Luisterhuis Reseller API

Version 2.0

Published by:

Luisterhuis, a subsidiary of Luisterrijk

More information:

Icontact B.V.
A. Hofmanweg 5a
2031 BH Haarlem
Netherlands

+31-23-7505125
support@icontact.nl

Table of contents

1 Introduction

1.1 About this document

Luisterhuis is an independent digital distributor that offers 1000+ audio books from different publishers in MP3 format to web shops.

Luisterhuis operates as a RESTfull webservice. It supplies a web shop with catalogue data, book-cover images and audio samples. It accepts orders from a web shop and delivers the digital download links directly to customers of the web shop.

Luisterhuis uses the award winning BooXtream social DRM protection system, which combines total MP3 compatibility with the assurance that illegal use of the MP3 files will be minimized.

In addition to the webservice, Luisterhuis also has a webbased human interface that offers web shops and publishers a real time overview of all audio books sold in a certain period.

This documentation is prepared for software developers and systems integrators who wants to add the Luisterhuis digital distribution service to a web shop.

1.2 Revision history

v2.0 16-08-2010
Rewritten documentation, now including description of 2 interface scenarios
New API function: update by date

v1.4 21-07-2010
New API function: orders accepted by product ID or EAN13

v1.3 04-06-2010
New API function: send downloadlink by e-mail

v1.2 31-05-2010
Bug fixes

v1.1 27-05-2010
Stub (testing) environment added

v1.0 06-05-2010
First release

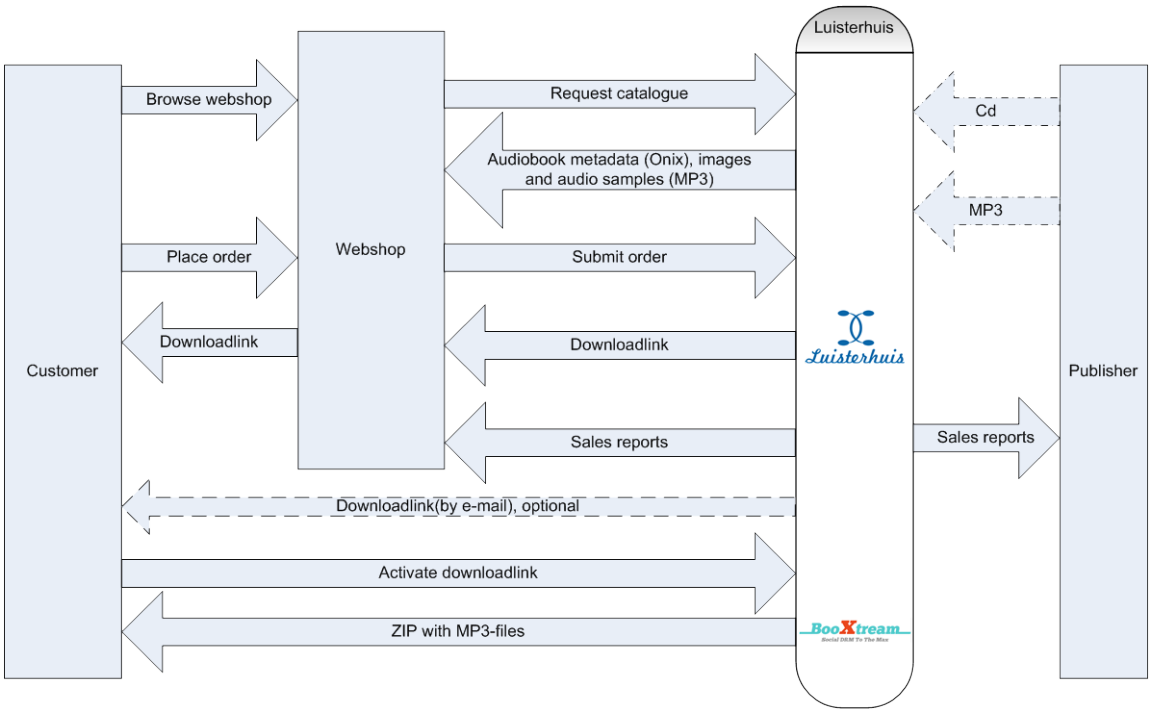
2 Architecture

2.1 Luisterhuis as a digital distribution service

Luisterhuis operates as a distribution service between publishers of audio books and web shops.

