Alletiders API

Providers (Attractions)

Version 1.3

Version: 1.3
Date: 2014.09.15
Main Author: kje@combine.dk

Document History

Date	Version	Initials	Description
2013.01.29	0.1	kje	Document created, first draft
2013.02.11	0.2	kje	Offline sales, OrderLine, API interface protocol
2013.02.27	0.3	kje	Adjustments for Tilpasning ifb. med projektering
2013.03.26	0.4	kje	Attraction url's renamed to provider, Notification resource UML-sequence diagrams
2013.09.19	0.5	kje	Change of price structure
2013.10.22	0.6	jdg	Booking and accomodation product import, barcode import
2013.12.12	0.7	jdg	Member status and updated information on date format
2014.02.05	0.8	sny	Added missing customer info for realtime sale
2014.03.14	0.9	sny,dtr, jri	New layout of the document to improve readability, improved parameter documentation and examples
2014.04.03	1.0	sny,dtr, jri	CSV format option for member import, fixed spelling, minor textual corrections
2014.04.03	1.01	jdg, sny	Spelling errors corrected, clarified the order import process.
2014.06.30	1.02	sny	Removed 'enabled' attribute from product import xml
2014.07.01	1.03	dtr	Added productLabel to xml that is used for realtime sales
2014.07.01	1.10	dtr	Result of GET action for product and member resources changed. Added proper xml example.
2014.07.03	1.11	dtr	Minor text corrections regarding barcodes. + Added barcodeType to order export xml.
2014.07.07	1.20	dtr	Updated member & product resources GET response.
2014.07.09	1.21	dtr	Added unitPrice to xml that is used for realtime sales

2014.09.03	1.22	sny	Added support for inventory for standard products, updated xml example
2014.09.10	1.3	dtr	Added currency and language resources
2014.10.22	1.31	dtr	Added new barcode type and specified which barcodes are supported by the pdf receipt.

Table of Contents

- Document History
- Table of Contents
- Document Purpose
- API
  - RESTful service
  - Security, authorization, and identity
  - Auth-token
  - Data Format
  - Barcode types
- Provider
  - Resource Overview
    - Product Import
    - Member Import
    - Order (Push)
    - Order (Pull)
    - Language, Currency
  - Product Import
    - Architecture Overview
    - Product import - step 1/3
    - Product import - step 2/3
    - Product import - step 3/3
  - Member Import
    - Architecture overview
    - Member import - step 1/3
    - Member import - step 2/3
      - XML example
      - CSV example
    - Member import - step 3/3
    - New ticket code type
    - Warnings
  - Order
    - Push Order (realtime)
      - Architecture overview
      - Create order - step ½
      - Create order - step 2/2
      - Delete order
    - Pull Order (anytime)
      - Architecture overview
      - Export order - step 1/2
      - Export order - step 1/2
  - Language, Currency

Language
Currency

Document Purpose

This document describes the API between providers (attractions) and Alletiders API.

Please note: This is a living document, if the API changes it will be updated accordingly and new versions issued.

API

RESTful service

The API uses RESTful web architecture.
Alletiders API implements CRUD functionality on a series of resources allowing product imports, member imports, and order creations.

Security, authorization, and identity

It is required to use https and basic authentication.

Each provider will be using its own login and hence identity. This will allow Alletiders API to separate the data from the individual providers.

Auth-token

There is currently no need for an authorization token as we are dealing with pure B2B communication between two trusted servers.

Data Format

XML is currently the standard supported. For member import, an alternative CSV option is available.

Barcode types

There are no constraints on the actual barcode types, but it would be good idea to reuse the same barcode type value to avoid duplicates.

Pretty much anything can be sent to the api via member import or via tickets and be treated as barcode type. The same type will be included in order lines when orders are exported.

The following is list is an example list of values that should be used to make integration easier for both parties.

EAN2	
EAN5	
EAN8	Supported by the Alletiders Api PDF receipt engine.
EAN13	Supported by the Alletiders Api PDF receipt engine.
UPC-A	Supported by the Alletiders Api PDF receipt engine.
UPC-E	Supported by the Alletiders Api PDF receipt engine.
Code128	Supported by the Alletiders Api PDF receipt engine.
Code39	Supported by the Alletiders Api PDF receipt engine.
Code93	
i25	Code 25 - Interleaved 2 of 5
QRCode	
PostNet	Supported by the Alletiders Api PDF receipt engine.

Provider

This section describes the available resources and how to use them. Examples included.

Resource Overview

Product Import

Resource	Description
POST /provider/product	Create a new transaction to be used to transfer the product data. Returns url to the new product including transaction id.
PUT /provider/product/{transaction-id}	Create or update the current product with product data.
GET /provider/product/{transaction-id}	Returns the status of the transaction

Member Import

Resource	Description
POST /provider/member/	Create a new transaction for member import. Returns url including transaction id.
PUT /provider/member/{transaction-id}	Import or update the current product with product data.
GET /provider/member/{transaction-id}	Returns the status of the transaction

Order (Push)

These resources are not located on Alletiders API, they are implemented on product provider's system. Thus URL mentioned here has a prefix defined by the individual providers in the admin.

