

easybill Import Tool CSV Import

This document describes the custom import format (csv file) to use when importing orders from external systems that are not yet supported by easybill.

*easybill GmbH, Düsselstr. 21, 41564 Kaarst
www.easybill.de*

Document Version: 0.34 - Last Updated 12.05.14 14:55

1 Overview

1.1 Summary

Merchants can use the easybill custom csv interface to import their orders and invoicing data from shop systems and marketplaces which are not directly supported by easybill. This document describes the import format and the required fields.

2 Overview

2.1 Import via File Upload

We support uploading import files via our web interface. The merchant can simply upload the csv file. This direct upload helps the merchant and/or developers to easily test their files before setting up an import via FTP polling or API calls.

2.2 FTP Polling

For an automatic import the merchant can store FTP host and path information at easybill. Our system will poll the FTP host regularly for new files in the custom csv format and automatically process them.

2.3 REST API Import

We also offer a REST API which can be used to upload the csv-file via a simple HTTP(S) POST request to our servers.

3 File Format CSV

The custom csv format allows merchants to import data into the easybill system when they are using a system which is not directly supported by us.

3.1 General format

We expect the files encoded in UTF-8. Although it's not advised the merchant can change the encoding of the file in their import tool settings to ISO-8859-1 (LATIN/1). If the merchant is free to choose between encodings we strongly advise to use UTF-8.

The fields are separated via semi-colons: ";". The merchant can also change the default settings to use other delimiters like comma "," or colon ":" or tab-space.

Quotes ("") are used as text separator. Line breaks should be either in windows ("\\r\\n") or Linux-style ("\\n"). Quotes in text-fields must be escaped (i.e. replaced by a double-quote: the term *Toy "Plastic Duck"* text becomes *Toy ""Plastic Duck""*).

The csv file must contain the header names of each column. This helps us to extend the format (e.g. new fields which should be imported) without breaking backwards-compatibility. Our system will use the first row (headers) to identify which columns need to be parsed. We don't rely on the order of the fields therefore the import file does not need to have the columns in a specific order.

3.2 Field Types

The fields are defined as follows:

Type	Description	Sample
String	Any text including line-breaks if enclosed with text delimiter ("")	A SAMPLE TEXT
Decimal	Decimal values in the format XXX.YY. Precision of 16,5. That means values up to 9,999,999,999.9999 (9 billions)	100.00 9999999999.99999
Integer	Plain numbers without precision	1
Date	Date format, YYYY-MM-DD	2012-07-19
Datetime	Time format according to ISO-8601 (http://de.wikipedia.org/wiki/ISO_8601)	2012-07-19T19:29:20+02:00

3.3 Order specific fields

The following list of fields contains order specific fields. If an order contains more than a single product/item we'll use the information in the first row for the order. Therefore in subsequent rows this fields, except order_number, can be empty or filled, because they are ignored.

Name (Header)	Comment	Required	Format
order_number	Merchants internal order number to identify an order. Must be unique because our system uses this identifier to prevent multiple imports of the same order.	YES	String (1..255)
order_number_2	Secondary order number. Can be used to store another external order number to this order. In most cases it is not needed.	NO	String (0..255)
email	Email address of customer. Although this field is optional we suggest to use it because we use the email address to send invoices to the customer automatically	NO, but recommended	String (0..255)
name	Name of the customer. If you cannot split the name into first- and lastname because the shop system does not store this separately. Mandatory if not last-and firstname given.	see comment	String (1..255)
firstname	Firstname of the buyer. Not needed if "name" field is set.	see comment	String (1..255)
lastname	Lastname of the buyer. Not needed if "name" field is set.	see comment	String (1..255)
address_1, address_2, address_3	Contains the street number and additional information that should be printed on the invoice. Street should always be at the last entry. Mandatory if no street is given.	see comment	String (1..255)

street	If not additional address information is used you can use “street” to import the street and housenumber. Internally we concatenate the street with the additional address information (see above)	see comment	String (1...255)
zipcode	Zipcode (Postal Code) of the buyer	YES	String (1..255)
city	City	YES	String(1..255)
state	State (e.g. “Bundesland” in Germany) of the buyer	NO	String (0..255)
country	Two letter ISO-Country Code (e.g. DE for Germany) of the buyer ¹	YES	String (2)
phone_number	Buyers phone number. This may be needed for creating delivery orders (e.g. Hermes, DHL)	NO	String (0..255)
fax_number	Buyers fax number. Most often not needed.	NO	String (0..255)
shipping_name	In case of separate shipping address: use this field for the recipient (see name field)	NO	String (0..255)
shipping_firstname	In case of separate shipping address: see firstname	NO	String (0..255)
shipping_lastname	In case of separate shipping address: see lastname field	NO	String (0..255)
shipping_address_1 shipping_address_2 shipping_address_3	In case of separate shipping address: see address_1 – address_3 field	NO	String (0..255)
shipping_street	In case of separate shipping address: see street field.	NO	String (0..255)
shipping_zipcode	In case of separate shipping address: see zipcode	NO	String (0..255)
shipping_city	In case of separate shipping address: see city	NO	String(0..255)