It enables a web shop to sell audio book downloads, without actually storing and maintaining the audio book files in the web shop. Instead, the web shop only has to implement an interface to Luisterhuis to add audio book product data to its catalogue and to place orders. Every else is handled by the Luisterhuis system, including the delivery of the protected MP3 files to the end user.

A simplified overview of the architecture:



2.2 Delivering audio books to the end user

Luisterhuis stores all audio files of the audio books on its own secure servers. Luisterhuis does not distribute the audio files to the web shops. Instead, Luisterhuis only distributes information about the audio books to the web shops, such as price data, author, narrator, images and sound samples. After a successful transaction, Luisterhuis delivers the audio files directly to the end user.

The size of a digital audio book is between 20 MB and over 1 GB each, depending of the length of the recording. Unlike ebooks, where the delivery includes only one ePub file, a digital audio book consists of several MP3 audio files, with up to several hundreds of MP3 files for one audio book.

However, manually downloading a lot of MP3 files for one audio book, is very unfriendly for the end user.

Because of this, all MP3 files are packed together into up to eight ZIP files. A Luisterhuis ZIP file has a maximum size of 250 MB, which is the typical size of an digital audio book. Therefore, most audio books are delivered using one ZIP file containing all MP3's. Only the larger audio books use 2 or more ZIP files. This enables the end user to selectively download large audio book in parts, which could come in handy when download speeds are low.

The ZIP files are build using the following naming and URL convention:

```
Name  title%20(setnumber-maxnumberofsets).zip

URL   http://download.luisterhuis.nl/temporarylink/
            setnumber/title%20(setnumber-maxnumberofsets).zip
```

For instance, a single ZIP file:

```
Name  dijkshoorn%20leest%20(01-01).zip

URL   http://download.luisterhuis.nl/4950838014c88b09e07bad8.57098465
      /01/dijkshoorn%20leest%20(01-01).zip
```

or a set of multiple ZIP files:

ZIP file 1

```
Name  Belofte%20onder%20het%20Zuiderkruis%20(01-02).zip

URL   http://download.luisterhuis.nl/11759749014c8e31bc2ccd28.87142863
      /01/Belofte%20onder%20het%20Zuiderkruis%20(01-02).zip
```

ZIP file 2

```
Name  Belofte%20onder%20het%20Zuiderkruis%20(02-02).zip

URL   http://download.luisterhuis.nl/11759749014c8e31bc2ccd28.87142863
      /02/Belofte%20onder%20het%20Zuiderkruis%20(02-02).zip
```

Read chapter 6 for more details about the delivery of the download links to the webshop and the customer.

2.3 BooXtream social DRM

Luisterhuis uses BooXtream social DRM. This means that every MP3 file is created on the fly, with dynamic created information about Luisterhuis, the web shop and the customer. The MP3 file and the enclosed (cover) image within the MP3 file header contains personalized information and other digital fingerprints.

A MP3-file with BooXtream social DRM can be traced back to the individual customer and to the web shop where the customer had bought the audio book, but the files are 100% compatible with every (hardware and software) MP3 player.

3 Connecting your web shop to Luisterhuis

3.1 Different interface scenarios

The API of Luisterhuis offers several functions to connect to your web shop. The interface technology is based on REST (Representational State Transfer, see http://en.wikipedia.org/wiki/Representational_State_Transfer). The different REST functions of the Luisterhuis API are described in chapter 4, 5 and 6. Which functions to choose, depends of the way you want to include the Luisterhuis content into your own catalogue and the way your shop operates.

We distinguish two different interface scenarios:

- real time
- batch

3.2 Real time operation mode

In real time operation mode, the web shop queries Luisterhuis whenever a visitor of the web shop is looking for audio books. This mode is especially useful for web shops that have a limited number of vistors or where the shop catalogue only contains audio books from Luisterhuis and no other products.

The API offers several functions to query the Luisterhuis audio book products, such as a list of all products, a list of all authors, a list of all publishers and faceted search. For details, see chapter 4.

When an end user places an order in the web shop, the web shop places the same order at Luisterhuis, which in turn delivers the download link to the shop or directly to the end user. For details, see chapter 5.

Because in real time operation mode, the performance of the web shop is dependent of the performance of the Luisterhuis server and the internet connection in between, it is advised to cache the most common queries and results.