Resource	Description
POST /order/	Create a new order. Returns url to the new order including order id.
GET /order/{order-id}	Executes the order
DELETE /order/{order-id}	Delete an order

Order (Pull)

Resource	Description
POST /provider/order-export/	Create a new transaction for order export. Returns url including transaction id.
GET /provider/order-export/{transaction-id}	Returns the status of the transaction

Language, Currency

Resource	Description
GET /provider/language	Retrieve a list of supported languages
GET /provider/currency	Retrieve a list of supported currencies.

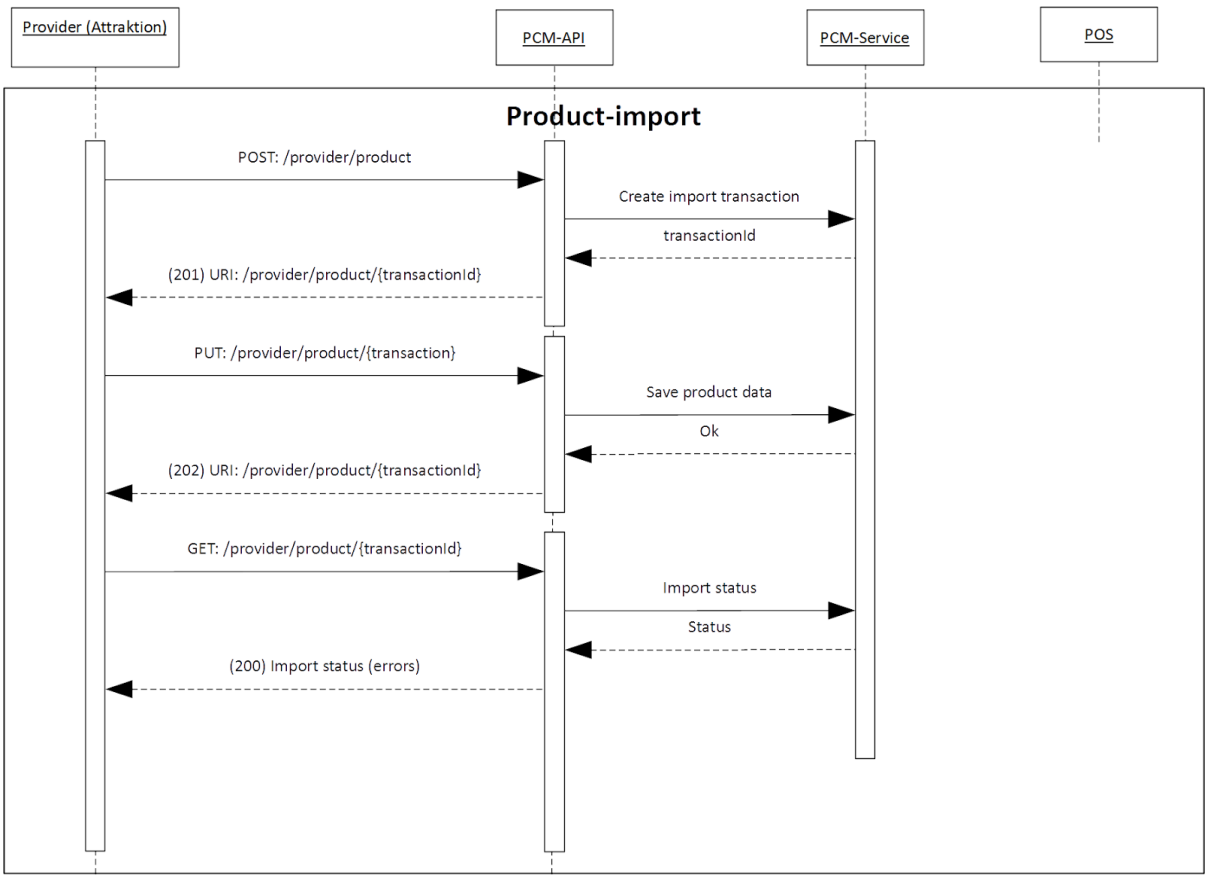
Product Import

Alletiders API provides a REST API allowing products to be imported.

The import process contains 3 required steps:

1. Create transaction
2. Send product data
3. Do the import

Architecture Overview



Product import - step 1/3

`POST /provider/product/`

Create a new product import transaction. The transaction will be used to transfer product data.

Response code: **201 Created**
Location URI: `/provider/product/{transaction}`

The location URI is returned from the POST-request and that exact URI is used in the next step

Product import - step 2/3

`PUT /provider/product/{transaction}`
Create and/or update products. Create or update will be determined by the product id, if it already exists an update will be done, otherwise a new product will be created.
All product attributes are optional, except the "id" and "type". In order to disable a product the sales period must be set to an expired date.

NOTE: The request will be completed as soon as the XML document has been transferred. The processing of data is done asynchronously afterwards.

