Data import in Aplikace Zásilky

WE|DO by allegro

Data import
in Aplikace Zásilky

There are implemented data import for selected entities in Aplikace Zásilky (called AZ).
Logged user has access to this tools depending on assigned rights.

Article receivers import

Article receivers import should be used by any user in AZ,
who has right **Vytvářet příjemce zásilek**. Import accepts CSV file with fixed structure
and there is no variant. Sample CSV file and description should be downloaded
in AZ section **Ke stažení**:

http://zasilky.intime.cz/download/prijemci.csv

http://zasilky.intime.cz/download/prijemci.txt

Popis CVS souboru pro import příjemců zásilek:

- **columns separated by semicolon**
- **text encoding windows-1250**
- **value containing semicolon have to be encapsulated into "**
- **description of columns**

1. column	- mandatory field - receiver name - varchar(100)
2. column	- mandatory field - receiver address state - char(3) - accepted values: CZ, SK, UK, PL, DE, AU
3. column	- mandatory field - receiver address street - varchar(100)
4. column	- mandatory field - receiver address town - varchar(50)

5. column	- mandatory field - receiver address postal code - char(5)
6. column	- receiver first name - varchar(30)
7. column	- receiver surname - varchar(30)
8. column	- receiver email - varchar(50)
9. column	- receiver phone - char(15)
10. column	- receiver mobile - char(15)
11. column	- receiver fax - char(15)
12. column	- receiver external ID - varchar(255) - for pairing data during import to existing record in AZ
13. column	- receiver note - varchar(75)

Articles import

Articles import should be used by any AZ user, who has right **Upravovat informace o zásilkách**.
Articles import should be realised by these ways:

- CSV file with fixed structure, this structure is serious then
- CSV file and corresponding CVS converter in AZ
- XML request sent to AZ XML server

Articles imports from CSV file with fixed structure

Description of CVS file:

- **columns separated by semicolon**
- **text encoding windows-1250**
- **value containing semicolon have to be encapsulated into "**
- **description of columns**

1. column	- receiver external ID - varchar(255) - for pairing data during import to existing record in AZ
2. column	- mandatory field - receiver name - varchar(100)
3. column	- mandatory field - receiver address state - char(3) - accepted values: CZ, SK, UK, PL, DE, AU
4. column	- mandatory field - receiver address street - varchar(100)
5. column	- mandatory field - receiver address town - varchar(50)
6. column	- mandatory field - receiver address postal code - char(5)
7. column	- receiver first name - varchar(30)
8. column	- receiver surname - varchar(30)
9. column	- receiver email - varchar(50)
10. column	- receiver phone - char(15)
11. column	- receiver mobile - char(15)
12. column	- receiver fax - char(15)
13. column	- receiver note - varchar(75)
14. column	- article reference number - varchar(50)
15. column	- mandatory field - number of packages - integer
16. column	- mandatory field - total weight - float (two decimal places separated by point mark)
17. column	- total volumetric weight - float (two decimal places separated by point mark)
18. column	- mandatory field - total value of article - float (two decimal places separated by point mark)

19. column	- cash on delivery value - float (two decimal places separated by point mark) - if empty no cash on delivery
20. column	- insurance request - enum(Y,N)
21. column	- document back request - enum(Y,N)
22. column	- note for document back request - varchar(255)
23. column	- phone notification request - enum(Y,N)
24. column	- phone number for phone notification - char(15) - in national format (for example 00420123456789)
25. column	- note for phone notification - varchar(255)
26. column	- SMS notification request - enum(Y,N)
27. column	- phone number for SMS notification - char(15) - in national format (for example 00420123456789)
28. column	- carry request - enum(Y,N)
29. column	- loss request - enum(Y,N)
30. column	- pay by receiver request - enum(Y,N)
31. column	- article note - varchar(255)
32. column	- list of package codes separated by comma - varchar(255)
33. column	- article second reference number - varchar(50)
34. column	- authentication request - enum(Y,N)
35. column	- delivery type - mandatory field - delivery type short name (list of acceptable values should be retrieved from WE|DO sales department) - actual acceptable delivery type values HD, VM, BOX, AD - varchar(20)
36. column	- reverse order request - enum(Y,N)
37. column	- direct order request - enum(Y,N)
38. column	- applicable only in case of direct order - direct order sender external ID - varchar(255) - for pairing data during import to existing record in AZ

39. column	- applicable only in case of direct order - mandatory field - sender name - varchar(100)
40. column	- applicable only in case of direct order - mandatory - sender address street - varchar(100)
41. column	- applicable only in case of direct order - mandatory field - sender address town - varchar(50)
42. column	- applicable only in case of direct order - mandatory field - sender address postal code - char(5)
43. column	- applicable only in case of direct order - mandatory field - sender address state - char(3) - acceptable values: CZ, SK, AU, HU
44. column	- applicable only in case of direct order - sender first name - varchar(30)
45. column	- applicable only in case of direct order - sender surname - varchar(30)
46. column	- applicable only in case of direct order - sender email - varchar(50)
47. column	- applicable only in case of direct order - sender phone - char(15)
48. column	- applicable only in case of direct order - sender mobile - char(15)
49. column	- applicable only in case of direct order - sender fax - char(15)
50. column	- applicable only in case of direct order - sender note - varchar(75)
51. column	- authentication request note - varchar(255)
52. column	- email notification request - enum(Y,N)
53. column	- email address for email notification - varchar(50)

54. column	- exchange request - enum(Y,N)
55. column	- exchange request note - varchar(255)
56. column	- WE|DO product request, to be used for delivering - product short name (list of acceptable values should be retrieved from WE|DO sales department)
57. column	- not used field - have to be empty
58. column	- applicable only in case of direct order - sender phone number for phone notification - char(15) - in national format (for example 00420123456789)
59. column	- applicable only in case of direct order - sender phone number for SMS notification - char(15) - in national format (for example 00420123456789)
60. column	- applicable only in case of direct order - sender email address for email notification - varchar(50)
61. column	- not used field - have to be empty
62. column	- COFIDIS contract request - enum(Y,N)
63. column	- COFIDIS contract number - varchar(20)
64. column	- Poštomat box identifier - mandatory field if column 35 (delivery type)=BOX - char(9)
65. column	- phone contact for Poštomat box - mandatory field if column 35 (delivery type)=BOX - char(15)
66. column	- personal purchase request - enum(Y,N)
67. column	- personal purchase place indentifier - integer
68. column	- requested delivery date of article - format YYYY-MM-DD
69. column	- SCAN document back request - enum(Y,N)
70. column	- pup branch identifier - mandatory field if column 35 (delivery type)=VM - char(9)
71. column	- phone contact for pup branch - mandatory field if column 35 (delivery type)=VM - char(15)

Articles import from CSV file and corresponding CVS converter in AZ

This way is useful for customer with systems, that have no possibility to change export data structure.

Creating of CSV file converter

First step is to set-up columns of customer CSV file where matching data for AZ are stored. These settings should be saved by user with rights **Administrátor zákazníka** in section **Moje údaje**.

At bottom of the screen is located section called **Převod CSV pro zásilky**. After clicking button **Přidat** it's possible to create new CSV converter. The list of existing converters is also displayed here. Existing converter should edited by clicking button **Detail** or deleted by clicking button **Smazat**.

Editing of converter

Each converter has name (item **Označení nastavení převodu CSV**). Name of converter is useful to choose accurate, because of choosing converter later for import of articles.

It's needed to specify where matching data for each information AZ require are located in CSV file. Bold fields are mandatory and they are required during imports of articles.

Value of each field should be one these possibilities:

- **static value** – in case you want article to have predefined value, type this value into field (for example to import all article with weight 1kg, type value „1" to field **Hmotnost celé zásilky)**
- **value of CSV file column** – in case you want article to have value from CSV file column, type value „{sloupec<number of column>}" into field (for example if weight of article is in first column of CSV file type value „{sloupec1}" to field **Hmotnost celé zásilky)**
- **value of CSV file columns** – in case you want article to have value of more than one column from CSV file, type value meaning each column in order you want to compose final value (for example if street name is in first column and house number is in second column and you want to concat these values, type value „{sloupec1} {sloupec2}"

- **combination of static value and value of CSV file column** – in case you want to have value as combination of static value and value of CSV file column, type value representing static value a reference to CSV column same way as in previous cases (for example if you want to add character „A“ to first column of CSV file type „A{sloupec1}“)
- **formula** – in some cases it’s not possible to determine value easily, that’s why there is formula system, which should be used; this system is explained later

When all mandatory fields of CSV converter are filled in, converter should be save by clicking button **Uložit nastavení převodu CSV pro zásilky**.

Using of CSV converter

Saved CSV converter should be used by any user of customer, who has rights **Upravovat informace o zásilkách**.

List of available CSV converters is displayed during articles import (in section **Zásilky** after clicking button **Importovat zásilky**) in field **Použít převod CSV**.

By selecting CSV converter user confirms, that CSV file format is relevant to CSV converter settings.

After clicking button **Importovat** CSV file is loaded, automatically converted to format AZ expects and articles imported to AZ. If error occurs during import, relevant error message is displayed and no article is imported.

