Table of Contents

General description of the XML communication interface

General

If desired, a data exchange interface can be set up on each Directo base, enabling two-way data traffic. It is important to note that the interface is not present and available by default, but is configured on a case-by-case basis according to the needs of the specific task set. The installation of the interface involves fees and the amount of fee depends on the number of data flows in each direction (in and out). Therefor it is necessary to contact a Directo sales representative and agree on the exact needs and coordinate the cost of the project before starting work.

Regardless of the specifics of a particular interface project, there are certain principles that must be followed in any case.

- The data format is XML
- Data encoding is UTF-8
- The transport protocol is HTTPS (with a publicly recognized certificate)
- The POST method is used to access the interface
- Data traffic is always initiated by an external party, regardless of whether the data is requested or sent, the Directo interface is inactive
- The POST request is answered by an interface with a synchronous XML message, the processing of which may be important for the success of the process
- The structure of the data must be descriptive as an XSD schema, preferably a schema where the values are carried by attributes rather than elements
- Document numbers in Directos are of *integer* type, which means that the maximum possible number is *2147483647*

Access to interfaces

The interface is accessed via the POST method, for the interface URLs listed under form / urlencoded . You must use the appkey to gain access. The demo database has an appkey: 2852DD553B767B463C807ADB36B5BB2F. The appkey must be sent to the given URL in the KEY variable (similar to posting an html form) to get the result. There is an answer if the appkey is sent incorrectly or incorrectly

```
<result type="5" desc="Unauthorized"/>
```

Example for command line using cURL:

```
curl -v -d
"key=2852DD553B767B463C807ADB36B5BB2F&what=item&get=1&ts=01.01.2021&code=000
001" -H "Content-Type: application/x-www-form-urlencoded" -X POST
"https://login.directo.ee/xmlcore/demo_ee/xmlcore.asp"
```

Examples

Please consider the examples below as approximate only and it is important to note, that the specific interface to be created will have different **Shemas, URLs, Parameters, Restrictions, Reply messages and Keys** than given in the example.

Use this form to test: https://login.directo.ee/xmlcore/demo_ee/webupload.html

Querying data from Directo (or output)

Data retrieval is performed with the GET parameter in the POST method and at least three parameters are always met:

- `get=1`
- `what=ANDMELIIK`
- `appkey=PREDEFINEDKEY`

and possible additional parameters. The data type verbs used, key values, and additional data type-specific parameters are agreed during each specific installation.

A practical example

This type of output is often used by an webshop to download customer, product and product availability information from the Direct webshop.

The URLs in this example are for illustration purposes only. In any case, there is no guarantee that these URLs will actually match your query. If the query results in data, it is 100% fictitious and does not relate to any real company data.

Article output:

Antud näiteks kasutatakse verbi `what=item` ja `get=1`. Kõik filtrid on võimalikud, juhul kui vastav väli on skeemis olemas. Näiteks: „class“,„code“,„barcode“,„supplier“,„supplieritem“,„closed“,„ts“ Artiklite struktuuri kirjeldav skeem: In this example, the verbs `what=item` and `get=1` are used. All filters are possible if the corresponding field is present in the schema. For example, „class“, „code“, „barcode“, „supplier“, „supplieritem“, „closed“, „ts“
Schema describing Articles: https://login.directo.ee/xmlcore/demo_ee/ws_artiklid.xsd

Article request for changes (ts = Time stamp):
https://login.directo.ee/xmlcore/demo_ee/xmlcore.asp?what=item&get=1&ts=01.01.2021

The time filter can also use the time as follows: &ts=01.03.2021 12:30 / &ts=01.03.2021%2012:30

Query the entire product database:
https://login.directo.ee/xmlcore/demo_ee/xmlcore.asp?get=1&what=item

Query for an singel product by product code using additional parameter `code=0000`:

https://login.directo.ee/xmlcore/demo_ee/xmlcore.asp?get=1&what=item&code=0000

Customer output:

https://login.directo.ee/xmlcore/demo_ee/xmlcore.asp?get=1&what=customer&ts=01.01.2021

Sample filter options: „code“, „loyaltycard“, „regno“, „email“, „phone“, „closed“, „ts“.
Schema describing Customer: https://login.directo.ee/xmlcore/demo_ee/ws_kliendid.xsd

Output of quantity in stocks:

https://login.directo.ee/xmlcore/demo_ee/xmlcore.asp?get=1&what=stocklevel

Sample filter options: „code“, „stock“.
Schema describing quantity in stocks: https://login.directo.ee/xmlcore/demo_ee/ws_laoseis.xsd

Sending data to Directo (or input)

The data is sent using the POST method, in relation to the `form/urlencoded` interface URL.

Three form variables must be filled in:

- `put=1`
- `what=DATATYPE`
- `xmldata=<?xml version=„1.0“ encoding=„utf-8“?><PAYLOAD>`

In the input interface, XML must be posted to: https://login.directo.ee/xmlcore/demo_ee/xmlcore.asp
POST with variables „put=1“, „key=[appkey value]“, „what=order“ (or „what=customer“ or „what=item“) and „xmldata=[XML'i content]“
The xml input content must match the schema.