Parameters that you can specify: (* = required parameters)

id*	product id [type: int |maxlength: 10]
type*	product type [type: varchar | maxlength: 255] Possible values: ● standard ● booking ● accommodation ● subscription
title**	product title [type: varchar | maxlength: 255] Defaults to "?????" if omitted during creation.
intro**	product short description [type: text] Defaults to "?????" if omitted during creation.
description**	product description [type: text|html] Defaults to "?????" if omitted during creation.
sale	sale can include the following properties: ● from [type: datetime | format: Y-m-d\TH:i:sP] ● to [type: datetime | format: Y-m-d\TH:i:sP] ● price ○ currency [ISO 4217 format e.g. EUR, DKK] ○ value [type: int] (in cents, to avoid rounding errors) ○ comment [string] Standard products supports inventory update: ● inventory [int] (positive value = add, negative = substract)

	If inventory exists on the product, the new value will be added or substracted to the existing inventory. additional options for accommodation only: ● interval_length* [number of days/weeks you have to book according to the interval_unit] ● interval_unit* [possible values: week | day | hour] ● week_days* [comma separated list of days as 1-7]
owner	product owner [type: varchar | maxlength: 80]
calendar	calendar properties for booking products only: ● interval_length* [length of interval as integer value] ● interval_unit* [possible values: week | day | hour] ● sales_deadline* ● sales_deadline_unit* ● active* [true | false] ● from* [Y-m-d] ● to* [Y-m-d]

** needs to be specified for each language [ISO 639-1 standard e.g. da, en]

Standard product creation (XML example)

```
<?xml version="1.0" encoding="UTF-8"?>
<products>
   <product id="423">
      <title>
         <values>
            <value lang="da">Entre</value>
            <value lang="en">Single day</value>
         </values>
      </title>
      <intro>
         <values>
            <value lang="da">Entre ...</value>
            <value lang="en">Single day ...</value>
         </values>
      </intro>
      <description>
         <values>
            <value lang="da">Entre billet gyldig for en enkelt dag</value>
            <value lang="en">Ticket valid for a single day</value>
         </values>
      </description>
      <type>standard</type>
```

```
      <sale>
         <from>2013-01-29T09:00:00+01:00</from>
         <to>2013-09-01T00:00:00+02:00</to>
         <prices>
            <price currency='DKK'>
               <value>4200</value>
               <comment>Udsalgspris</comment>
            </price>
            <price currency='EUR'>
               <value>600</value>
            </price>
         </prices>
        <inventory>10</inventory>
      </sale>
      <owner id="234">
         <name>Kim Jersin</name>
      </owner>
   </product>
</products>
```

Subscription product creation (XML request example)

```
<?xml version="1.0" encoding="UTF-8"?>
<products>
   <product id="4223">
      <title>
         <values>
            <value lang="da">Abonnements produkt</value>
            <value lang="en">Subscription product</value>
         </values>
      </title>
      <intro>
         <values>
            <value lang="da">Intro tekst ...</value>
            <value lang="en">Short description ...</value>
         </values>
      </intro>
      <description>
         <values>
            <value lang="da">Dette er et abonnement-produkt</value>
            <value lang="en">This is a subscription product</value>
         </values>
      </description>
      <type>subscription</type>
      <sale>
         <from>2014-01-01T09:00:00+01:00</from>
         <to>2014-03-01T00:00:00+02:00</to>
         <prices>
            <price currency='DKK'>
               <value>4200</value>
```

```
            <comment>Udsalgspris</comment>
         </price>
         <price currency='EUR'>
            <value>600</value>
         </price>
      </prices>
   </sale>
   <owner id="5634">
      <name>sny</name>
   </owner>
</product>
</products>
```

Booking product creation (XML example)
The differences to the standard product are marked as **bold**:

```
<?xml version="1.0" encoding="UTF-8"?>
<products>
   <product id="428">
      <title>
         <values>
            <value lang="da">Entre</value>
            <value lang="en">Single day</value>
         </values>
      </title>
      <intro>
         <values>
            <value lang="da">Entre ...</value>
            <value lang="en">Single day ...</value>
         </values>
      </intro>
      <description>
         <values>
            <value lang="da">Entre billet gyldig for en enkelt dag</value>
            <value lang="en">Ticket valid for a single day</value>
         </values>
      </description>
      <type>booking</type>
      <calendar interval_unit="day" sales_deadline="0" sales_deadline_unit="day">
         <entry active="true">
            <quantity>40</quantity>
            <from>2013-08-29</from>
            <to>2013-12-01</to>
            <comment>Imported sale</comment>
         </entry>
      </calendar>
      <sale>
         <from>2013-01-29T09:00:00+01:00</from>
         <to>2013-11-17T00:00:00+02:00</to>
         <prices>
            <price currency='DKK'>
```

```
            <value>4300</value>
            <comment>Udsalgspris</comment>
         </price>
         <price currency='EUR'>
            <value>600</value>
         </price>
      </prices>
   </sale>
   <owner id="234">
      <name>Kim Jersin</name>
   </owner>
</product>
</products>
```