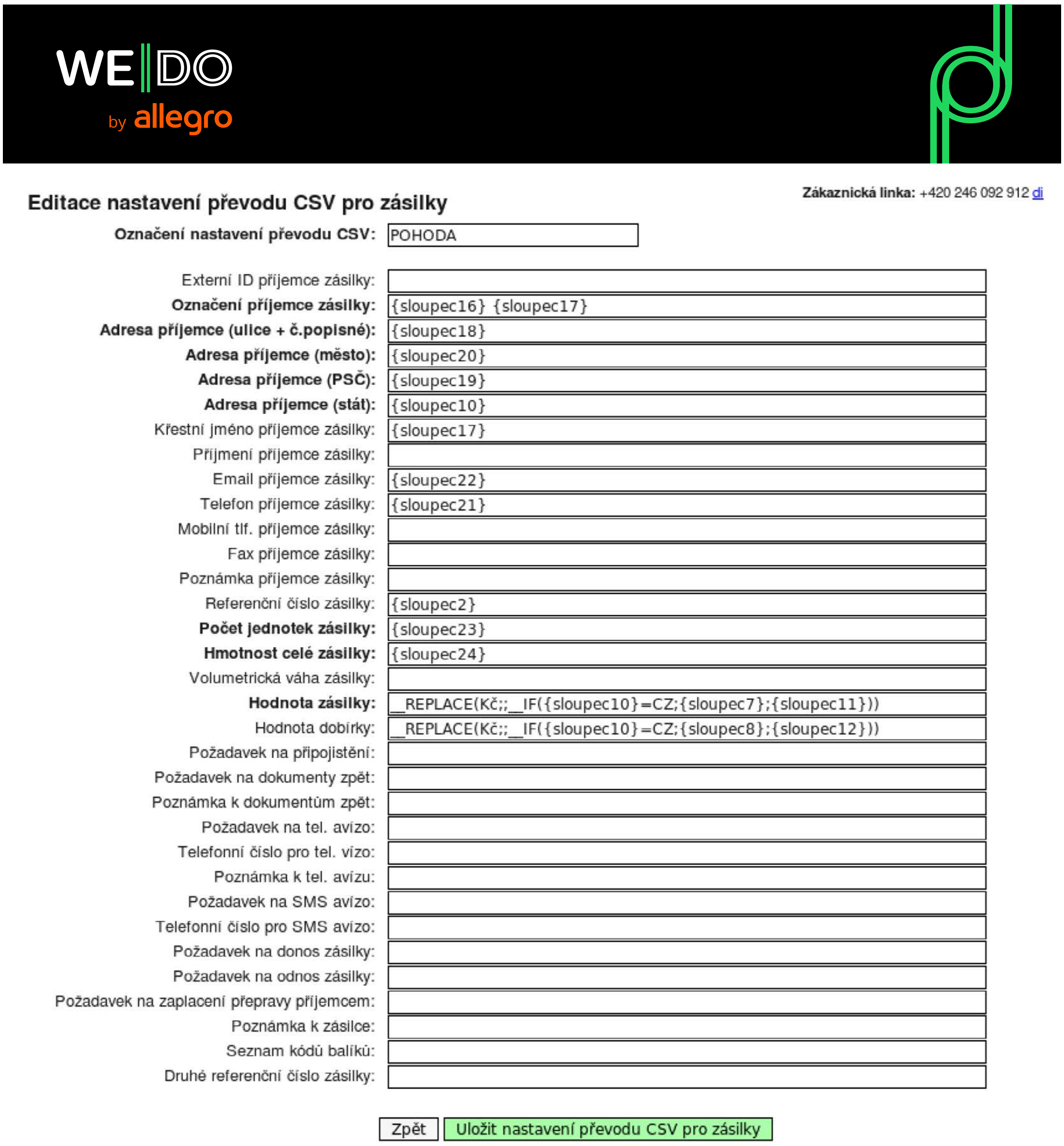
Formulas

In some cases it’s not possible to determine value easily, that’s why there is formula system. AZ should recognize these functions, which should be used to determine value of field:

- **IF** – __IF(<condition>;<true value>;<false value>}
    - by this function it’s possible evaluate some condition and determine different value if condition is fulfill or not fulfill
    - it’s possible to type „{sloupec1}=CZ“, {sloupec2}>2, {sloupec3}<0 into first **parameter** condition
    - second and third parameters should be filled as simple values described earlier
    - **example:** __IF({sloupec1}=CZ;{sloupec2};{sloupec3})
        - in this example the result is value of second column of CSV file in case first column of CSV file is equal to ‘CZ’, otherwise the result is value of third column of CSV file

- **REPLACE** - __REPLACE(<what to replace>;<replacement>;<string>)
    - by this function it's possible to replace portion of string with another string
    - **example:** __REPLACE(Kč;;{sloupec1})
        - in this example the result is value of first column of CSV file but if first column of CSV file was containing string 'Kč', this part was erased

Example of whole CSV converter settings



Editace nastavení převodu CSV pro zásilky

Zákaznická linka: +420 246 092 912 di

Označení nastavení převodu CSV:	POHODA
Externí ID příjemce zásilky:	
Označení příjemce zásilky:	{sloupec16} {sloupec17}
Adresa příjemce (ulice + č.popisné):	{sloupec18}
Adresa příjemce (město):	{sloupec20}
Adresa příjemce (PSČ):	{sloupec19}
Adresa příjemce (stát):	{sloupec10}
Křestní jméno příjemce zásilky:	{sloupec17}
Příjmení příjemce zásilky:	
Email příjemce zásilky:	{sloupec22}
Telefon příjemce zásilky:	{sloupec21}
Mobilní tlf. příjemce zásilky:	
Fax příjemce zásilky:	
Poznámka příjemce zásilky:	
Referenční číslo zásilky:	{sloupec2}
Počet jednotek zásilky:	{sloupec23}
Hmotnost celé zásilky:	{sloupec24}
Volumetrická váha zásilky:	
Hodnota zásilky:	__REPLACE(Kč;;__IF({sloupec10}=CZ;{sloupec7};{sloupec11}))
Hodnota dobírky:	__REPLACE(Kč;;__IF({sloupec10}=CZ;{sloupec8};{sloupec12}))
Požadavek na připojistění:	
Požadavek na dokumenty zpět:	
Poznámka k dokumentům zpět:	
Požadavek na tel. avizo:	
Telefonní číslo pro tel. vízo:	
Poznámka k tel. avízu:	
Požadavek na SMS avizo:	
Telefonní číslo pro SMS avizo:	
Požadavek na donos zásilky:	
Požadavek na odnos zásilky:	
Požadavek na zaplacení přepravy příjemcem:	
Poznámka k zásilce:	
Seznam kódů balíků:	
Druhé referenční číslo zásilky:	

This CSV converter settings defines fields as:

- **Označení příjemce zásilky** = value of 16. column connected with delimiter space with value of 17. column
- **Adresa příjemce (ulice + č.popisné)** = value of 18. column
- **Adresa příjemce (město)** = value of 20. column
- **Adresa příjemce (PSČ)** = value of 19. column
- **Adresa příjemce (stát)** = value of 10. column
- **Křestní jméno příjemce** = value of 17. column
- **Email příjemce** = value of 22. column
- **Telefon příjemce** = value of 21. column
- **Referenční číslo zásilky** = value of 2. column
- **Počet jednotek zásilky** = value of 23. column
- **Hmotnost celé zásilky** = value of 24. column
- **Hodnota zásilky** = in case that value of 10. column is equal to CZ, the result value is value of 7. column without substring Kč, otherwise the result value is value of 11. column without substring Kč
- **Hodnota dobírky** = in case that value of 10. column is equal to CZ, the result value is value of 8. column without substring Kč, otherwise the result value is value of 12. column without substring Kč

icles import based on XML request sent to AZ XML server

This way is useful to customers with robust systems, which allow to define dynamic online exports. Benefit of this way is online communication without any delay.

XML server for Application Zásilky

This module of the Application Zásilky (AZ below) was written on the basis of request for online interconnection from information systems of WE|DO CZ s.r.o. Customer. Therefore this application server, which can communicate with AZ core trough XML protocol, was designed.

Testing version of the application server is running at this URL:

http://zasilky.intime.cz/test/xml_server_v2.php

Production version of the application server is running at:

http://zasilky.intime.cz/xml_server_v2.php

Server is accepting request methods GET/POST /RAW.
Request method GET is excepting XML string in parametr **xml**.

Request is not converted on server-side by default. In case of urlencoded request it's needed to include HTTP header in request which will identify content as urlencoded string (for example Content-type: application/x-www-form-urlencoded).

Each request contains an user identification (username and password to login to AZ) and information of demanded operation with all necessary data. The application server always processes a given request and trough **XML** it sends the result to the client. In the **STATUS** section, each answer from **XML** server contains a state information – whether the request was accepted successfully (**field code = 0**) or any error occurred (**field code > 1**). If there were any errors during the processing, the error description is stored in the **MESSAGE** field.

Query types processed by the XML server:

1. Articles import to AZ
2. Change attributes of existing article in AZ
3. Add package to existing article in AZ
4. Remove package from existing article in AZ
5. Get delivering status of existing article in AZ
6. Delete existing article in AZ
7. Reserve of article collection
8. Cancel reservation of article collection
9. Send articles in AZ to WE|DO
10. Get list of Poštomat boxes
11. Get list of WE|DO products
12. Get list of places for personal purchase
13. Get list of pup branches

Store module

14. Receipt import to AZ – Store module
15. Get status of existing receipt – Store module
16. Delete existing receipt – Store module
17. Reservation import to AZ (personal purchase)– Store module
18. Reservation import to AZ (WE|DO expedition) – Store module
19. Get status of existing reservation – Store module
20. Delete of existing reservation – Store module
21. Transfer between stores import – Store module

1.

Articles import to AZ

User can import any number of articles to AZ.

element *request*

- **name** attribute must be set to import_article

User can decide whether each article should be imported separately or import must be run as a whole (in case that any article can't be imported, no article will be imported).

element *option*

- *transaction* **attribute**
    - **yes** – import is processed only if it successes for all articles
    - **no** – import is run for each single article

User can decide whether imported articles should be automatically sent for processing from AZ to the WE|DO CZ s.r.o. central system or they should be just uploaded to AZ for further data modification.