¹ ISO_3166-1_alpha-2, http://en.wikipedia.org/wiki/ISO_3166-1_alpha-2

shipping_state	In case of separate shipping address: see state field	NO	String (0..255)
shipping_country	In case of separate shipping address: see country field	NO	String (2)
customer_username	If buyer has an external username (e.g. eBay nickname) at the shop system, it can be imported and printed on the invoices.	NO	String (0..255)
customer_number	If user already has an customer_number at the shop system this number can be imported and printed on the invoice.	NO	String (0..255)
currency	Currency of the order. Three-letter currency code (e.g. EUR, GBP, USD)	YES	String (0..255)
order_shipping_price	Gross shipping costs If the order has general shipping costs (e.g. independ of ordered products) this shipping costs needs to be imported here.	NO	Decimal (16,5)
purchase_date	Date/Time when the order was purchased by the buyer.	NO, but recommended	Date or Datetime
shipping_date	Date/Time when the order was shipped to the customer.	NO	Date or Datetime, or empty if not yet shipped.
shipping_tracking	Tracking number of shipping	NO	String(0..255)
shipping_type	Type of shipping, e.g. DHL, Hermes. Can be any custom string. Merchants can use this information in a placeholder to print this information on the documents.	NO	String(0..255)
payment_date	Date/Time when the order was paid	NO, but recommended	Date or Datetime, or empty if not paid.

payment_type	Identifier for payment type used. Can be printed on the invoice. No parsing is done here, these will be transferred 1:1 onto the document.	No	String (0..255)
payment_reference	A external payment reference which you may want to be printed on the invoice using the placeholder {{payment_reference}}	No	Text
invoice_number	If the system already generated an invoice number it can be imported to as a reference.	No	String (0..255)
store_id	Identifier for the store (e.g. for multistore solutions). Can be used to modify document text and control document template.	No	String(0..255)
vat_id	VAT-ID of the customer	No	String(0...255)
tax_type	Use this field to override the automatic detection of the tax type which should apply to the invoice. Can be empty or one of the following: <ul style="list-style-type: none"> • intra-community-trade • export 	No	String(0...255)
comments	Comments made by the buyer (e.g. specific delivery instructions etc).	No	Textfield

3.4 Item specific fields

Each row must include item specific information. If an order contains two different items, there must be two rows for this order. The order_number is used to identify order items that belong to a single order.

Furthermore the rows must be consecutive, for example:

Correct:

order-1;john doe;....;first-item;....

order-1;john doe;....;second-item;....

order-2;peter pan;...;first-item;....

Incorrect:

order-1;john doe;...;first-item;....

order-2;peter pan;...;first-item;....

order-1;john doe;...;second-item;....

The order specific information (see 3.3) can be repeated or empty for the subsequent rows and will be ignored by our system. Therefore only the first row of order specific information will be used for the import.

Name (Header)	Comment	Required	Format
item_number	Internal item order number used to identify an item.	NO	String (0..255)
item_number_2	Secondary item order number for internal purposes (e.g. can be printed on the invoice)	NO	String (0..255)
sku	SKU (stock keeping unit) for the ordered product. If defined in the external system this should be stored in the csv file because we use the SKU for certain features (e.g. automatic VAT calculations based on product) etc.	NO, but recommended	String (0..255)
title	Product description that will be printed on the invoice	YES	String (1..255)
title_2	Optional additional description	NO	String (0..255)
quantity	How many products of this kind were ordered	YES	Integer
item_price	Gross item price for a single unit (quantity 1).	YES	Decimal
item_shipping_price	Gross shipping costs for this item	NO	Decimal

vat_percent	Percentage of the VAT rate which applies to this product, e.g. 19.00 (= 19%). If no vat_percent is given we try to calculate the vat percentage by the settings defined by the merchant (e.g. default 19%) and probably looking up SKU specific settings, or use the vat_rate field if it is set.	NO, but recommended	Decimal
vat_rate	Alternatively to vat_percent the vat_rate description can be given. EU-wide we support these types of vat_rates: default (e.g. Germany 19%) discount (e.g. Germany 7%)	NO	String (0..255)
item_type	You can specify an alternative type for this item (e.g. if you're sending discounts along with the item or adding additional payment costs like Cash on Delivery / Nachnahme)	NO	Valid contents: 'product' (default) 'discount' 'virtual'
item_weight	Weight of the item. Can be used in the resulting document as template variable <code>{{item_weight}}</code>	NO	
variant_yourKey	You can specify as many additional variant information about the ordered item as you need. These information are not used by our system but you can access and use this information using the placeholder variables. Example: If you specify variant_color you can use the placeholder {{item_variant.color}} to build a proper product title on your invoice.		Examples: variant_color variant_size variant_model Please note that the second part of the variant_yourKey name can only consist of letters, dash (-), underscore (_) and numbers.