Accommodation product (XML example)
The differences to the standard product are marked as **bold**:

```
<?xml version="1.0" encoding="UTF-8"?>
<products>
   <product id="428">
      <title>
         <values>
            <value lang="da">Entre</value>
            <value lang="en">Single day</value>
         </values>
      </title>
      <intro>
         <values>
            <value lang="da">Entre ...</value>
            <value lang="en">Single day ...</value>
         </values>
      </intro>
      <description>
         <values>
            <value lang="da">Entre billet gyldig for en enkelt dag</value>
            <value lang="en">Ticket valid for a single day</value>
         </values>
      </description>
      <type>accommodation</type>
      <calendar sales_deadline="0" sales_deadline_unit="day">
         <entry active="true">
            <quantity>40</quantity>
            <from>2013-08-29</from>
            <to>2013-12-01</to>
            <comment>Imported sale</comment>
         </entry>
      </calendar>
      <sale interval_length="2" interval_unit="week" week_days="6,7">
         <from>2013-01-29T09:00:00+01:00</from>
         <to>2013-11-17T00:00:00+02:00</to>
         <prices>
            <price currency='DKK'>
```

```
            <value>4300</value>
            <comment>Udsalgspris</comment>
         </price>
         <price currency='EUR'>
            <value>600</value>
         </price>
      </prices>
   </sale>
   <owner id="234">
      <name>Kim Jersin</name>
   </owner>
</product>
</products>
```

Calendar lines
For booking and accomodation products it is possible to add calendar lines (<calendar>).
A calendar line is uniquely identified by the combination of:

product-id,
start-date,
end-date,
quantity

Active/inactive lines
When importing a product all the lines are disabled before the actual import and only enabled if there are any matching lines. If there are no matches found on a imported line, it will be created in the system.

As an example, let's look at an existing product with following calendar data:
product-id: 1
start date: 1.September 2013
end date: 1.November 2013
quantity: 100

If a line with the same data is imported, the current line is updated to active / inactive.
If a line with a different quantity, such as 50, is imported, the current line is deactivated and a new quantity line will be added with 50.
A line can not be deleted.

Response code: **202 Accepted**

Product import - step 3/3

`GET /provider/product/{transaction}`

The GET method will show the current status of the import by returning one of following response codes:

Pending or processing: **202 Accepted**
Done with success: **200 OK**
Nothing was uploaded: **204 No Content**
Done with errors: **206 Partial Content**

Following xml example will be returned only for status code 206. For pending or successful imports response body will be empty.

```
<?xml version="1.0" encoding="UTF-8"?>
<errors xmlns="http://netimage.dk/pcm/provider/order-export">
 <error>
   <sequence>1</sequence>
   <id>18</id>
   <code>1048</code>
   <message>Example text</message>
 </error>
</errors>
```

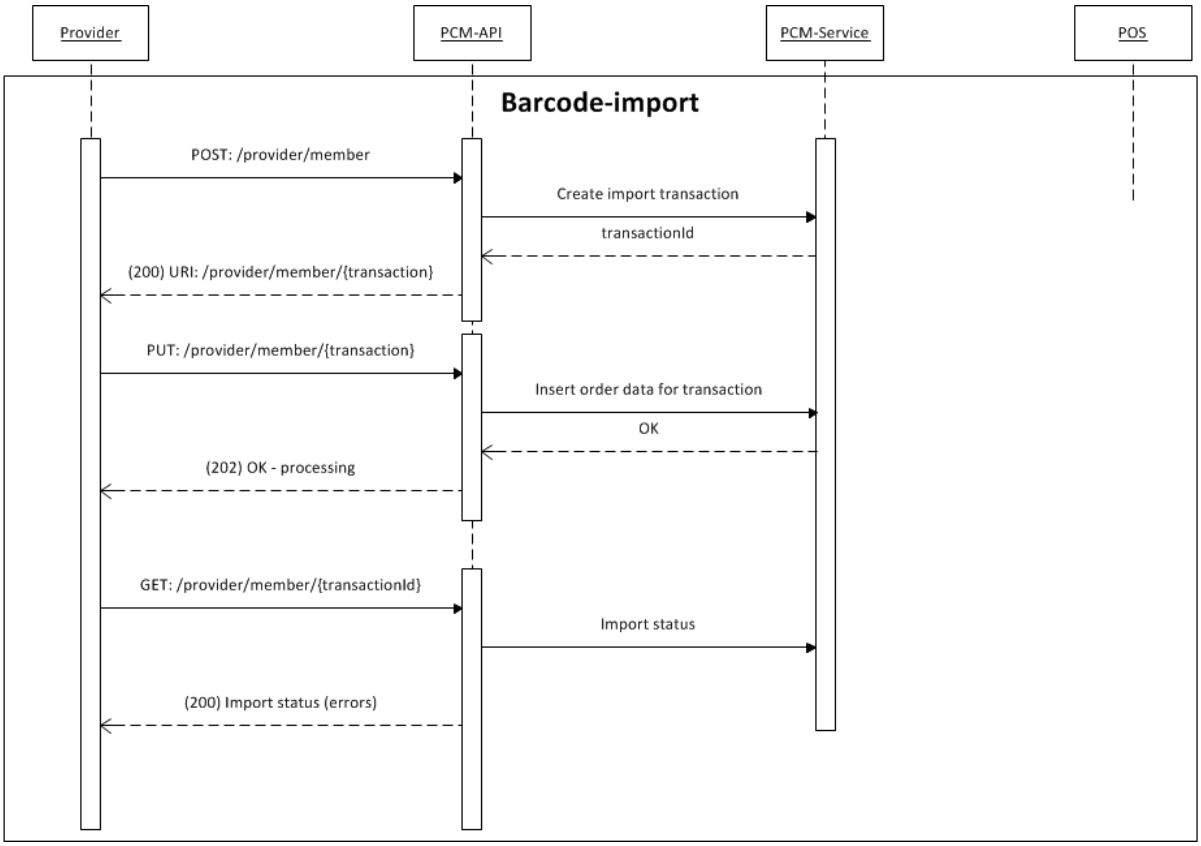
Member Import

Members include person, product, and ticket information. Since members are registered to a product, this product needs to be imported before the member is imported (if it does not exist in Alletiders API already). Furthermore in order to be able to scan tickets the product type needs to be subscription.
The member import currently supports two content types: XML and CSV.