3.3 Batch operation mode

In batch operation mode, the web shop queries the Luisterhuis distribution server once a day to process the product information. This is the most common operation mode, as it offers the possibility to include the Luisterhuis audio book catalogue into a larger catalogue of the web shop. This can be useful if the web shop also sells normal books, e-books and or audio cd's, because it allows the web shop to process queries on author, title and publisher and display the combined results including audio books from Luisterhuis.

The API offers several functions to query the Luisterhuis audio book products, such as a list of all products and a list of changed or new products since a specific date. For details, see chapter 4.

When an end user places an order in the web shop, the web shop places the same order at Luisterhuis, which in turn delivers the download link to the shop or directly to the end user. For details, see chapter 5.

In batch operation mode, the performance of the web shop is independent of the performance of the Luisterhuis server and the internet connection in between. The only exception to this rule is the order placement process, which always takes place in real time.

3.4 How to use the webservices?

No matter what mode is being used, there are always some basic steps to take when using the Luisterhuis webservices. These webservices will be described in detail in the next chapters, but first a simplified overview.

First of all, your web shop will need some additional functions to connect to the Luisterhuis distribution server. Because the REST technology is being used, all calls will use the HTTP protocol (and HTTPS); which means that your firewall does not need any changes.

Using REST means that your web shop sends out requests and queries with the HTTP protocol, and gets responses back in the same way. A response is either a confirmation or an error, and - if no error - the response also contains XML data you will have to process.

The commands, functions and queries will be described in detail in chapter 4, 5 and 6. In some cases, the XML contains product information about audio books. Whenever possible, Luisterhuis uses the industry standard XML representation for book information called Onix.

3.5 Onix

The Luisterhuis catalogue of audio book products contains all metadata of the audio books, like author, publisher, prices et cetera. The data is supplied in XML following the Onix 3.0 guidelines.

Onix for Books is a worldwide XML standard to describe book information, to connect the servers of publishers, distributors and retailers, without any conversion or manual data entry of the book data.

Information about the Onix standard is available online on www.editeur.org

It is important to know that to process the XML files of Luisterhuis, you will have to 'know' how Onix works. We can recommend to download the documentation on http://www.editeur.org/93/Release-3.0-Downloads/

3.6 The basic steps

It all starts with the retrieval of the product catalogue from Luisterhuis, to incorporate it in your own web shop catalogue. After the Luisterhuis audio book products are available in your own web shop, you can sell them to your customers. You will also have to check regularly if there are new or changed products. (New products will be added on a daily basis; and sometimes the price or the description of an existing product can change). *The functions for catalogue retrieval are describe in Chapter 4.*

When an audio book is being sold from your web shop, it is common practice to first check if the product is still available at Luisterhuis. So before you accept payment from your customer, check the availability. There is an API call for that. (In fact: there are two ways of checking: you can simply check, or you can combine the checking with the placement of an order). The reason is not that the product can be out of stock, but the product can be withdrawn from the catalogue due to commercial or legal reasons. *The functions for order placement are described in Chapter 5.*

After you have received a confirmation that the product is still available, you can proceed to the checkout procedure in your own shop and accept payment from your customer.

When your shop places an order at Luisterhuis, you will have to supply Luisterhuis with some details of the order, like the product(s) to order, the customer name and/or his e-mail address and your internal order number. After Luisterhuis accepts your order, it will supply you with a Luisterhuis order number.

Using this Luisterhuis order number, your shop can request the download link. This download link can be stored in a private area on your site, available only for your end user. And / or you can sent the link by e-mail to your end user. There is also a possibility that Luisterhuis sents the link directly to your end user. Either way, using this link, the end user can download the ZIP files containing the audio books. *The functions for handling the download links are described in Chapter 6.*

3.7 Accounting

Depending of the agreement you have with Luisterhuis, you will get a monthly or quarterly receipt with a detailed description of all audio books sold.

You also have access to a real time report of all transactions and sales reports through www.luisterhuis.nl/reseller or stub.www.luisterhuis.nl/reseller (testing purposes).

Every reseller is provided with a loginname and password for authentication.

4 Reseller API for catalogue retrieval

4.1 Summary of functions

Luisterhuis uses a RESTfull webservice to communicate with a reseller.

The interface has the following functions:

- **Product Information in ONIX 3 Format**
- **(Faceted) search**
- **List of products in ONIX 3 Format**
- **List of publishers**
- **Publisher by ID**
- **List of authors**
- **Author by ID**
- **List of tags**
- **Tag by ID**
- **Top 10**

It is up to the web shop to use all functions in real time or to load and buffer all catalogue data in a repeating cycle (for instance every night), see also chapter 3.2 and 3.3 (different interface scenarios).

