUANTITY	Invalid quantity
ERR_INVALID_BUY_PRICE	Invalid buy price
ERR_INVALID_DEPOSIT_BUY_PRICE	Invalid deposit buy price
ERR_INVALID_EXTERNAL_PACKAGE_ID	Invalid external package id
ERR_EMPTY_PACKAGE	Empty item list

Request example:

```
{
   "doLogmCreatePackage": {
      "sessionId": "QX65v09SbmwT8bGfZYwryEru9tZFxez0019618_D",
      "packMergeKey": "1",
      "externalPackageId": "00092224",
      "packageItemList": {
         "packageItem": [
            {
               "srcDocId": "00001620",
               "ILkod": "111",
               "quantity": 20,
               "buyPrice": 10.01,
               "hasDeposit": 0
            },
            {
               "srcDocId": "00001621",
               "ILkod": "2222",
               "quantity": 5,
               "buyPrice": 100.01,
               "hasDeposit": 0
            }
         ]
      }
   }
}
```

Response example:

```
{
   "doLogmCreatePackageResponse": {
      "labelPdfData": "SWRlYWx5IH (...) uaW1pIGtpZXJvd2FjLg0K"
   }
}

Or

{
"doLogmCreatePackageResponse": {
         "specificationPdfData": "SWRlYWx5IH (...) uaW1pIGtpZXJvd2FjLg0K",
         "courierLabelList": {
         "label" : [
            {
               "type": "pdf",
               "data": "SWRlYWx5IH (...) uaW1pIGtpZXJvd2FjLg0K"
            },
            {
               "type": "zpl",
             "data": "SWRlYWx5IH(...)uaW1pIGtpZXJvd2FjLg0K"
            }
         ]
      }
   }
 }
```

PAYMENTS

getMyPayments

Get list of payments

Parameters:

1 sessionId	(string)	Session id obtained with doLogin method
2 paymentStatus	(int)	Payment status (0-unpaid, 1-paid, default: 0)
3(?) filterOptions	(struct)	Payments filter
1(?) dateFrom	(date)	[format: yyyy-mm-dd] Begin of date range.
2(?) dateTo	(date)	[format: yyyy-mm-dd] End of date range.
4(?) sortOptions	(struct)	Informacje o sposobie sortowania
1(?) sortType	(int)	Sort type (1 - by date of create (default)), 2 – by date oy payment)
2(?) sortOrder	(int)	Sort order (1-ascending, 2-descending (default))
5(?) pageSize	(int)	Number of payment on page (min: 1, max: 50, default 10)
6(?) pageNumber	(int)	Page number (default: 1)

Return:

1 paymentList	(struct[])	List of payments
1 payment	(struct)	Payment information
1 id	(string)	Payment Id
2 createDate	(date)	[format: yyyy-mm-dd] Date of create
3 paymentDate	(date)	[format: yyyy-mm-dd] Date of payment
4 status	(int)	Payment status (0-unpaid, 1-paid)
5 documentNumber	(string)	Document Id
6 daysLeft	(int)	Number of days to payment's date (negative means number days after to payment's date)
7 warehouseId	(string)	Warehouse Id
8 valueTotal	(float)	Total value
9 valueToPay	(float)	Unpaid value
10 currency	(string)	Currency
11 isRelDocument	(string)	Payment is related with document (0 = No,1 = Yes)
12 relDocumentId	(string)	Related document id
13 relDocumentType	(string)	Related document type
14 payItemList	(struct[])	Payment's item
1 payItem	(struct[])	Payment's item information
1 amount	(int)	Amount
2 date	(date)	Date
3 description	(string)	Description
2 item ...		
2 payment ...		
2 paymentsCount	(int)	Number of payments that meets the criteria value can be approximated