The member import process contains 3 steps:

1. Create a transaction for the import
2. Send member data
3. Do the import

Architecture overview



Member import - step 1/3

`POST /provider/member`

Create a new member import transaction. The transaction is afterwards used to transfer members to Alletiders API.

Response code: **201 Created**
Location URI: `/provider/member/{transaction}`

The location URI is returned from the POST-request and that exact URI is used in the next step

Member import - step 2/3

`PUT /provider/member/{transaction}`

Imports the members to Alletiders API. If a member already exists data will be updated. Expected file encoding is UTF 8 and content type must be set in the header.

Parameters that you can specify: (* = required fields**)**

address	address includes following properties: ● name [type: varchar | maxlength: 255] ● address [type: varchar | maxlength: 255] ● zipCode [type: varchar | maxlength: 5] ● city [type: varchar | maxlength: 64] ● country [type: varchar | maxlength: 25] ● phone [type: varchar | maxlength: 16] ● email [type: varchar | maxlength: 255]
ticketIdentity*	ticket identity consists of the following properties: ticket code: ● code* [type: varchar |maxlength: 255] type of the ticket code: ● type [type: varchar | maxlength:50] See suggested values the validation period: ● valid ○ from* [type: datetime | format: YYYY-MM-DDThh:mm:ssTZD ○ (W3C)]

	○ to* [type: datetime | format: YYYY-MM-DDThh:mm:ssTZD ○ (W3C[1])]
product*	product consists of the product id that the member has subscribed to: ● label* [type: int | maxlength: 10] NOTE: the product needs to be of type subscription.

XML example

In order to import as xml the content type must be “application/xml”.

```
<?xml version="1.0" encoding="UTF-8"?>
<orders>
   <order>
      <members>
         <member>
            <address>
               <name>David Davidoff Davidsen</name>
               <address>Gadenavngade 10</address>
               <zipCode>8000</zipCode>
               <city>Aarhus C</city>
               <country>Danmark</country>
               <phone>86123456</phone>
               <email>david@example.com</email>
            </address>
            <ticketIdentity>
               <code>1234561234568</code>
               <type>barcode</type>
               <valid>
                      <from>2013-01-30T09:00:00+01:00</from>
                      <to>2013-05-25T00:00:00+00:00</to>
               </valid>

            </ticketIdentity>
            <product>
               <label>423</label>
            </product>
         </member>
         <member>
            <address>
               <name>William W. Willemoes</name>
               <address>Gadenavngade 103B</address>
               <zipCode>8240</zipCode>
               <city>Risskov</city>
               <country>Danmark</country>
               <phone>86654321</phone>
               <email>will@example.com</email>
            </address>
```

[1] http://www.w3.org/TR/NOTE-datetime

```
            <ticketIdentity>
               <code>1231231231233</code>
               <type>EAN13</type>
               <valid>
                      <from>2013-01-30T09:00:00+01:00</from>
                      <to>2013-05-25T00:00:00+00:00</to>
               </valid>
            </ticketIdentity>
            <product>
               <label>423</label>
            </product>
         </member>
      </members>
   </order>
</orders>
```

CSV example

In order to import as csv the content type must be “text/csv”.

delimiter	;
enclosure	"
escape	\\

```
label;type;code;validFrom;validTo;name;address;zip_code;city;country;phone;ema
il

1443;EAN13;1234561234568;2014-01-25T09:00:00+01:00;2014-05-25T00:00:00+00:00;"
David Davidoff Davidsen";"Gadenavngade 10";8000;"Aarhus
C";"Danmark";"86123456";"david@example.com"

5237;EAN13;1231231231233;2014-05-25T09:00:00+01:00;2014-05-25T00:00:00+00:00;"
William W. Willemoes";"Gadenavngade
103B";8240;"Risskov";"Danmark";"86654321";"will@example.com"
```

Please note: the example is shown with line breaks for a better layout only

Response code: **202 Accepted**
Location URI: `/provider/member/{transaction}`

Member import - step 3/3

`GET /provider/member/{transaction}`
The GET method will show the current status of the import by returning one of following response codes:

Pending or processing: **202 Accepted**
Done with success: **200 OK**
Nothing was uploaded: **204 No Content**
Done with errors: **206 Partial Content**

Following xml example will be returned only for 206 response code. For pending or successful imports response body will be empty.

```
<?xml version="1.0" encoding="UTF-8"?>
<errors xmlns="http://netimage.dk/pcm/provider/order-export">
 <error>
   <sequence>1</sequence>
   <id>18</id>
   <code>1048</code>
   <message>Example text</message>
 </error>
</errors>
```

It is not possible to delete a member /barcode. If a ticket is lost or needs to be deleted, you will have to change the valid to date to a date in the past. Thus, it is not possible to reuse old codes.