The interface and sample data can also be found on https://rest.luisterhuis.nl or https://stub.rest.luisterhuis.nl (testing purposes).

The authentication is described in chapter 7. The testing environment is described in chapter 8.

4.2 Product Information in ONIX 3 Format

Type	`GET`
URL	`/product/id/[PRODUCTID]`
or	`/product/isbn/[ISBN]`

4.3 (Faceted) search

This resource can also be run without a query

Type	`GET`
URL	`/products/`
Query	`Search=[TERM], PublisherID=[PUBLISHERID], AuthorID=[AUTHORID], Tags=[TAGS], NUR=[NURs], PriceFrom=[PRICE], PriceTo[PRICE]`
Tags	`Comma seperated list of TagID's (AND relationship): 88,3`
NURs	`Comma seperated list of NUR values and/or ranges (OR relationship): 210,212` or `200-299,323`
Price	`Price (including tax) in cents (ex. 23,95 = 2395)`
Count	`/products/count/(?query)`
Limits	`/products/[OFFSET]/[LIMIT]/ (max limit is 100)`

4.4 List of products in ONIX 3 Format

Type GET
URL /inventory/
Limits /inventory/[OFFSET]/[LIMIT]/ (max limit is 100)
Query ModifiedFrom=[YYYYMMDDHHIISS]
Modified Lists modified and new products from specified date and time
Y = year, M = month, D = day, H = hour, I = minute, S = second

4.5 List of publishers

Type GET
URL /publishers/

4.6 Publisher by ID

Type GET
URL /publisher/[PUBLISHERID]

4.7 List of authors

Type GET
URL /authors/

4.8 Author by ID

Type GET
URL /author/[AUTHORID]

4.9 List of tags

Type GET
URL /tags/

4.10 Tag by ID

Type GET
URL /tag/[TAGID]

4.11 Top 10

Type GET
URL /topten/?NUR=[NURs]
NURs Comma seperated list of NUR values and/or ranges (OR relationship)

5 Reseller API for order placement

Orders can be placed either through a submit/confirm scheme or by a direct order. With the 'submit/confirm' scheme the reseller can place an order just before the customer pays for it. If Luisterhuis responses with `200 OK`, this ensures all the products are available. After the customer has payed for the order, the reseller confirms the order at Luisterhuis. If the order is not confirmed it is discarded by Luisterhuis.

With 'direct order' the reseller places the order after the customer has payed for it. A confirmation is not necessary, but the reseller only knows if the order is correct, after the payment.

Products can be identified by either the Luisterhuis ProductID or the EAN13 (ISBN).

The submit/confirm scheme described in chapters 5.2 and 5.3 is recommended.

5.1 Variables

In the API description, the following variables are used:

ResellerOrderID	`varchar(40)`
CustomerEmailAddress	`varchar(255)`

CustomerName	`varchar(255)`
	`/* either CustomerEmailAddress`
	`or CustomerName is required */`

OrderID	`int(11)`
OrderStatus	`0=pending, 1=confirmed`
SuppliersPrice	`eurocents`

5.2 Submit Order

REQUEST

Action: /submitOrder

Method: POST

Fieldname: Order

Fieldvalue:

```
<Order>
   <ResellerOrderID></ResellerOrderID>
   <CustomerEmailAddress></CustomerEmailAddress>
   <CustomerName></CustomerName>
   <Products>
     <Product>
       <ProductID>1</ProductID>     /* either ProductID */
       <EAN13></EAN13>              /* or EAN13 is required */
     </Product>
     <Product>
       <ProductID>3</ProductID>     /* either ProductID */
       <EAN13></EAN13>              /* or EAN13 is required */
     </Product>
   </Products>
</Order>
```

RESPONSE

HTTP Status 200 OK

```
<Message>
   <Request>
      <Order>
          <!— Repetition of Request Order -->
      </Order>
   </Request>
   <Response>
      <Code>200</Code>
      <Msg>OK</Msg>
      <Order>
         <OrderID></OrderID>
         <OrderStatus>0</OrderStatus>
         <Products>
            <Product>
               <ProductID>1</ProductID>
               <SuppliersPrice>1879</SuppliersPrice>
            </Product>
            <Product>
               <ProductID>3</ProductID>
               <SuppliersPrice>3092</SuppliersPrice>
            </Product>
         </Products>
      </Order>
   </Response>
</Message>
```