A practical example

This type of input is used, for example, when sales orders are generated in an external system that need to be reflected in Directos. In addition, there is a description of the Customer Scheme

The URLs and XML messages in this example are completely fictitious and cannot be used to test the service. Testing is always performed on the actual installation.

Sales order input interface:

The data is described in the scheme https://login.directo.ee/xmlcore/demo_ee/xml_IN_tellimused.xsd

An XML that conforms to the schema but does not use all the possible attributes might look like this:

```
<?xml version="1.0" encoding="utf-8"?>
<orders>
  <order number="1" customercode="customercode1"
customername="customername1" date="1900-01-01T01:01:01+02:00">
    <rows>
      <row item="item1" description="description1" price="1" quantity="1"
sum="1" rn="1" />
      <row item="item2" description="description2" price="0" quantity="1"
sum="0" rn="2" />
    </rows>
  </order>
 </orders>
```

The post is answered by an interface with a synchronous XML message that tells what happened. For example, the post was successful and sales order number 1 was accepted at Directo:

```
<?xml version="1.0" encoding="UTF-8" ?>
<results>
  <Result Type="0" Desc="OK" docid="1" doctype="ORDER" submit="Orders" />
</results>
```

If element `Result` attribute `Type` value is not 0, then there is an error. In which case the attribute `Desc` describes you what was the problem.

For example, order number 1 was declined because an order with that number already existed on Directos:

```
<?xml version="1.0" encoding="UTF-8" ?>
<results>
  <Result Type="1" Desc="Duplicate" docid="1" doctype="ORDER"
submit="Orders" />
</results>
```

There can be several orders in one batch. Each order will have its own `Result`:

```
<?xml version="1.0" encoding="UTF-8" ?>
<results>
  <Result Type="1" Desc="Duplicate" docid="1" doctype="ORDER"
submit="Orders" />
  <Result Type="0" Desc="OK" docid="2" doctype="ORDER" submit="Orders" />
</results>
```

Processing the reply message is important! If the message indicating a successful reception cannot be received, the sending must be considered unsuccessful and retried after an interval (not less than 60 seconds).

Customer input interface:

The data is described in the scheme https://login.directo.ee/xmlcore/demo_ee/xml_IN_kliendid.xsd.

Posting data is the same way as posting orders. An XML that conforms to the schema, but does not use all the possible attributes might look like this:

```
<?xml version="1.0" encoding="utf-8"?>
<customers>
  <customer code="Customercode1" name="Name" address1="address11"
address2="address21" address3="address31" email="email1" regno="regno1"
class="class1" type="1" >
  </customer>
 </customers>
```

PHP examples

Output

```
$url = 'https://login.directo.ee/xmlcore/demo_ee/xmlcore.asp';
$key = '2852DD553B767B463C807ADB36B5BB2F';
$ch = curl_init();
curl_setopt($ch, CURLOPT_HEADER, 0);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_FOLLOWLOCATION, true);
curl_setopt($ch, CURLOPT_URL, $url );
curl_setopt($ch, CURLOPT_POSTFIELDS, 'key='.urlencode($key) .
'&what=item&get=1&ts=01.01.2021&code=000001' );
curl_setopt($ch, CURLOPT_POST, 1);

$result = curl_exec($ch);

header('Content-Type: text/html; charset=utf-8');

print '<textarea cols=80 rows=10>' . $result . '</textarea>';

$data = new SimpleXMLElement($result);

print '<pre>';
print_r($data);
foreach ($data->item as $item) {
    print 'Code:'.$item['code'] . chr(9) .  '<b>' . $item['name'] . '</b>' .
chr(9) .  $item['class'] . '<br>Datafields:<br>';
    foreach ($item->datafields->data as $df ) {
        print chr(9) . $df['code'] . ':' . $df['content'] . '<br>';
    }
}
```

Input

```
$url = 'https://login.directo.ee/xmlcore/demo_ee/xmlcore.asp';
```

```
$key = '2852DD553B767B463C807ADB36B5BB2F';

$xmldata = <<<XML
<?xml version="1.0" encoding="utf-8"?>
<items>
  <item code="LMP0001" name="Lambikas" class="LAMP" barcode="123456789"
salesprice="11.25" description="See on suur lamp" >
    <datafields>
      <data code="ART_VARV" content="Punane" />
      <data code="VEEBIS" content="Jah" />
    </datafields>
  </item>
</items>
XML;


$ch = curl_init();

curl_setopt($ch, CURLOPT_HEADER, 0);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_FOLLOWLOCATION, true);
curl_setopt($ch, CURLOPT_URL, $url );
curl_setopt($ch, CURLOPT_POST, 1);
curl_setopt($ch, CURLOPT_POSTFIELDS,
'key='.urlencode($key).'&xmldata='.urlencode($xmldata).'&put=1&what=item' );

$result = curl_exec($ch);
curl_close($ch);

echo '<pre>';

print '</pre><textarea cols=80 rows=10>' . $result . '</textarea>';
```

Postman

Output and Input for the Postman environment can be downloaded here and then imported to Postman as a Collection and tested there.