New ticket code type

It is not possible to change the type (ticketIdentity-type) on already imported member. This will result in a warning.

Warnings

Code	Description
995	This warning will be shown when updating an already imported member with a different product label. The rest will be updated
1062	The code exist for another provider. If this happens the import will be interrupted and nothing will be imported. Please contact Combine if this error occurs.
1452	The product-id (product label) does not exist. Please check if <product><label>xxx</label></product> is specified correctly and that the product exists in Alletiders API.

Order

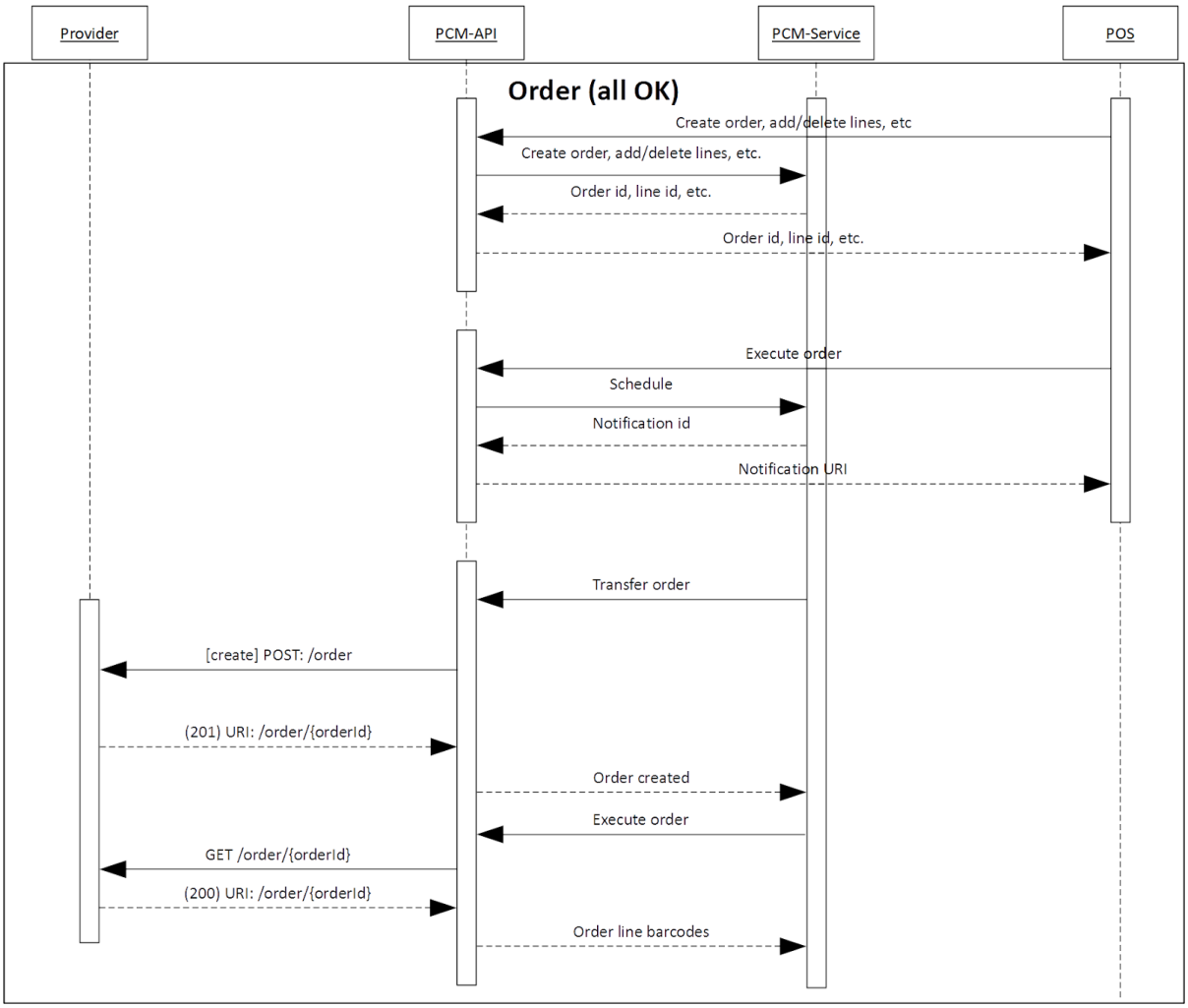
This API supports two ways of processing orders: push and pull. By push method a complete order is finalized in realtime while with pull method the providers would need to finalize the orders by themselves.

Push Order (realtime)

With push orders, the orders are sent in real time to the provider. The provider then needs to send the barcodes back to Alletiders API, which will then create the ticket and complete the order. This requires the provider to implement CRUD functionality for an order and provide RESTful service for the Alletiders API to fetch the barcodes.

Depending on the sales channel, alletiders API can be responsible for sending the order confirmation email with the tickets. In order to be able to create them successfully, barcode must be of type EAN13 or Code39.
On the other hand, if sales channel chose to send order confirmations - supported barcode types will vary. Please use this table to decide which barcode name to use.

Architecture overview



The following sections all describes examples of requests **to** a given provider. The endpoint is defined for each provider in the administration panel.
The API will keep calling the endpoints until a valid response code is returned. If the provider is down the call will thus be scheduled to execution within near future.