- *auto_complete* **attribute**
    - **yes** – imported articles will be sent to WE|DO automatically
    - **no** – imported articles will be just uploaded to AZ

If authuser is allowed to import articles for more customers, he has to specify the customer of the current import.

- *customer* **attribute**
    - **value** – customer ID; the list of authorized customer IDs can be got from technical support of the WE|DO

If authuser is allowed to import articles for more departments, he has to specify the department of the current import

- *department* **attribute**
    - **value** – department ID; the list of authorized departments can be got from technical support of the WE|DO

If authuser wants ZPL codes for imported packages to print on special barcode pointer, he has to use option **zpl_code** .

- *zpl_code* **attribute**
    - **yes** – XML answer will include element **zpl** for each package containing ZPL code for printing on special barcode printer
    - **no** – XML answer will not include element **zpl**

Now follows the list of imported articles:

article **element**

- for each imported article there is one article element
- each **article** element can contains this tags:
    - **receiver** – **mandatory element**, defining the article recipient
        - **external_id** – varchar(255); AZ automatically tries to find corresponding article recipient according to this ID; if found, it is updated with data listed in the article; otherwise the new recipient is created
        - **name** – varchar(100); **mandatory field**; recipient name
        - **street** – varchar(100); **mandatory field**; street + land registry number/house number
        - **city** - varchar(50); **mandatory field**; city
        - **postal_code** - varchar(5); **mandatory field**; postal code
        - **state** – enum(CZ, SK); **mandatory field**; country
        - **firstname** - varchar(30); first name
        - **surname** - varchar(30); surname
        - **email** – varchar(50); email address
        - **phone** – char(15); phone
        - **mobile** – char(15); mobile phone
        - **fax** – char(15); fax
        - **note** – varchar(75); note
    - **sender** – mandatory element in case of use direct order (**direct_order**), defining the article tender; there is only one address list defined in AZ, this list can be used for both elements (sender and receiver)
        - **external_id** – varchar(255); AZ automatically tries to find corresponding article sender according to this ID; if found, it is updated with data listed in the article; otherwise the new sender is created
        - **name** – varchar(100); **mandatory field**; sender name
        - **street** – varchar(100); **mandatory field**; street + land registry number/house number
        - **city** - varchar(50); **mandatory field**; city
        - **postal_code** - varchar(5); **mandatory field**; postal code
        - **state** – enum(CZ, SK); **mandatory field**; country
        - **firstname** - varchar(30); first name
        - **surname** - varchar(30); surname
        - **email** – varchar(50); email address
        - **phone** – char(15); phone
        - **mobile** – char(15); mobile phone
        - **fax** – char(15); fax
        - **note** – varchar(75); note
    - **reference_number** – varchar(20); reference number / article ID
    - **reference_number2** – varchar(50); second reference number / article ID
    - **package_count** – integer; **mandatory field**; number of parcels in the article
    - **package_number** – varchar(50); optional customer package number; if used number of this elements have to correspond to package count of article in element package_count
    - **weight** – float (‘,’ as delimiter, two decimal places max.); **mandatory field**; total weight of the article in kg
    - **volumetric_weight** – float (‘,’ as delimiter, two decimal places max.); total volumetric weight of article in kg
    - **value** – float (‘,’ as delimiter, two decimal places max.); **mandatory field**; total value of the article

- **comment** – varchar(255); note
- **additive** – enum(‘Y’,’N’); request for enabling adding packages when article is created; if used with ‘Y’ value, it will be possible to add packages to article in future
- **product** – varchar(255); WE|DO product to be used with article delivery; if not used, product will be determined automatically by XML server; list of possible values can be obtained from sales department WE|DO
- **packmachine** – char(9); identifier of Poštomat box; **is acceptable only in case of choosing WE|DO product Poštomat**; then this attribute is mandatory; identification of product Poštomat should be obtained from WE|DO sales department; list of acceptable Poštomat boxes should be retreived by request **Get list of acceptable Poštomat boxes**
- **packmachine_contact** – char(15); receiver phone contact in case of choosing WE|DO product Poštomat; then this attribute is mandatory; have to be in national format (for example 00420123456789)
- **additional_service** – additional service for the article transportation; **name** attribute determines the service type, **value** contains its attributes
    - **cash_on_delivery** -float (‘,’ as delimiter, two decimal places max.); cash on delivery value to get payed from article recipient
    - **reverse_order** – enum(‘Y’,’N’); request for reverse article service
    - **direct_order** – enum(‘Y’,’N’); request for direct article service
    - **exchange_order** - enum(‘Y’,’N’); request for exchange article service
    - **exchange_order_note** – varchar(255); optional not for exchange article service
    - **document_back** – enum(‘Y’,’N’); request for the document back service
    - **document_back_note** – varchar(255); optional note for document back service
    - **document_back_scan** – enum(‘Y’,’N’); request for SCAN document back service
    - **insurance** – enum(‘Y’,’N’); request for the additional insurance service
    - **phone_notification** – enum(‘Y’,’N’); request for the phone confirmation of delivery
    - **phone_notification_number_direct** – char(15); phone number of sender for phone confirmation request; can be filled only in case of use direct order (**direct_order**)
    - **phone_notification_number** – char(15); phone number of receiver for phone confirmation request
    - **phone_notification_note** – varchar(255); optional note for phone confirmation request
    - **sms_notification** – enum(‘Y’,’N’); request for the sms confirmation of delivery
    - **sms_notification_number_direct** – char(15); phone number of sender for sms confirmation request; can be filled only in case of use direct order (**direct_order**)
    - **sms_notification_number** – char(15); phone number of receiver for sms confirmation request
    - **email_notification** – enum(‘Y’,’N’); request for the email confirmation of delivery
    - **email_notification_address_direct** – varchar(50); email address of sender for email confirmation request; can be filled only in case of use direct order (**direct_order**)
    - **email_notification_address** – varchar(50); email address of receiver for email confirmation request
    - **carry** – enum(‘Y’,’N’); request for the carry service
    - **loss** – enum(‘Y’,’N’); request for the loss service
    - **pay_by_receiver** – enum(‘Y’,’N’); request for transportation payed by recipient service
    - **authentication** – enum(‘Y’,’N’); request for receiver authentication

- **authentication_note** – varchar(255); optional note for receiver authentication request
- **cofidis** – enum('Y','N'); request for COFIDIS contract
- **contract_number** – varchar(20); contract number
- **takeover** – enum('Y','N'); request for personal purchase
- **takeover_place** – integer; identifier of place for personal purchase; list of acceptable places for personal purchase should be obtained via special XML request

As the answer for this query, for each input **article** element user gets one **article** element containing the result of the article import.

If import was run successfully, the result contains in **article** element six elements
(in some cases eight or nine elements)

- **order_number** – assigned number of the article for WE|DO
- **reference_number** – reference number of the article; either it was given by the customer or it was also assigned
- **barcode** – list of barcodes of all packages of article
- **sorting_code** – sorting code of article (for logistics purposes for WE|DO)
- **product_name** – label of WE|DO delivery product used for article (defined by WE|DO price list)
- **delivery_price, delivery_price_currency** – if authenticated user has access to delivery prices, XML answer contains two elements with price for deliery and it's currency
- **zpl** – zpl code for barcode print
- **code** – return code with value 0

In case the import wasn't successful, **article** element contains a tag **error** with error message and tag **code** with return code 1.

If automatic dispatch of articles to the WE|DO central system requested (**option** element set to **auto_complete**), the answer contains a **batch** element with the code, name of manifest importing given articles to the WE|DO. Element **batch** also contains URL for downloading of collection protocol. In exception condition, **batch** with imported articles cannot be send automatically because of overloading of WE|DO central system. In that case the **batch** element contains additional element **error** where this error is described and batch has to be send manually from AZ.

Example of a XML request:

```
<?xml version=“1.0‟ encoding=“utf-8“?>
<request name=“import_article“>
    <auth username=“test‟ password=“test“/>
    <option name=“transaction‟ value=“no“/>
    <option name=“auto_complete‟ value=“yes“/>
    <article>
        <receiver>
            <external_id>12345</external_id>
            <name>IN TIME spedice</name>
            <street>Dopravaku 723</street>
            <city>Praha 8</city>
            <postal_code>18400</postal_code>
            <state>CZ</state>
            <firstname>Marek</firstname>
            <surname>Pihrt</surname>
            <email>marek.pihrt@intime.cz</email>
            <phone>111111111</phone>
            <mobile>222222222</mobile>
            <fax>333333333</fax>
            <note>majitel spolecnosti</note>
        </receiver>
        <reference_number>999999</reference_number>
        <package_count>3</package_count>
        <package_number>123456</package_number>
        <package_number>123457</package_number>
        <package_number>123458</package_number>
        <weight>5,50</weight>
        <volumetric_weight>0,51</volumetric_weight>
        <value>1000</value>
        <comment>prosim co nejrychleji</comment>
        <additional_service name=“cash_on_delivery‟ value=“250,50“/>
        <additional_service name=“document_back‟ value=“yes“/>
        <additional_service name=“document_back_note‟ value=“kontr. OP“/>
        <additional_service name=“insurance‟ value=“yes“/>
        <additional_service name=“phone_notification‟ value=“yes“/>
        <additional_service name=“phone_notification_number‟ value=“601123456“/>
        <additional_service name=“sms_notification‟ value=“no“/>
        <additional_service name=“email_notification‟ value=“yes“/>
        <additional_service name=“email_notification_address‟ value=“novak@seznam.cz“/>
        <additional_service name=“carry‟ value=“no“/>
        <additional_service name=“loss‟ value=“no“/>
        <additional_service name=“pay_by_receiver‟ value=“no“/>
        <additional_service name=“authentication‟ value=“yes“/>
        <additional_service name=“authentication_note‟ value=“Prosim zkontrolovat OP“/>
    </article>
    <article>
        <receiver>
            <external_id>12345</external_id>
            <name>IN TIME spedice</name>
            <street>Dopravaku 723</street>
            <city>Praha 8</city>
            <postal_code>18400</postal_code>
            <state>CZ</state>
            <firstname>Marek</firstname>
            <surname>Pihrt</surname>
            <email>marek.pihrt@intime.cz</email>
        </receiver>
        <reference_number>999999</reference_number>
        <package_count>3</package_count>
        <weight>2,5</weight>
        <value>1000</value>
        <comment>prosim co nejrychleji</comment>
    </article>
</request>
```