HTTP Status 409 Conflict

```
<Message>
   <Request>
```

```
      <Order>
      <!– Repetition of Request Order -->
      </Order>
   </Request>
   <Response>
      <Code>1000</Code>
      <Msg>Products Not Available</Msg>
      <Products>
         <Product>
            <ProductID>1</ProductID>
            <Available>1</Available>
         </Product>
         <Product>
            <ProductID>3000</ProductID>
            <Available>0</Available>
         </Product>
      </Products>
   </Response>
</Message>
```

HTTP Status 422 Unprocessable Entity

```
<Message>
   <Request>
      <Order>
      <!– Repetition of Request Order -->
      </Order>
   </Request>
   <Response>
      <Code>1001</Code>
      <Msg>One Or More Mandatory Fields Are Empty</Msg>
   </Response>
</Message>
```

HTTP Status 400 Bad Request

```
<Message>
   <Request>
      <Order>
         <!– Repetition of Request Order -->
      </Order>
   </Request>
   <Response>
      <Code>400</Code>
      <Msg>Bad Request</Msg>
   </Response>
</Message>
```

5.3 Confirm Order

REQUEST

Action: /confirmOrder

Method: POST

Fieldname: Order

Fieldvalue:
```
<Order>
   <OrderID></OrderID>
</Order
```

RESPONSE

HTTP Status 200 OK
```
<Message>
   <Request>
      <Order>
        <OrderID></OrderID>
      </Order>
   </Request>
   <Response>
      <Code>200</Code>
      <Msg>OK</Msg>
      <Order>
         <OrderID></OrderID>
         <OrderStatus>1</OrderStatus>
      </Order>
   </Response>
</Message>
```

HTTP Status 422 Unprocessable Entity
```
<Message>
   <Request>
      <Order>
        <OrderID></OrderID>
      </Order>
   </Request>
   <Response>
      <Code>1000</Code>
      <Msg>Invalid OrderID</Msg>
   </Response>
</Message>
```

HTTP Status 422 Unprocessable Entity
```
<Message>
   <Request>
      <Order>
         <OrderID></OrderID>
      </Order>
   </Request>
   <Response>
      <Code>1001</Code>
      <Msg>Invalid OrderID/ResellerID Combination</Msg>
```

```
   </Response>
</Message>
```

HTTP Status 400 Bad Request

```
<Message>
   <Request>
      <Order>
         <!— Repetition of Request Order -->
      </Order>
   </Request>
   <Response>
      <Code>400</Code>
      <Msg>Bad Request</Msg>
   </Response>
</Message>
```

5.4 Order Direct

REQUEST

Action: /orderDirect

Method: POST

Fieldname: Order

Fieldvalue:

```
<Order>
   <ResellerOrderID></ResellerOrderID>
   <CustomerEmailAddress></CustomerEmailAddress>
   <CustomerName></CustomerName>
   <Products>
     <Product>
       <ProductID>1</ProductID>     /* either ProductID */
       <EAN13></EAN13>              /* or EAN13 is required */
     </Product>
     <Product>
       <ProductID>3</ProductID>     /* either ProductID */
       <EAN13></EAN13>              /* or EAN13 is required */
     </Product>
   </Products>
</Order>
```

RESPONSE

HTTP Status 200 OK

```
<Message>
   <Request>
      <Order>
          <!— Repetition of Request Order -->
      </Order>
   </Request>
   <Response>
      <Code>200</Code>
      <Msg>OK</Msg>
      <Order>
         <OrderID></OrderID>
         <OrderStatus>0</OrderStatus>
         <Products>
            <Product>
               <ProductID>1</ProductID>
               <SuppliersPrice>1879</SuppliersPrice>
            </Product>
            <Product>
               <ProductID>3</ProductID>
               <SuppliersPrice>3092</SuppliersPrice>
            </Product>
         </Products>
      </Order>
   </Response>
</Message>
```

HTTP Status 409 Conflict

```
<Message>
   <Request>
```

```
      <Order>
      <!– Repetition of Request Order -->
      </Order>
   </Request>
   <Response>
      <Code>1000</Code>
      <Msg>Products Not Available</Msg>
      <Products>
         <Product>
            <ProductID>1</ProductID>
            <Available>1</Available>
         </Product>
         <Product>
            <ProductID>3000</ProductID>
            <Available>0</Available>
         </Product>
      </Products>
   </Response>
</Message>
```