Create order - step ½

The first step in push order consists of the Alletiders API making a POST request to the provider with the order data.

`POST /order/`

Creates new order with order lines.
Note: channelId may be null;
productLabel on the line tag is the product id specified when importing product, it will not be included if product was created in admin interface.

Example of post data:

```
<?xml version="1.0" encoding="UTF-8"?>
<order id="1">
  <created>2013-01-30T09:00:00+01:00</created>
  <modified>2013-01-30T09:00:00+01:00</modified>
  <address id="3">
    <name>gd</name>
    <address>gd</address>
    <zipCode>gds</zipCode>
    <city>gd</city>
    <country></country>
    <phone></phone>
    <email>dtr@netimage.dk</email>
  </address>
  <currency>DKK</currency>
  <channelId>5</channelId>
  <lines>
    <line id="1">
      <productDataId>6</productDataId>
      <productDataVer>1</productDataVer>
      <productLabel>5147</productLabel>
      <price>19900</price>
      <unitPrice>9950</unitPrice>
      <quantity>2</quantity>
      <providerId>1</providerId>
    </line>
  </lines>
</order>
```

Response code: **201 Accepted**

Location URI: `/order/{orderId}`

Create order - step 2/2

The second step consists of the Alletiders API making a GET request to the provider with the order id. Alletiders API expects the provider to return barcodes for the given order id. The orderId used here is the one extracted from location header from previously made POST request.

`GET /order/{orderId}`

Get barcodes from provider and execute the order.

Response code: **200 OK**
Location URI: `/order/{orderId}`

Example of expected response:

```
<?xml version="1.0" encoding="UTF-8"?>
<order>
   <lines>
      <line id="0">
         <barcodes>
            <barcode type="Code39">Wikipedia</barcode>
         </barcodes>
      </line>
      <line id="1">
         <barcodes>
            <barcode type="EAN13">899067654321</barcode>
            <barcode type="Code39">102467654321</barcode>
            <barcode type="Code39">204867654321</barcode>
         </barcodes>
      </line>
   </lines>
</order>
```

Delete order

`DELETE /order/{orderId}`

Deletes an existing order. The orderId used here is the one extracted from location header from previously made POST request.

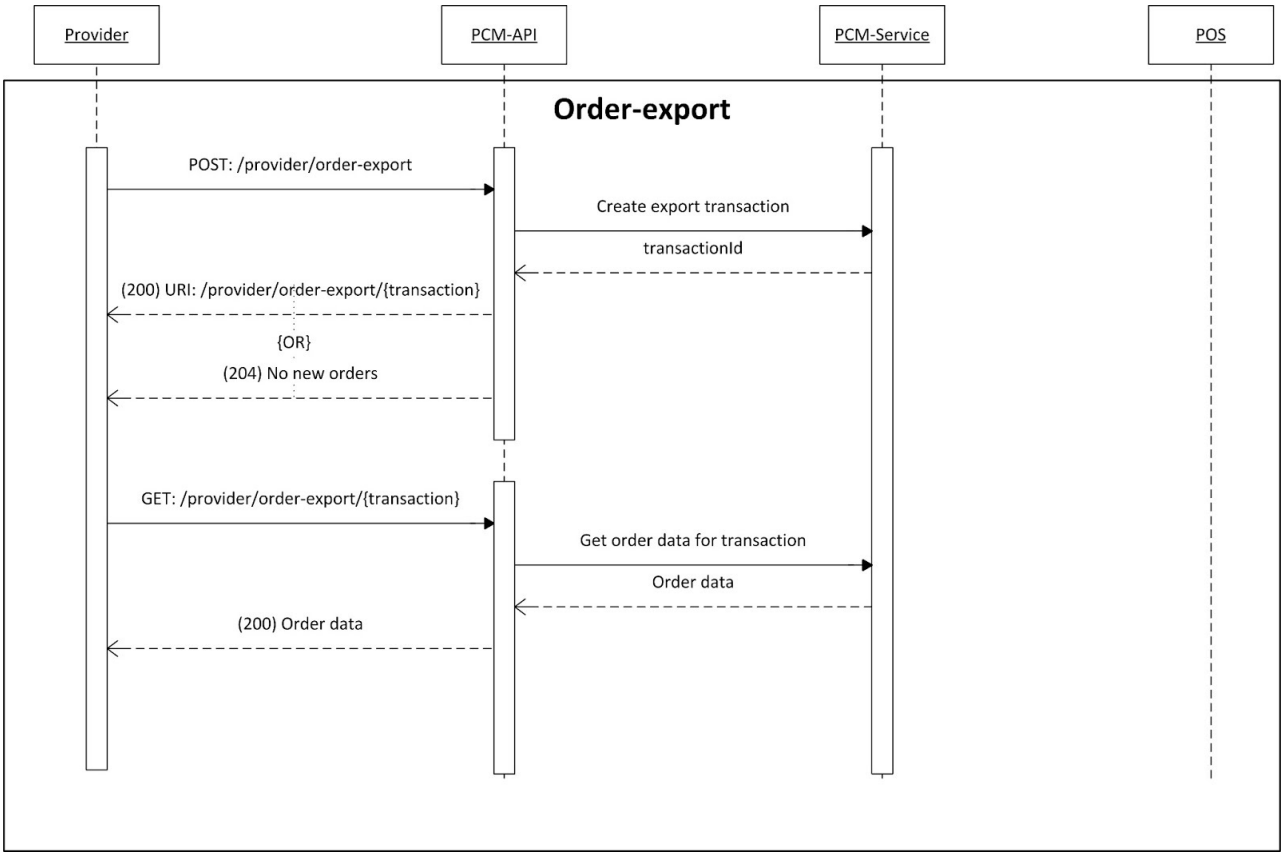
NOTE: If the GET was invoked for this order (ticket identities were created), it is then the responsibility of the provider to do any necessary cleaning in their systems.