</>

Example of a XML answer:

```
<?xml version="1.0" encoding="utf-8"?>
<response name="import_article">
    <article>
        <error>Zákazník nemá Odnos povolen, zásilku nelze uložit.</error>
        <code>1</code>
    </article>
    <article>
        <order_number>01200000072</order_number>
        <reference_number>042077</reference_number>
        <barcode>012S00000072*001003</barcode>
        <barcode>012S00000072*002003</barcode>
        <barcode>012S00000072*003003</barcode>
        <sorting_code>_NOCODE</sorting_code>
        <product_name>M-24-CZ</product_name>
        <delivery_price>100</delivery_price>
        <delivery_price_currency>CZK</delivery_price_currency>
        <code>0</code>
    </article>
    <batch>
        <id>123456</id>
        <number>IT-012-20100415012045</number>
        <protocol_url>http://www.intime.cz/protocol.html</protocol_url>
    </batch>
    <status>
        <code>0</code>
        <message>Požadavek byl úspěšně přijat.</message>
    </status>
</response>
```

2.

Change attributes of existing article in AZ

User can change some attributes of existing article in AZ.

request **element**

- **name** attribute must be set to edit_article

User can decide whether each package should be added separately or package adding must be run as a whole (in case that any package can't be added, no package will be added).

option **element**

- *transaction* **attribute**
  - **yes** – package adding is processed only if it successes for all packages
  - **no** – package adding is run for each single package

If authuser wants ZPL codes for each added package to print on special barcode pointer, he has to use option **zpl_code**.

- *zpl_code* **attribute**
  - **yes** – XML answer will include element **zpl** for each package containing ZPL code for printing on special barcode pointer
  - **no** – XML answer will not include element **zpl**

Now follows the list of articles to be changed

article **element**

- for each article to change attributes there is one **article** element
- each **article** element can contains this tags:
  - **order_number** – varchar(255); **mandatory field**; unique number of article to change attributes
  - **weight** – float (',' as delimiter, two decimal places max.); new weight of the article in kg
  - **value** – float (',' as delimiter, two decimal places max.); new value of the article
  - **cash_on_delivery** - float (',' as delimiter, two decimal places max.); new value of cash on delivery to get payed from article recipient

As the answer for this query, for each input **article** element user gets one **article** element containing the result of the changing attributes.

If changing attributes was run successfully, the result contains in **article** element three elements:

- **order_number** – unique number of the article
- **product_name** – label of WE|DO delivery product used for article
- **zpl** – zpl code for barcode printer
- **code** – return code with value 0

In case the changing attributes wasn't successful, **article** element contains a tag **error** with error message and tag **code** with return code 1.

If automatic dispatch of articles to the WE|DO central system requested (**option** element set to **auto_complete**), the answer contains a **batch** element with the code of manifest importing given articles to the WE|DO. In exception condition, batch with imported articles cannot be send automatically because of overloading of WE|DO central system. In that case the **batch** element contains additional element **error** where this error is described and batch has to be send manually from AZ.

</>
Example of a XML request:

```
<?xml version="1.0" encoding="utf-8"?>
<request name="edit_article">
    <auth username="test" password="test"/>
    <option name="auto_complete" value="yes"/>
    <option name="transaction" value="no"/>
    <option name="zpl_code" value="no"/>
    <article>
        <order_number>01200123456</order_number>
        <weight>70</weight>
        <value>100,50</value>
        <cash_on_delivery>100,50</cash_on_delivery>
    </article>
</request>
```

</>
Example of a XML answer:

```
<?xml version="1.0" encoding="utf-8"?>
<response name="edit_article">
    <article>
        <order_number>01200123456</order_number>
        <product_name>M-24-CZ</product_name>
        <code>0</code>
    </article>
    <batch>
        <id>12345</id>
        <number>IT-012-20110101235959</number>
    </batch>
    <status>
        <code>0</code>
        <message>Požadavek byl úspěšně přijat.</message>
    </status>
</response>
```

3.

Add package to existing article in AZ

User can add any number of packages to existing article in AZ.

request **element**

- **name** attribute must be set to article_add_package

User can decide whether each package should be added separately or package adding must be run as a whole (in case that any package can't be added, no package will be added).

option **element**

- *transaction* **attribute**
    - **yes** – package adding is processed only if it successes for all packages
    - **no** – package adding is run for each single package

If authuser wants ZPL codes for each added package to print on special barcode pointer, he has to use option **zpl_code**.

- *zpl_code* **attribute**
    - **yes** – XML answer will include element **zpl** for each package containing ZPL code for printing on special barcode printer
    - **no** – XML answer will not include element **zpl**

Now follows the list of articles to which packages will be added:

article **element**

- for each article to add package there is one article element
- each **article** element can contains this tags:
    - **order_number** – varchar(255); **mandatory field**; unique number of article to which packages will added
    - **package_count** – integer; **mandatory field**; number of parcels to add to article
    - **package_number** – varchar(50); optional customer package number; number of tags have to correspond to number of packages to add
    - **weight** – float (',' as delimiter, two decimal places max.); **mandatory field**; weight of the package in kg to add to total weight of article
    - **volumetric_weight** - float (',' as delimiter, two decimal places max.); volumetric weight of package in kg to add to total volumetric weight of article

As the answer for this query, for each input **article** element user gets one **article** element containing the result of the adding package.

If adding package was run successfully, the result contains in **article** element four elements:

- **order_number** – unique number of the article
- **barcode** – list of barcodes of added packages
- **product_name** – label of WE|DO delivery product used for article
- **zpl** – zpl code for barcode printer
- **code** – return code with value 0

In case the adding package wasn't successful, **article** element contains a tag error with **error** message and tag **code** with return code 1.

</>
Example of a XML request:

```
<?xml version="1.0" encoding="utf-8"?>
<request name="article_add_package">
    <auth username="test" password="test"/>
    <option name="transaction" value="yes"/>
    <option name="zpl_code" value="no"/>
    <article>
        <order_number>01200201451</order_number>
        <package_count>1</package_count>
        <package_number>123456</package_number>
        <weight>1</weight>
        <volumetric_weight>2</volumetric_weight>
    </article>
</request>
```

</>
Example of a XML answer:

```
<?xml version="1.0" encoding="utf-8"?>
<response name="article_add_package">
    <article>
        <order_number>01200201451</order_number>
        <barcode>012S00201451*004000</barcode>
        <product_name>M-24-CZ</product_name>
        <code>0</code>
    </article>
    <status>
        <code>0</code>
        <message>Požadavek byl úspěšně přijat.</message>
    </status>
</response>
```

4.

Remove package from existing article in AZ

User can remove any number of packages from existing article in AZ.

request **element**

- **name** attribute must be set to article_remove_package

User can decide whether each package should be removed separately or package removing must be run as a whole (in case that any package can't be removed, no package will be removed).

option **element**

- *transaction* **attribute**
  - **yes** – package removing is processed only if it successes for all packages
  - **no** – package removing is run for each single package

If authuser wants ZPL codes for remaining packages to print on special barcode pointer, he has to use option **zpl_code**.

- *zpl_code* **attribute**
  - **yes** – XML answer will include element **zpl** for each remaining package containing ZPL code for printing on special barcode printer
  - **no** – XML answer will not include element **zpl**

Now follows the list of articles from which packages will be removed:

article **element**

- for each article to remove package there is one article element
- each article element can contains this tags:
  - **order_number** – varchar(255); **mandatory field**; unique number of article from which packages will removed
  - **package_count** – integer; **mandatory field**; number of parcels to remove from article
  - **weight** – float (',' as delimiter, two decimal places max.); **mandatory field**; weight to substract from total weight of article; remaing weight have to be greater than zero
  - **volumetric_weight** - float (',' as delimiter, two decimal places max.); volumetric weight to substract from total volumetric weight of article

As the answer for this query, for each input **article** element user gets one **article** element containing the result of the removing package.

If removing package was run successfully, the result contains in **article** element four elements:

- **order_number** – unique number of the article
- **barcode** – list of barcodes of remaining packages
- **product_name** – label of WE|DO delivery product used for article
- **zpl** – zpl code for barcode printer
- **code** – return code with value 0

In case the removing package wasn't successful, **article** element contains a tag **error** with error message and tag **code** with return code 1.

</>
Example of a XML request:

```
<?xml version="1.0" encoding="utf-8"?>
<request name="article_remove_package">
    <auth username="test" password="test"/>
    <option name="transaction" value="yes"/>
    <option name="zpl_code" value="no"/>
    <article>
        <order_number>01200201451</order_number>
        <package_count>1</package_count>
        <weight>1</weight>
        <volumetric_weight>2</volumetric_weight>
    </article>
</request>
```

</>
Example of a XML answer:

```
<?xml version="1.0" encoding="utf-8"?>
<response name="article_remove_package">
    <article>
        <order_number>01200201451</order_number>
        <barcode>012S00201451*001000</barcode>
        <product_name>M-24-CZ</product_name>
        <code>0</code>
    </article>
    <status>
        <code>0</code>
        <message>Požadavek byl úspěšně přijat.</message>
    </status>
</response>
```

5. Get delivery status of existing article in AZ

User can get delivering status of existing article in AZ.

request **element**

- **name** attribute must be set to get_article

Now follows the list of articles to get delivering status for:

article **element**

- for each article to get delivering status there is one article element
- each **article** element can contains this tag:
    - **order_number** – varchar(255); **mandatory field**; unique number of article to get delivering status for

As the answer for this query, for each input **article** element user gets one **article** element containing the informations about store article.