HTTP Status 422 Unprocessable Entity

```
<Message>
   <Request>
      <Order>
      <!– Repetition of Request Order -->
      </Order>
   </Request>
   <Response>
      <Code>1001</Code>
      <Msg>One Or More Mandatory Fields Are Empty</Msg>
   </Response>
</Message>
```

HTTP Status 400 Bad Request

```
<Message>
   <Request>
      <Order>
         <!– Repetition of Request Order -->
      </Order>
   </Request>
   <Response>
      <Code>400</Code>
      <Msg>Bad Request</Msg>
   </Response>
</Message>
```

6 Download links

After an order has been placed and confirmed, the download links for the zip-files are available. They can either be sent to the customer directly by email (6.2) or be retrieved through the API (6.1). If retrieved through the API, it is up to the webshop to offer the download links though a protected user area on the website and / or to mail the download links to the end user.

6.1 Download links to ZIP files

The download link to a single ZIP file will look as follows:

```
http://download.luisterhuis.nl/4950838014c88b09e07bad8.57098465/01/dijkshoo
rn%20leest%20(01-01).zip
```

A typical set of ZIP files (all ZIP-files, except for the last one, having a size of 250 MB) will look as follows:

The download links to a set of ZIP file looks as follows:

```
http://download.luisterhuis.nl/11759749014c8e31bc2ccd28.87142863/01/Belofte
%20onder%20het%20Zuiderkruis%20(01-02).zip
```

```
http://download.luisterhuis.nl/11759749014c8e31bc2ccd28.87142863/02/Belofte
%20onder%20het%20Zuiderkruis%20(02-02).zip
```

The links will have a limited lifetime. Everytime a download link is requested through the API, a new (set of) link(s) will be generated.

6.2 Retrieve download links through API

REQUEST

Action: /downloadinfo/[orderid]

Method: GET

RESPONSE

HTTP Status 200 OK

```
<Message>
   <Request>
      <OrderID>[orderid]</OrderID>
   </Request>
   <Response>
      <Code>200</Code>
      <Msg>OK</Msg>
      <Order>
         <OrderID>[orderid]</OrderID>
         <Products>
            <Product>
               <ProductID>69</ProductID>
               <SuppliersPrice>0</SuppliersPrice>
               <Sets>
                  <Set>
                     <Number>01</Number>
                     <Filesize format="megabytes">213</Filesize>
```

```
                  <DownloadLink>[link]</DownloadLink>
               </Set>
               <Set>
                  <Number>02</Number>
                  <Filesize format="megabytes">169</Filesize>
                  <DownloadLink>[link]</DownloadLink>
               </Set>
            </Sets>
         </Product>
         <Product>
            <ProductID>211</ProductID>
            <SuppliersPrice>2230</SuppliersPrice>
            <Sets>
               <Set>
                  <Number>01</Number>
                  <Filesize format="megabytes">157</Filesize>
                  <DownloadLink>[link]</DownloadLink>
               </Set>
            </Sets>
         </Product>
      </Products>
   </Order>
</Response>
</Message>
```

6.3 Email with download links

If the reseller wishes, an email with download links can be sent automatically to the customer. This can only be done if the customer email-address was provided in the original order.

To enable this feature, please contact Luisterhuis.

7 Authentication for the REST API

Authentication for the REST API is implemented through the Basic Authentication protocol. **Every reseller is provided with a Loginname and an AuthenticationKey**. The **HTTPS** protocol is used for communication with the REST interface.

Please note: when using the test environment (see the next chapter), there is a certificate warning. To skip this warning, PHP developers can add the following lines to their code:

```
curl_setopt($s, CURLOPT_SSL_VERIFYPEER, FALSE);
curl_setopt($s, CURLOPT_SSL_VERIFYHOST, FALSE);
```

8 Testing the API through the stub interface

For testing purposes the following url can be used: https://stub.rest.luisterhuis.nl

The stub API has the same functionality as the production API. There is one exception: only “free” products can be downloaded. Furthermore, the reseller will not be charged for the orders that are placed in the stub.

Please note: the products database is a recent copy of the live database but it will not be modified on a daily basis. Therefore, it will contain the almost same products as the live database, but differences will apply.

You also have access to a real time report of all transactions and sales reports through stub.www.luisterhuis.nl/reseller.