Error code

none

Example XML:

```
Request:
   <?xml version="1.0" encoding="utf-8" ?>
      <getMyPayments>
         <sessionId>2BZY8zSx0D0OUmM0ylFKSZjMsgDG564YxTMvju_D</sessionId>
         <paymentStatus>0</paymentStatus>
         <filterOptions>
            <dateFrom>2013-03-01</dateFrom>
            <dateTo>2013-03-07</dateTo>
         </filterOptions>
         <sortOptions>
            <sortType>1</sortType>
            <sortOrder>1</sortOrder>
         </sortOptions>
         <pageSize>50</pageSize>
         <pageNumber>1</pageNumber>
      </getMyPayments>

Response:
   <?xml version="1.0" encoding="utf-8" ?>
      <getMyPaymentsResponse>
         <paymentList>
            <payment>
               <id>00001277</id>
               <createDate>2014-08-04</createDate>
               <paymentDate>2014-08-11</paymentDate>
               <status>0</status>
               <documentNumber>F/1024/CE/2014</documentNumber>
               <daysLeft>-3</daysLeft>
               <warehouseId>CE</warehouseId>
               <valueTotal>123.45</valueTotal>
               <valueToPay>123.45</valueToPay>
               <currency>PLN</currency>
               <isRelDocument>1</isRelDocument>
               <relDocumentId>00001234</relDocumentId>
               <relDocumentType>WZ</relDocumentType>
            </payment>
            <payment>
               ...
            </payment>
         </paymentList>
      </getMyPaymentsResponse>
```

RETURNS

getMyReturns

Get list of users returns

Parameters:

1 sessionId	(string)	Session id obtained with doLogin method
2(?) returnIdList	(string[])	List of returns
1 id	(string)	Return id
2 id ...		
3(?) filterOptions	(struct)	Returns filter
1(?) state	(int)	State return: 0-waiting, 1-completed, (default: 0)
2(?) packageNumber	(int)	Package number

Return:

1 returnsCount	(int)	Count of returns. This value can be approximate.
2 returnList	(struct[])	List of returns
1 return	(struct)	Information about return
1 id	(string)	Return id
2 documentNumber	(string)	Number of the linked sales document
3 dateCreate	(date)	Date of create return
4 state	(int)	Return state: 0-waiting, 1-completed
5 warehouseId	(string)	Warehouse id
6 productId	(string)	Product id
7 productCode	(string)	Product reference
8 productName	(string)	Product name in user language
9 quantity	(float)	Quantity
10 ildec	(int)	Decimal places
11 status	(int)	Return status: 0-added, 1-sent, 2-package received, 3-accepted, 4-rejected
12 packageNumber	(int)	Package number
13 (?) packageBarcode	(string)	EAN-13 Barcode
14 dateReceived	(date)	Date of received package
15 info	(text)	Information about return
16 canEdit	(int)	Is edition available
2 return ...		

Error codes:

ERR_RETURNS_INACTIVE	Returns are not active

doReturnCreatePackage

Create number package for list of returns.

Parameters:

1 sessionId	(string)	Session id obtained with doLogin method
2 returnIdList	(string[])	List of returns
1 id	(string)	Return id
2 id ...		
3 overwrite	(int)	Generate package number for returns again. (1-yes, 0-no), (default:0)

Return:

1 packageNumber	(int)	Package number

Error codes:

ERR_RETURNS_INACTIVE	Returns are not active
ERR_MULTI_WAREHOUS	All returns must come from one warehouse
ERR_PACKAGE_ALREADY_EXIST	Package has already generated
ERR_EMPTY_RETURN_ID_LIST	Empty list of returns id
ERR_INVALID_RETURN_ID	Invalid return id
ERR_INVALID_RETURN_STATE	Invalid return state
ERR_INVALID_RETURN_STATUS	Invalid return status
ERR_INTERNAL	Generating labels

doDeleteReturn

Delete registered return

Parameters:

1 sessionId	(string)	Session id obtained with doLogin method
2 returnId	(string)	Return id
3 deleteIfSent	(int)	Delete return, which was sent (1-yes, 0-no), (default:0)

Return:

1 status	(int)	Status: 1-succeed

Error codes:

ERR_RETURNS_INACTIVE	Returns are not active
ERR_INVALID_RETURN_ID	Invalid return id
ERR_CANNOT_DELETE	Can’t delete return (contact administrator)
ERR_WAREHOUSE_ACCESS_DENIED	No access to chosen warehouse
ERR_RETURN_SENT	The return was already sent

doEditReturn

Edit Return

Parameters:

`1 sessionId`	`(string)`	Session id obtained with doLogin method
`2 returnId`	`(string)`	Return id
`3 returnInfo`	`(struct)`	Information about return
`1(?) info`	`(text)`	Information about return

Return:

`1 status`	`(int)`	Status: 1-succeed

Error codes:

`ERR_RETURNS_INACTIVE`	Returns are not active
`ERR_INVALID_RETURN_ID`	Invalid return id
`ERR_RETURN_STATUS`	Can’t edit return (status)
`ERR_INVALID_RETURN_INFO`	Invalid information about return

`doCreateReturn`

Create new return

Parameters:

`1 sessionId`	`(string)`	Session id obtained with doLogin method
`2 productId`	`(string)`	Product id
`3 warehouseId`	`(string)`	Warehouse id
`4 quantity`	`(float)`	Quantity to return
`5 (?) srcDocumentId`	`(string)`	Source document id

Return:

`1 returnId`	`(string)`	Return id

Error codes:

`ERR_RETURNS_INACTIVE`	Returns are not active
`ERR_INVALID_WAREHOUSE`	Invalid warehouse id
`ERR_WAREHOUSE_ACCESS_DENIED`	No access to chosen warehouse
`ERR_WRONG_PRODUCT_ID`	Wrong product id
`ERR_CANNOT_RETURN`	Can’t return product <reason>
`ERR_QUANTITY_TOO_MUCH`	Too much quantity (max: %%)
`ERR_QUANTITY`	Incorrect quantity (none or <= 0)
`ERR_QUANTITY_FORMAT`	Invalid format quantity (number of decimal places)
`ERR_WRONG_SOURCE_DOCUMENT`	Incorrect source document
`ERR_WRONG_SOURCE_DOCUMENT_WAREHOUSE`	Incorrect warehouse of the source document

getReturnQuantityAvailable

Get available quantity product possible to return

Parameters:

1 sessionId	(string)	Session id obtained with doLogin method
2 productId	(string)	Product id
3(?) warehouseId	(string)	Warehouse id
4(?) detailedQuantity	(int)	Detailed quantities grouped by document (1-yes, 0-no). Default: 0

Return:

1 quantity	(float)	Quantity possible to return
2 limitUsed	(float)	Used percent for limit granted returns
3 returnInfo	(string)	Additional information about return
4 (?) documentList	(struct[])	Detailed quantities grouped by document
1 document	(struct)	
1 id	(string)	Document id
2 numer	(string)	Document number
3 quantity	(float)	Quantity on the document which is possible to be returned
4 quantityAvailable	(float)	Quantity on the document which is available to be returned

Error codes:

ERR_RETURNS_INACTIVE	Returns are not active
ERR_INVALID_WAREHOUSE	Invalid warehouse id
ERR_WAREHOUSE_ACCESS_DENIED	No access to chosen warehouse
ERR_WRONG_PRODUCT_ID	Wrong product id
ERR_CANNOT_RETURN	Can’t return product <reason>

getReturnNowProductInfo

Additional information related to product quick return

Parameters:

1 sessionId	(string)	Session id obtained with doLogin method
2 (?) warehouseId	(string)	Warehouse id (by default it is the customer's default warehouse)
3 product	(struct)	Product information
1 (?)* id	(string)	Product id
2 (?)* reference	(string)	Product symbol
3 (?)* tecidd	(string)	TecDoc supplier id
4 (?)* tecnum	(string)	TecDoc product id
5 (?)* ILkod	(string)	Ilkod product id
6 (?)* motonet	(string)	Motonet number

Return:

1 isReturnBlocked	(int)	Is product return blocked (1 – yes, 0 – no)
2 isMainWarehouseAcceptBlocked	(int)	Is product accept blocked by the main warehouse (1 – yes, 0 – no)
3 returnSectorInfo	(struct)	Sector info
1 id	(string)	sector id
2 name	(string)	sector name

Error codes:

ERR_NO_WAREHOUSE	No available warehouses to order
ERR_WAREHOUSE_UNAVAILABLE	In warehouse %% user can't make orders
ERR_METHOD_BLOCKED	Method is blocked
ERR_WRONG_PRODUCT_ID	Wrong product id

Request example:

```
{
   "getReturnNowProductInfo": {
      "sessionId": "elcD41ENipXxmn19N1uBasurWOb6FY20010825_D",
      "warehouseId": "01",
      "product": {
         "id": "0003AZ"
      }
   }
}
```

Response example:

```
{
   "getReturnNowProductInfoResponse": {
      "isMainWarehouseAcceptBlocked": "0",
      "isReturnBlocked": "0",
      "returnSectorInfo": {
         "id": "0001",
         "name": "sector 1"
      }
   }
}
```

doReturnClearPackage

Delete information about package from list of returns

Parameters:

1 sessionId	(string)	Session id obtained with doLogin method
2 returnIdList	(string[])	List of returns
1 id	(string)	Return id
2 id ...		