If article exists, the result contains in **article** element eight (ten) elements:

- **order_number** – unique number of the article
- **reference_number** – customer number of the article
- **barcode** – list of barcodes of packages
- **sorting_code** – sorting code of the article (for WE|DO internal use)
- **product_name** – label of WE|DO delivery product used for article
- **delivery_price, delivery_price_currency** – if user have rights to see delivery price of article, the answer will contains these tag with price and currency
- **code** – return code with value 0
- **state** – actual delivering state of article; possible values are: CREATED, DELIVERING, DELIVERED, DELETED
- **state_time** – date and time of actual delivering state of article in format YYYY-MM-DD HH:mm:ss

In case the article doesn't exists, **article** element contains a tag **error** with error message and tag **code** with return code 1.

</>
Example of a XML request:

```
<?xml version="1.0" encoding="utf-8"?>
<request name="get_article">
    <auth username="test" password="test"/>
    <article>
        <order_number>01200000204</order_number>
    </article>
</request>
```

</>
Example of a XML answer:

```
<?xml version="1.0" encoding="utf-8"?>
<response name="get_article">
    <article>
        <order_number>01200000204</order_number>
        <code>0</code>
        <reference_number>999999</reference_number>
        <barcode>012S00000204*001000</barcode>
        <barcode>012S00000204*002000</barcode>
        <barcode>012S00000204*003000</barcode>
        <sorting_code>S110800</sorting_code>
        <product_name>M-24-CZ</product_name>
        <delivery_price>370</delivery_price>
        <delivery_price_currency>CZK</delivery_price_currency>
        <state>CREATED</state>
        <state_time>2013-02-18 21:41:16</state_time>
    </article>
    <status>
        <code>0</code>
        <message>Požadavek byl úspěšně přijat.</message>
    </status>
</response>
```

6. Delete existing article from AZ

User can delete any article from AZ.

request **element**

- **name** attribute must be set to delete_article

User can decide whether each article should be deleted separately or article deleting must be run as a whole (in case that any article can't be deleted, no article will be deleted).

option **element**

- *transaction* **attribute**
    - **yes** – article deleting is processed only if it successes for all articles
    - **no** – article removing is run for each single article

Now follows the list of articles to delete:

article **element**

- for each article to delete there is one article element
- each article element can contains this tag:
    - **order_number** – varchar(255); **mandatory field**; unique number of article to delete

As the answer for this query, for each input **article** element user gets one **article** element containing the result of the deleting article.

If article deleting was run successfully, the result contains in **article** element two elements:

- **order_number** – unique number of the article
- **code** – return code with value 0

In case the article deleting wasn't successful, **article** element contains a tag **error** with error message and tag **code** with return code 1.

Example of a XML request:

```
<?xml version=“1.0‟ encoding=“utf-8“?>
<request name=“delete_article“>
    <auth username=“test‟ password=“test“/>
    <option name=“transaction‟ value=“no“/>
    <article>
        <order_number>01200201460</order_number>
    </article>
    <article>
        <order_number>01200201449</order_number>
    </article>
</request>
```

Example of a XML answer:

```
<?xml version=“1.0‟ encoding=“utf-8“?>
<response name=“delete_article“>
    <article>
        <error>Neexistující zásilka.</error>
        <code>1</code>
    </article>
    <article>
        <order_number>01200201449</order_number>
        <code>0</code>
    </article>
    <status>
        <code>0</code>
        <message>Požadavek byl úspěšně přijat.</message>
    </status>
</response>
```

7.
Reserve of article collection

User can reserve article collection for delivery. Article collection should saved for specific date not earlier than next working day.

request **element**

- **name** attribute must be set to import_transportreservation

User can decide whether each article collection should be reserved separately or reservation of article collection must be run as a whole (in case that any reservation can't be saved, no reservation will be saved).

option **element**

- *transaction* **attribute**
    - **yes** – reservation of article collection is processed only if it successes for all collections
    - **no** – reservation of article collection is run for each single collection

If authuser is allowed to reserve article collection for more customers, he has to specify the customer of the current import.

- *customer* **attribute**
    - **value** – customer ID; the list of authorized customer IDs can be got from technical support of the WE|DO

If authuser is allowed to reserve article collection for more departments, he has to specify the department of the current import

- *department* **attribute**
    - **value** – department ID; the list of authorized departments can be got from technical support of the WE|DO

Now follows the list of article collections to reserve.

transportreservation **element**

- for each article collection there is one transportreservation element
- each **transportreservation** element can contains tag:
    - **date** – required date of collection; **mandatory field**

As the answer for this query, for each input **transportreservation** element user gets one **transportreservation** element containing the result of the collection reservation.

If article collection was run successfully, the result contains **in transportreservation** element three elements:

- **id** – internal identification of reservation
- **date** – date of article collection
- **code** – return code with value 0

In case the article collection wasn't successful, **transportreservation** element contains a tag **error** with error message and tag **code** with return code 1.

</>

Example of a XML request:

```
<?xml version="1.0" encoding="utf-8"?>
<request name="import_transportreservation">
    <auth username="test" password="test"/>
    <option name="transaction" value="no"/>
    <transportreservation>
        <date>2014-11-11</date>
    </transportreservation>
</request>
```

</>

Example of a XML answer:

```
<?xml version="1.0" encoding="utf-8"?>
<response name="import_transportreservation">
    <transportreservation>
        <id>1234567</id>
        <date>2014-11-11</date>
        <code>0</code>
    </transportreservation>
    <status>
        <code>0</code>
        <message>Požadavek byl úspěšně přijat.</message>
    </status>
</response>
```

8.
Cancel reservation of article collection

User can cancel reservation of article collection for delivery. Reservation of article collection should canceled not later than working day before date of collection.

request **element**

- **name** attribute must be set to delete_transportreservation

User can decide whether cancelation of each reservation of article collection should be done separately or cancelation of reservation of article collection must be run as a whole (in case that any reservation can't be canceled, no reservation will be canceled).

option **element**

- *transaction* **attribute**
  - **yes** – cancel of reservation of article collection is processed only if it successes for all reservations
  - **no** – cancel of reservation of article collection is run for each single reservation

If authuser is allowed to reserve article collection for more customers, he has to specify the customer of the current import.

- *customer* **attribute**
  - **value** – customer ID; the list of authorized customer IDs can be got from technical support of the WE|DO

If authuser is allowed to reserve article collection for more departments, he has to specify the department of the current import

- *department* **attribute**
  - **value** – department ID; the list of authorized departments can be got from technical support of the WE|DO

Now follows the list of reservations of article collections to cancel.

transportreservation **element**

- for each article collection there is one transportreservation element
- each element **transportreservation** can contain these tags (one of then si required):
  - **id** – internal identication of reservation
  - **date** – required date of collection

As the answer for this query, for each input **transportreservation** element user gets one **transportreservation** element containing the result of the cancel of collection reservation.

If cancel of reservation of article collection was run successfully, the result contains in **transportreservation** element three elements:

- **id** – internal identification of reservation
- **date** – date of article collection
- **code** – return code with value 0

In case cancel of reservation of article collection wasn't successful, **transportreservation** element contains a tag error with **error** message and tag **code** with return code 1.

</>
Example of a XML request:

```
<?xml version="1.0" encoding="utf-8"?>
<request name="delete_transportreservation">
    <auth username="test" password="test"/>
    <option name="transaction" value="no"/>
    <transportreservation>
        <date>2014-11-11</date>
    </transportreservation>
</request>
```

</>
Example of XML answer:

```
<?xml version="1.0" encoding="utf-8"?>
<response name="delete_transportreservation">
    <transportreservation>
        <id>1234567</id>
        <date>2014-11-11</date>
        <code>0</code>
    </transportreservation>
    <status>
        <code>0</code>
        <message>Požadavek byl úspěšně přijat.</message>
    </status>
</response>
```

9. Send articles in AZ to WE|DO

User can send selected articles in AZ to WE|DO for processing.

request **element**

- **name** attribute must be set to complete_article

User can decide whether sending articles should be run separately or sending articles must be run as a whole (in case that any article can't be sent, no article will be sent).

option **element**

- *transaction* **attribute**
  - **yes** – sending articles is processed only if it successes for all articles
  - **no** – sending articles is run for each single article

Now follows the list of articles to send to WE|DO.

article **element**

- for each article to send to WE|DO there is one **article** element
- each **article** element can contains this tags:
  - **order_number** – varchar(255); **mandatory field**; unique number of article to send to WE|DO

As the answer for this query, for each input **article** element user gets one **article** element containing the result of the sending article.

If sending article was run successfully, the result contains in **article** element two elements:

- **order_number** – unique number of the article
- **code** – return code with value 0

There are three tags in element **batch**:

- **id** – internal identification of batch
- **number** – name of batch
- **protocol_url** – URL for delivery protocol download

In case the sending article wasn't successful, **article** element contains a tag **error** with error message and tag **code** with return code 1.