Response code: **204 No Content**

Pull Order (anytime)

This method enables the providers to pull a list of orders ready for processing using the Alletiders API. It is then up to the provider to complete the order process by providing the customer the confirmation mail and the ticket.

Architecture overview



Export order - step 1/2

`POST /provider/order-export/`
Create a new transaction for order export.

Response code: **200 OK**
Location URI: `/provider/order-export/{transaction}`

Response code: **204 No Content**
No orders on current transaction or no new orders. Try again later

Export order - step 1/2

`GET /provider/order-export/{transaction}`
Get all completed orders for given transaction id

New transaction:

- All new orders are marked as exported (with the transaction) and returned

Used transaction

- Orders already marked (exported or tried exported) would be returned

Response code: **200 OK**

```
<?xml version="1.0" encoding="UTF-8"?>
<orders xmlns="http://netimage.dk/pcm/provider/order-export">
  <order id="7">
    <insertTime>2013-02-07T14:46:23+01:00</insertTime>
    <modified>2013-02-08T12:37:04+01:00</modified>
    <address id="3">
      <name>John Doe</name>
      <address>Evergreen Terrace 742</address>
      <zipCode>1000</zipCode>
      <city>København K</city>
      <country>Denmark</country>
      <phone></phone>
      <email>jd@example.com</email>
    </address>
    <channelId>3</channelId>
    <currency>EUR</currency>
    <orderLines>
      <orderLine id="23">
        <quantity>2</quantity>
        <prices>
```

```
          <price id="54">
            <value>26000</value>
          </price>
        </prices>
        <product id="6" ver="1">
          <title>Dobbeltværelse (pris for 2 pers.)</title>
        </product>
        <ticketIdentities>
          <ticketIdentity id="23">
            <type>barcode</type>
            <code>8705647250635</code>
            <barcodeType>EAN13</barcodeType>
            <scanLog/>
          </ticketIdentity>
        </ticketIdentities>
      </orderLine>
      <orderLine id="24">
        <quantity>2</quantity>
        <prices>
          <price id="51">
            <value>26000</value>
          </price>
        </prices>
        <product id="6" ver="1">
          <title>Dobbeltværelse (pris for 2 pers.)</title>
        </product>
        <ticketIdentities>
          <ticketIdentity id="24">
            <type>barcode</type>
            <code>5432758218607</code>
            <barcodeType>EAN13</barcodeType>
            <scanLog/>
          </ticketIdentity>
        </ticketIdentities>
      </orderLine>
    </orderLines>
  </order>
</orders>
```

channelId
Only if an order is made using a sales channel the channel id will be specified

Language, Currency

Language

`GET /provider/language`

Retrieve a list of supported languages.
Response code: **200 OK**

Example responses

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<languages xmlns="http://netimage.dk/pcm/provider/language">
  <language id="1" code="da" locale="da_DK">danish</language>
  <language id="2" code="en" locale="en_GB">english</language>
  <language id="3" code="de" locale="de_DE">german</language>
  <language id="4" code="no" locale="nb_NO">norwegian</language>
  <language id="5" code="sv" locale="sw_SE">swedish</language>
  <language id="6" code="nl" locale="nl_NL">dutch</language>
</languages>
```

JSON

```
[
   {
      "id":1,
      "label":"danish",
      "locale":"da_DK",
      "code":"da"
   },
   {
      "id":2,
      "label":"english",
      "locale":"en_GB",
      "code":"en"
   },
   {
      "id":3,
      "label":"german",
      "locale":"de_DE",
      "code":"de"
   },
   {
      "id":4,
      "label":"norwegian",
      "locale":"nb_NO",
      "code":"no"
   },
   {
      "id":5,
      "label":"swedish",
      "locale":"sw_SE",
      "code":"sv"
   },
   {
      "id":6,
      "label":"dutch",
      "locale":"nl_NL",
      "code":"nl"
   }
]
```

Currency

`GET /provider/currency`

Retrieve a list of supported currencies.
Response code: **200 OK**
Example responses

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<currencies xmlns="http://netimage.dk/pcm/provider/currency">
  <currency id="1">DKK</currency>
  <currency id="2">EUR</currency>
  <currency id="3">USD</currency>
  <currency id="4">SEK</currency>
  <currency id="5">NOK</currency>
</currencies>
```

JSON

```
[
   {
      "id":1,
      "currency":"DKK"
   },
   {
      "id":2,
      "currency":"EUR"
   },
   {
      "id":3,
      "currency":"USD"
   },
   {
      "id":4,
      "currency":"SEK"
   },
   {
      "id":5,
      "currency":"NOK"
   }
]
```