Return:

1 status	(int)	Status: 1-succeed

Error codes:

ERR_RETURNS_INACTIVE	Returns are not active
ERR_INVALID_RETURN_ID	Invalid return id
ERR_RETURNS_CLOSED	The returns was already closed
ERR_RETURNS_SENT	The returns was already sent

`doReturnGetPackageLabel`

Get package label for return

Parameters:

`1 sessionId`	`(string)`	Session id obtained with doLogin method
`2 packageNumber`	`(int)`	Package number

Return:

`1 labelPDFData`	`(string)`	PDF print label (base64-encoded)

Error codes:

`ERR_RETURNS_INACTIVE`	Returns are not active
`ERR_WRONG_PARAM`	Missing or incorrect parameter (parameter)
`ERR_PACKAGE_NOT_FOUND`	Package number was not found
`ERR_INVALID_RETURN_STATE`	Invalid return state
`ERR_INVALID_RETURN_STATUS`	Invalid return status
`ERR_INTERNAL`	Generating labels

`doReturnSendPackage`

Mark return from package as sent

Parameters:

`1 sessionId`	`(string)`	Session id obtained with doLogin method
`2 packageNumber`	`(int)`	Package Number

Return:

`1 status`	`(int)`	Status: 1-succeed

Error codes:

`ERR_RETURNS_INACTIVE`	Returns are not active
`ERR_PACKAGE_NOT_FOUND`	Package number was not found
`ERR_INVALID_RETURN_STATE`	Invalid return state
`ERR_INVALID_RETURN_STATUS`	Invalid return status
`ERR_INTERNAL`	Generating labels

doReturnNow

Quick return

Parameters:

1 sessionId	(string)	Session id obtained with doLogin method
2 (?)warehouseId	(string)	Warehouse id (def.: default client warehouse)
3 product	(struct)	Product information
1 (?)*id	(string)	Product id
2 (?)*reference	(string)	Product reference
3 (?)*tecidd	(string)	TecDoc supplier id
4 (?)*tecnum	(string)	TecDoc product id
5 (?)*Ilkod	(string)	ILKod product id
6 (?)*motonet	(string)	Motonet number
4 quantity	(float)	Return quantity
5 (?)*boxId	(string)	Box id
6 (?)*returnAsManyAsPossible	(int)	Return as many as possible (1-yes, 0-no, def. 0)

Return:

1 quantityReturned	(float)	Quantity returned
2 documentInfo	(struct)	Document info
1 id	(string)	Document id
2 dateCreate	(date)	Create date
3 number	(string)	Document number
4 isSplitPayment	(int)	Does document have the split payment marker (1-yes, 0-no)
3 correctedDocumentInfo	(struct)	Corrected document info
1 id	(string)	Corrected document id
2 dateCreate	(date)	Create date
3 number	(string)	Corrected document number
4 itemInfo	(struct)	Item info
1 id	(string)	Item id
2 productId	(string)	Product id
3 productReference	(string)	Product reference
4 Ilkod	(string)	ILKod product id
5 productName	(string)	Product name
6 decimals	(int)	Decimal places(0-3)
7 tax	(float)	Tax rate
8 quantity	(float)	Quantity
9 discountPercent	(float)	Discount percent
10 startingPriceExclTax	(float)	Price without tax in document currency
11 startingPriceInclTax	(float)	Price with tax in document currency
12 correctedItem	(struct)	Corrected item info
1 quantity	(float)	Quantity
2 discountPercent	(float)	Discount percent
3 startingPriceExclTax	(float)	Price without tax in document currency
4 startingPriceInclTax	(float)	Price with tax in document currency

Error codes:

ERR_NO_WAREHOUSE	No available warehouses to order
ERR_WAREHOUSE_UNAVAILABLE	In warehouse %% user can't make orders
ERR_METHOD_BLOCKED	Method blocked
ERR_WRONG_PRODUCT_ID	Invalid product id
ERR_CANNOT_RETURN	Can’t return product <reason>
ERR_QUANTITY_TOO_MUCH	Too much quantity (max: %%)
ERR_QUANTITY	Incorrect quantity (none or <= 0)
ERR_QUANTITY_FORMAT	Invalid format quantity (number of decimal places)
ERR_INVALID_BOX_ID	Invalid box id

`ERR_BOX_STATUS`	Invalid box status
`ERR_WRONG_RETURN_SECTOR`	Invalid sector

Comments:

* One of these is required: "id" or "reference" or both "tecidd" and "tecnum" or “motonet”.
* In case of sending TecDoc or Motonet data - information about first fitting product is returned.