</>
Example of a XML request:

```
<?xml version=“1.0‟ encoding=“utf-8“?>
<request name=“complete_article“>
    <auth username=“test‟ password=“test“/>
    <option name=“transaction‟ value=“no“/>
    <article>
        <order_number>01200201460</order_number>
    </article>
    <article>
        <order_number>01200201449</order_number>
    </article>
    <article>
        <order_number>01200201450</order_number>
    </article>
</request>
```

</>
Example of a XML answer:

```
<?xml version=“1.0‟ encoding=“utf-8“?>
<response name=“complete_article“>
    <article>
        <error>Neexistující zásilka.</error>
        <code>1</code>
    </article>
    <article>
        <order_number>01200201449</order_number>
        <code>0</code>
    </article>
    <article>
        <order_number>01200201450</order_number>
        <code>0</code>
    </article>
    <batch>
        <id>12345</id>
        <number>IT-012-20110316205019</number>
        <protocol_url>http://www.intime.cz/protocol.html</protocol_url>
    </batch>
    <status>
        <code>0</code>
        <message>Požadavek byl úspěšně přijat.</message>
    </status>
</response>
```

10.

Get list of Poštomat boxes

User can get list of acceptable Poštomat boxes.

request **element**

- **name** attribute must be set to get_packmachine

option **element**

- **atribut** *state*
    - **CZ** – boxes for Czech republic will be included in answer
    - **SK** – boxes for Slovak republic will be included in answer

If this element will not be present in request, response will contain boxes for Czech republic.

As the answer for this query user will retrieve list of acceptable Poštomat boxes.
Each box is represented by element **packmachine** containing box attributes:

- **id** – identifier of box
- **name** – name of box
- **street** – address street of box
- **number** – address house number of box
- **town** – address town of box
- **postal_code** – address postal code of box
- **gps_lat** – GPS latitude coordinate
- **gps_lng** – GPS longitude coordinate
- **location** – description of box location

</>
Example of a XML request:

```
<?xml version=“1.0” encoding=“utf-8”?>
<request name=“get_packmachine”>
    <auth username=“test” password=“test”/>
    <option name=“state” value=“CZ”/>
</request>
```

</>
Example of a XML answer:

```
<?xml version=“1.0” encoding=“utf-8”?>
<response name=“get_packmachine”>
    <packmachine>
        <id>2279</id>
        <name>CZBRN0916</name>
        <street>Kamenice</street>
        <number>745/1</number>
        <town>Brno</town>
        <postal_code>625 00</postal_code>
        <gps_lat>49.20916</gps_lat>
        <gps_lng>16.68712</gps_lng>
        <location>sousední budova napravo od Albertu</location>
    </packmachine>
    <status>
        <code>0</code>
        <message>Požadavek byl úspěšně přijat.</message>
    </status>
</response>
```

11.

Get list of WE|DO products

User can get list of acceptable products (delivery types) for import of articles.

request **element**

- **name** attribute must be set to get_product

As the answer for this query user will retrieve list of acceptable products for import of articles.
Each place is represented by element **product** containing place attributes:

- **id** – identifier of product
- **name** – name of product
- **short_name** – short name of product, this attribute is using in request import article
- **state_code** - code of state acceptable for product
- **weight_to** – maximal weight of package acceptable for product; weight of package is average weight from total article weight and number of packages in article

</>

Example of a XML request:

```
<?xml version="1.0" encoding="utf-8"?>
<request name="get_product">
    <auth username="test" password="test"/>
</request>
```

</>

Example of a XML answer:

```
<?xml version="1.0" encoding="utf-8"?>
<response name="get_product">
    <product>
        <id>51</id>
        <name>Small Colli 24-CZ - hmotnost do 1kg</name>
        <short_name>S-24-CZ</short_name>
        <state_code>CZ</state_code>
        <weight_to>1</weight_to>
    </product>
    <status>
        <code>0</code>
        <message>Požadavek byl úspěšně přijat.</message>
    </status>
</response>
```

12.
Get list of places for personal purchase

User can get list of acceptable places for personal purchase.

request **element**

- **name** attribute must be set to get_takeover_place

As the answer for this query user will retrieve list of acceptable places for personal purchase.
Each place is represented by element **takeover_place** containing place attributes:

- **id** – identifier of place
- **name** – name of place
- **address** – address of place

</>
Example of a XML request:

```
<?xml version="1.0" encoding="utf-8"?>
<request name="get_takeover_place">
    <auth username="test" password="test"/>
</request>
```

</>
Example of a XML answer:

```
<?xml version="1.0" encoding="utf-8"?>
<response name="get_takeover_place">
    <takeover_place>
        <id>1</id>
        <name>IN TIME Praha</name>
        <address>Ke Zdibsku 1, Praha 9</address>
    </takeover_place>
    <status>
        <code>0</code>
        <message>Požadavek byl úspěšně přijat.</message>
    </status>
</response>
```

13.

Get list of pup branches

User can get list of acceptable pup branches.

request **element**

- **name** attribute must be set to get_pup_branch

option **element**

- **atribut** *state*
    - **CZ** – pup branches for Czech republic will be included in answer
    - **SK** – pup branches for Slovak republic will be included in answer

If this element will not be present in request, response will contain pup branches for Czech republic.

As the answer for this query user will retrieve list of acceptable pup branches.
Each box is represented by element **pup_branch** containing pup branch attributes:

- **id** – identifier of pup branch
- **name** – name of pup branch
- **street** – address street of pup branch
- **number** – address house number of pup branch
- **town** – address town of pup branch
- **postal_code** – address postal code of pup branch
- **gps_lat** – GPS latitude coordinate
- **gps_lng** – GPS longitude coordinate
- **location** – description of pup branch location

Example of a XML request:

```
<?xml version=“1.0‟ encoding=“utf-8“?>
<request name=“get_pup_branch“>
    <auth username=“test‟ password=“test“/>
    <option name=“state‟ value=“CZ“/>
</request>
```

Example of a XML answer:

```
<?xml version=“1.0‟ encoding=“utf-8“?>
<response name=“get_pup_branch“>
    <pup_branch>
        <id>5</id>
        <name>VM00005</name>
        <street>Boleslavská</street>
        <number>1452</number>
        <town>Brandýs nad Labem-Stará Boleslav</town>
        <postal_code>250 01</postal_code>
        <gps_lat>50.197725</gps_lat>
        <gps_lng>14.678762</gps_lng>
        <location>trafika u autobusového nádraží</location>
    </pup_branch>
    <status>
        <code>0</code>
        <message>Požadavek byl úspěšně přijat.</message>
    </status>
</response>
```

Module store

WE|DO customers with contract of storing products in WE|DO stores can use following special requests for operations with stored products.

14. Import of receipt

User can create new receipt for specified products, which will be delivered to WE|DO stores. The receipt will be processed by WE|DO operators and after confirmation of delivery all specified products will be available for delivery to end-customers.

request **element**

- **name** attribute must be set to import_receipt

User can decide whether each receipt should be imported separately or import must be run as a whole (in case that any receipt can't be imported, no receipt will be imported).

option **element**

- *transaction* **attribute**
    - **yes** – import is processed only if it successes for all receipts
    - **no** – import is run for each single receipt

If authuser is allowed to import receipts for more customers, he has to specify the customer of the current import.

- *customer* **attribute**
    - **value** – customer ID; the list of authorized customer IDs can be got from technical support of the WE|DO

If authuser is allowed to import receipts for more departments, he has to specify the department of the current import

- *department* **attribute**
    - **value** – department ID; the list of authorized departments can be got from technical support of the WE|DO

Now follows the list of receipts to import.

receipt **element**

- for each imported receipt there is one receipt element
- each **receipt** element contains these tags:
    - **external_id** – varchar(255), **mandatory field**, unique customer number of receipt, this number will be stored in AZ
    - **date** – date in format DAY.MONTH.YEAR; date of receipt
    - **comment** – receipt comment
    - **item** – list of receipt items; for each receipt item there is one **item** element
        - **external_id** – varchar(255), **mandatory field**, unique code of store item in customer systems, this code will be stored in AZ
        - **quantity** – int, **mandatory field**, item quantity on receipt
        - **name** – varchar(255), label of store item, **mandatory field** in case of creating new item in AZ
        - **provider** – varchar(255), provider of store item, optional field in case of creating new item in AZ

As the answer for this query, for each input receipt element user gets one **receipt** element containing the result of the receipt import.

If import was run successfully, the result contains in **receipt** element three elements:

- **receipt_id** – ID of created receipt
- **external_id** – unique number of receipt in request XML
- **code** – return code with value 0

In case the import wasn't successful, **receipt** element contains the tag **error** with error message and tag **code** with return value 1.

Example of XML request:

```
<?xml version=“1.0” encoding=“utf-8”?>
<request name=“import_receipt”>
    <auth username=“test” password=“test”/>
    <option name=“transaction” value=“yes”/>
    <receipt>
        <external_id>12345</external_id>
        <date>1.1.2012</date>
        <comment>poznamka</comment>
        <item>
            <external_id>ext1</external_id>
            <quantity>30</quantity>
        </item>
        <item>
            <external_id>ext2</external_id>
            <quantity>20</quantity>
        </item>
    </receipt>
</request>
```

Example of XML answer:

```
<?xml version=“1.0” encoding=“utf-8”?>
<response name=“import_receipt”>
    <receipt>
        <recept_id>1</recept_id>
        <external_id>12345</external_id>
        <code>0</code>
    </receipt>
    <status>
        <code>0</code>
        <message>Požadavek byl úspěšně přijat.</message>
    </status>
</response>
```

Example of XML request (with new store item creation)

```
<?xml version=“1.0” encoding=“utf-8”?>
<request name=“import_receipt”>
    <auth username=“test” password=“test”/>
    <option name=“transaction” value=“yes”/>
    <receipt>
        <external_id>12345</external_id>
        <date>1.1.2012</date>
        <comment>poznamka</comment>
        <item>
            <external_id>ext1</external_id>
            <quantity>30</quantity>
        </item>
        <item>
            <external_id>ext2</external_id>
            <quantity>20</quantity>
        </item>
        <item>
            <name>nová položka</name>
            <provider>vyrobce neznamý</provider>
            <external_id>ext3</external_id>
            <quantity>20</quantity>
        </item>
    </receipt>
</request>
```

15. Get status of existing receipt

User can get status of existing receipt at store.

request **element**

- **name** attribute must be set to get_receipt

If authuser is allowed to import receipts for more departments, he has to specify the department of the current import

- *department* **attribute**
  - **value** – department ID; the list of authorized departments can be got from technical support of the WE|DO

Now follows the list of receipts to get status for.

receipt **element**

- for each receipt to get status for there is one receipt element
- each **receipt** element contains one of these tags:
  - **id** – integer; internal ID of receipt in AZ; this ID is assigned to receipt during creating of receipt
  - **external_id** – varchar(255); unique customer number of receipt

As the answer for this query, for each input receipt element user gets one **receipt** element containing the informations about receipt.

If receipt exists, the result contains in **receipt** element nine elements:

- **id** – internal ID of receipt in WE|DO system
- **external_id** – unique number of receipt in request XML
- **code** – return code with value 0
- **date** – date of receipt
- **supplier** – supplier of receipt
- **note** – note of receipt
- **state** – status of receipt; value **CREATED** means receipt is not stocked; value **STOCKED** means receipt is stocked; value **DELETED** means receipt is deleted
- **item** – list of receipt items (**name** – name of receipt item, **external_id** – customer identifier of item, **quantity** – number of pieces of item on receipt)
- **total_quantity** – total number of peaces on receipt

In case the receipt doesn't exist, receipt element contains the tag **error** with error message and tag **code** with return value 1.

&lt;/&gt;

Example of XML request:

```
<?xml version="1.0" encoding="utf-8"?>
<request name="get_receipt">
    <auth username="test" password="test"/>
    <receipt>
        <external_id>9999</external_id>
    </receipt>
</request>
```

&lt;/&gt;

Example of XML answer:

```
<?xml version="1.0" encoding="utf-8"?>
<response name="get_receipt">
    <receipt>
        <id>656</id>
        <external_id>9999</external_id>
        <code>0</code>
        <date>2012-01-01</date>
        <supplier>Dodavatel #1</supplier>
        <note>poznamka</note>
        <state>STOCKED</state>
        <item>
            <name>Testovaci polozka</name>
            <external_id>EXT_#1</external_id>
            <quantity>3</quantity>
        </item>
        <total_quantity>3</total_quantity>
    </receipt>
    <status>
        <code>0</code>
        <message>Požadavek byl úspěšně přijat.</message>
    </status>
</response>
```

16. Delete existing receipt

User can get delete existing receipt at store.

request **element**

- **name** attribute must be set to delete_receipt

User can decide whether each receipt should be deleted separately or import must be run as a whole (in case that any receipt can't be deleted, no receipt will be deleted).

option **element**

- *transaction* **attribute**
  - **yes** – delete is processed only if it successes for all receipts
  - **no** – delete is run for each single receipt

If authuser is allowed to import receipts for more departments, he has to specify the department of the current delete

- *department* **attribute**
  - **value** –department ID; the list of authorized departments can be got from technical support of the WE|DO

Now follows the list of receipts to delete.

receipt **element**

- for each receipt to delete there is one receipt element
- each **receipt** element contains one of these tags:
  - **id** – integer; internal ID of receipt in AZ; this ID is assigned to receipt during creating of receipt
  - **external_id** – varchar(255); unique customer number of receipt

As the answer for this query, for each input receipt element user gets one **receipt** element containing the the result of deleting receipt.

If receipt exists, the result contains in **receipt** element three elements:

- **id** – internal ID of receipt in WE|DO system
- **external_id** – unique number of receipt in request XML
- **code** – return code with value 0

In case the receipt doesn't exist, **receipt** element contains the tag error with **error** message and tag **code** with return value 1.

Example of XML request:

```
<?xml version=“1.0“ encoding=“utf-8“?>
<request name=“delete_receipt“>
    <auth username=“test“ password=“test“/>
    <receipt>
        <external_id>9999</external_id>
    </receipt>
</request>
```

Example of XML answer:

```
<?xml version=“1.0“ encoding=“utf-8“?>
<response name=“delete_receipt“>
    <receipt>
        <id>656</id>
        <external_id>9999</external_id>
        <code>0</code>
    </receipt>
    <status>
        <code>0</code>
        <message>Požadavek byl úspěšně přijat.</message>
    </status>
</response>
```

17.
Import of reservation (personal purchase)

User can create new reservation for specified products, which will be delivered to end-customers. The reservation will be processed by WE|DO operators and after confirmation will be ready for personal purchase by end-customers.

request **element**

- **name** attribute must be set to import_reservation

User can decide whether each reservation should be imported separately or import must be run as a whole (in case that any reservation can't be imported, no reservation will be imported).

option **element**

- *transaction* **attribute**
  - **yes** – import is processed only if it successes for all reservations
  - **no** – import is run for each single reservation

If authuser is allowed to import reservations for more customers, he has to specify the customer of the current import.

- *customer* **attribute**
  - **value** – customer ID; the list of authorized customer IDs can be got from technical support of the WE|DO

If authuser is allowed to import reservations for more departments, he has to specify the department of the current import

- *department* **attribute**
  - **value** – department ID; the list of authorized departments can be got from technical support of the WE|DO

Now follows the list of reservations to import.

reservation **element**

- for each imported reservation there is one reservation element
- each **reservation** element contains these tags:
  - **external_id** – archar(255), **mandatory field**, unique customer number of receipt, this number will be stored in AZ
  - **comment** – reservation comment
  - **receiver** – **mandatory field**, receiver of product on reservation
    - **external_id** – varchar(255); AZ automatically tries to find corresponding article recipient according to this ID; if found, it is updated with data listed in the article; otherwise the new recipient is created

        - **name** – varchar(100); **mandatory field**; recipient name
        - **street** – varchar(100); **mandatory field**; street + land registry number/house number
        - **city** - varchar(50); **mandatory field**; city
        - **postal_code** - varchar(5); **mandatory field**; postal code
        - **state** – enum(CZ, SK); **mandatory field**; country
        - **firstname** - varchar(30); first name
        - **surname** - varchar(30); surname
        - **email** – varchar(50); email address
        - **phone** – char(15); phone
        - **mobile** – char(15); mobile phone
        - **fax** – char(15); fax
        - **note** – varchar(75); note
- **item** – list of reservation store items, for each reservation item there is one **item** element
    - **external_id** – varchar(255), **mandatory field**, unique code of store item in customer systems, this code is stored in AZ
    - **quantity** – int, **mandatory fielad**, quantity of store item on reservation
    - **comment** – reservation item comment

As the answer for this query, for each input reservation element user gets one **reservation** element containing the result of the reservation import.

If import was run successfully, the result contains in **reservation** element three elements:

- **reservation_id** – ID of created reservation
- **external_id** – unique number of reservation in request XML
- **code** – return code with value 0

In case the import wasn't successful, **reservation** element contains the tag **error** with error message and tag **code** with return value 1.

</>
Example of XML request:

```
<?xml version=“1.0‟ encoding=“utf-8“?>
<request name=“import_reservation“>
    <auth username=“test‟ password=“test“/>
    <option name=“transaction‟ value=“yes“/>
    <reservation>
        <external_id>12345</external_id>
        <receiver>
            <external_id>EXT_1</external_id>
            <name>IN TIME spedice</name>
            <street>Dopraváků 723</street>
            <city>Praha 8</city>
            <postal_code>18400</postal_code>
            <state>CZ</state>
            <firstname>Josef</firstname>
            <surname>Novak</surname>
            <email>pepa@email.cz</email>
            <phone>111111111</phone>
            <mobile>222222222</mobile>
            <fax>333333333</fax>
            <note>Je doma vecer</note>
        </receiver>
        <comment>poznamka</comment>
        <item>
            <external_id>ext1</external_id>
            <quantity>3</quantity>
            <comment>opatrne krehke</comment>
        </item>
        <item>
            <external_id>ext2</external_id>
            <quantity>3</quantity>
            <comment>hezky zabalit</comment>
        </item>
    </reservation>
</request>
```

</>
Example of XML answer:

```
<?xml version=“1.0‟ encoding=“utf-8“?>
<response name=“import_reservation“>
    <reservation>
        <reservation_id>1</reservation_id>
        <external_id>12345</external_id>
        <code>0</code>
    </reservation>
    <status>
        <code>0</code>
        <message>Požadavek byl úspěšně přijat.</message>
    </status>
</response>
```

18. Reservation import to AZ (WE|DO expedition)

If customer wants to create reservation to be delivered by WE|DO expedition it's needed to create article for reservation during import of reservation. Article parameters are needed to be specified in XML request. In this case customer should use combination of request for creating article and request for creating reservation. Both requests are described in sections **Import of article** and **Import of reservation (personal purchase)**.

Combination of these two requests will be encapsulated in **Import of article** request and elements for creating reservation will be included. **Reservation** element wil not include **receiver** element as it specified in **article** element.

Example of XML request:

```
<?xml version=“1.0‟ encoding=“utf-8“?>
<request name=“import_article“>
    <auth username=“test‟ password=“test“/>
    <option name=“transaction‟ value=“yes“/>
    <option name=“zpl_code‟ value=“yes“/>
    <article>
        <receiver>
            <external_id>12345</external_id>
            <name>IN TIME spedice</name>
            <street>Dopraváků 723</street>
            <city>Praha 8</city>
            <postal_code>18400</postal_code>
            <state>CZ</state>
            <firstname>Josef</firstname>
            <surname>Novak</surname>
            <email>jn@email.cz</email>
            <phone>111111111</phone>
            <mobile>222222222</mobile>
            <fax>333333333</fax>
            <note>dulezity zakaznik</note>
        </receiver>
        <reference_number>999999</reference_number>
        <reference_number2>111111</reference_number2>
        <package_count>3</package_count>
        <weight>5,50</weight>
        <volumetric_weight>0,51</volumetric_weight>
        <value>1000</value>
        <additive>yes</additive>
        <comment>prosím co nejrychleji</comment>
        <additional_service name=“cash_on_delivery‟ value=“250,50“/>
        <additional_service name=“sms_notification‟ value=“yes“/>
        <additional_service name=“sms_notification_number‟ value=“+420601001122“/>
        <reservation>
            <external_id>12345</external_id>
            <comment>poznamka</comment>
            <item>
                <external_id>ext1</external_id>
                <quantity>3</quantity>
                <comment>opatrne krehke</comment>
            </item>
            <item>
                <external_id>ext2</external_id>
                <quantity>3</quantity>
                <comment>hezky zabalit</comment>
            </item>
        </reservation>
    </article>
</request>
```

</>

Example of XML answeri:

```
<?xml version="1.0" encoding="utf-8"?>
<response name="import_article">
    <article>
        <order_number>01200000072</order_number>
        <reference_number>999999</reference_number>
        <reservation_id>1</reservation_id>
        <barcode>012S00000072*001003</barcode>
        <barcode>012S00000072*002003</barcode>
        <barcode>012S00000072*003003</barcode>
        <sorting_code>_NOCODE</sorting_code>
        <product_name>M-24-CZ</product_name>
        <code>0</code>
    </article>
    <status>
        <code>0</code>
        <message>Požadavek byl úspěšně přijat.</message>
    </status>
</response>
```

19. Get status of existing reservation

User can get status of existing reservation at store.

request **element**

- **name** attribute must be set to get_reservation

If authuser is allowed to import reservations for more departments, he has to specify the department of the current import

- *department* **attribute**
    - **value** – department ID; the list of authorized departments can be got from technical support of the WE|DO

If authuser wants ZPL codes for packages to print on special barcode pointer, he has to use option **zpl_code**.

- *zpl_code* **attribute**
    - **yes** – XML answer will include element **zpl** for each package containing ZPL code for printing on special barcode printer
    - **no** – XML answer will not include element **zpl**

Now follows the list of reservations to get status for.

reservation **element**

- for each reservation to get status for there is one reservation element
- each **reservation** element contains one of these tags:
    - **id** – integer; internal ID of reservation in AZ; this ID is assigned to reservation during creating of reservation
    - **external_id** – varchar(255); unique customer number of reservation

As the answer for this query, for each input reservation element user gets one **reservation** element containing the informations about reservation.

If reservation exists, the result contains in **reservation** element nine elements:

- **id** – internal ID of reservation in WE|DO system
- **external_id** – unique number of reservation from request XML
- **code** – return code with value 0
- **note** – note of reservation

- **state** – status of reservation; value **CREATED** means reservation is not processed; value **REALISED** means reservation is processed; value **DELETED** means reservation is deleted
- **state_timestamp** – date and time of status of reservation
- **item** – list of reservation items (**name** – name of reservation item, **external_id** – customer identifier of item, **quantity** – number of pieces of item on reservation)
- **article** – detail of WE|DO article correspoding to reservation for delivery to end-user (**order_number** – WE|DO number of article; **reference_number** – customer number of article; **package_count** – number of packages in article; **bar_code** – list of package barcodes; **zpl_code** –list of ZPL codes of packages for barcode printer)
- **total_quantity** – total number of peaces on reservation

In case the reservation doesn't exist, **reservation** element contains the tag **error** with error message and tag **code** with return value 1.

</>
Example of XML request:

```
<?xml version="1.0" encoding="utf-8"?>
<request name="get_reservation">
    <auth username="test" password="test"/>
    <receipt>
        <external_id>9999</external_id>
    </receipt>
</request>
```

</>
Example of XML answer:

```
<?xml version="1.0" encoding="utf-8"?>
<response name="get_reservation">
    <receipt>
        <id>656</id>
        <external_id>9999</external_id>
        <code>0</code>
        <note>poznamka</note>
        <state>REALISED</state>
        <state_timestamp>2013-01-30 22:58:53</state_timestamp>
        <item>
            <name>Testovaci polozka</name>
            <external_id>EXT_#1</external_id>
            <quantity>3</quantity>
        </item>
        <article>
            <order_number>11101119915</order_number>
            <reference_number>999999</reference_number>
            <package_count>3</package_count>
            <bar_code>111S01119915*001000</bar_code>
            <bar_code>111S01119915*002000</bar_code>
            <bar_code>111S01119915*003000</bar_code>
        </article>
        <total_quantity>3</total_quantity>
    </receipt>
    <status>
        <code>0</code>
        <message>Požadavek byl úspěšně přijat.</message>
    </status>
</response>
```

20.
Delete existing reservation

User can get delete existing reservation at store.

request **element**

- **name** attribute must be set to delete_reservation

User can decide whether each reservation should be deleted separately or import must be run as a whole (in case that any reservation can't be deleted, no reservation will be deleted).

option **element**

- *transaction* **attribute**
  - **yes** – delete is processed only if it successes for all reservations
  - **no** – delete is run for each single reservation

If authuser is allowed to import reservations for more departments, he has to specify the department of the current delete

- *department* **attribute**
  - **value** – department ID; the list of authorized departments can be got from technical support of the WE|DO

Now follows the list of reservations to delete.

reservation **element**

- for each reservation to delete there is one reservation element
- each **reservation** element contains one of these tags:
  - **id** – integer; internal ID of reservation in AZ; this ID is assigned to reservation during creating of reservation
  - **external_id** – varchar(255); unique customer number of reservation

As the answer for this query, for each input reservation element user gets one **reservation** element containing the the result of deleting reservation.

If reservation exists, the result contains in **reservation** element three elements:

- **id** – internal ID of reservation in WE|DO system
- **external_id** – unique number of reservation in request XML
- **code** – return code with value 0

In case the receipt doesn't exist, **reservation** element contains the tag **error** with error message and tag **code** with return value 1.

</>
Example of XML request:

```
<?xml version=“1.0‟ encoding=“utf-8“?>
<request name=“delete_reservation“>
    <auth username=“test‟ password=“test“/>
    <reservation>
        <external_id>9999</external_id>
    </reservation>
</request>
```

</>
Example of XML answer:

```
<?xml version=“1.0‟ encoding=“utf-8“?>
<response name=“delete_reservation“>
    <reservation>
        <id>656</id>
        <external_id>9999</external_id>
        <code>0</code>
    </reservation>
    <status>
        <code>0</code>
        <message>Požadavek byl úspěšně přijat.</message>
    </status>
</response>
```

21.
Transfer between stores import

User can import transfer of stock items between store he is allowed to manage.

request **element**

- **name** attribute must be set to import_transfer

User can decide whether each transfer should be imported separately or import must be run as a whole (in case that any transfer can't be imported, no transfer will be imported).

option **element**

- *transaction* **attribute**
    - **yes** – delete is processed only if it successes for all transfers
    - **no** – delete is run for each single transfer

If authuser is allowed to import transfers for more departments, he has to specify the department of the current import

- *department* **attribute**
    - **value** – department ID; the list of authorized departments can be got from technical support of the WE|DO

Now follows the list of transfers to import.

transfer **element**

- for each transfer to import there is one transfer element
- each transfer element contains these tags:
    - **department** – integer; ID of department where items will be stored to
    - **external_id** – varchar(255); unique customer number of transfer
    - **comment** – note of transfer
    - **date** – date in DD.MM.YYYY format; date of transfers
    - **item** – list of transfer items; for each item the transfer will contains one **item** element
        - **external_id** – varchar(255); **mandatory field**; unique customer identification of item
        - **quantity** – integer; **mandatory field**; number of peaces of item on transfer
        - **comment** – note of transfer item

As the answer for this query, for each input transfer element user gets one **transfer** element containing the the result of transfer import.

If import was run successfully, the result contains in **transfer** element four elements:

- **external_id** – unique number of transfer in request XML
- **code** – return code with value 0
- **reservation_id** – ID of created reservation on original store
- **receipt_id** – ID of created receipt on destination store

In case the import wasn't run successfully, **transfer** element contains the tag **error** with error message and tag **code** with return value 1.

</> Example of XML request:

```
<?xml version="1.0" encoding="utf-8"?>
<request name="import_transfer">
    <auth username="test" password="test"/>
    <option name="transaction" value="yes"/>
    <transfer>
        <department>123</department>
        <external_id>12345</external_id>
        <date>2013-02-01</date>
        <comment>poznamka</comment>
        <item>
            <external_id>EXT_#1</external_id>
            <quantity>3</quantity>
        </item>
        <item>
            <external_id>EXT_#2</external_id>
            <quantity>3</quantity>
            <comment>Opatrne</comment>
        </item>
    </transfer>
</request>
```

</> Example of XML answer:

```
<?xml version="1.0" encoding="utf-8"?>
<response name="import_transfer">
    <transfer>
        <external_id>12345</external_id>
        <code>0</code>
        <reservation_id>1</reservation_id>
        <receipt_id>1</receipt_id>
    </transfer>
    <status>
        <code>0</code>
        <message>Požadavek byl úspěšně přijat.</message>
    </status>
</response>
```

WE DO

by allegro