3.5 Comments

You can add comment lines to the generated CSV file for any purposes. We ignore any lines starting with `#!`. If you are developer of an external module or shopping system you should at least add the following comment to identify the exporting system:

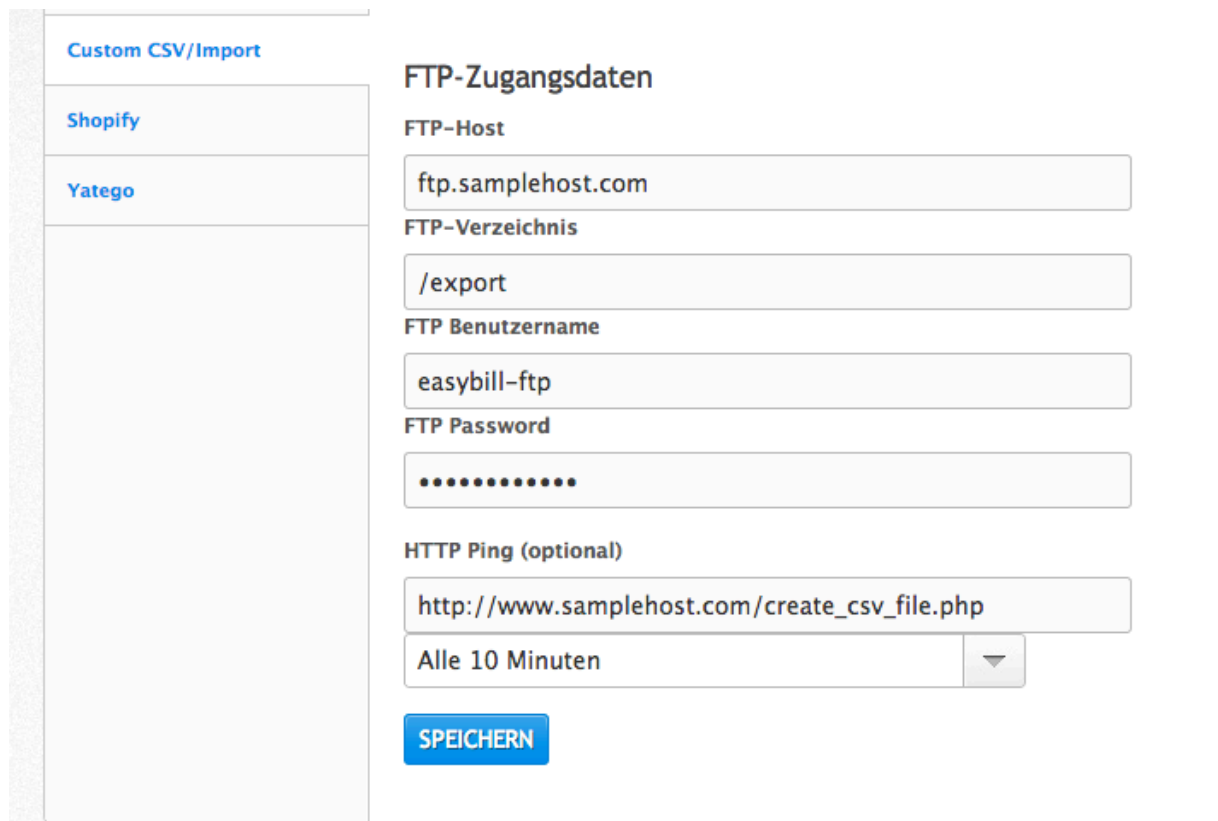
```
#! exported-by: YOUR-SYSTEM-NAME
```

This helps us identifying problems with CSV files you've created. We also show the merchant `YOUR-SYSTEM-NAME` in our frontend. This helps the merchant to identify imports made by your shopping system (e.g. in case the merchant is using the Custom CSV import from multiple systems).

4 FTP Polling

4.1 Settings

Our system can periodically query the server for the latest created orders. At import.easybill.de the merchant can setup his ftp account where we can download the orders as a CSV file.



The screenshot shows a web interface for configuring FTP access. On the left, there is a sidebar with menu items: 'Custom CSV/Import', 'Shopify', and 'Yatego'. The main content area is titled 'FTP-Zugangsdaten' and contains the following fields:

- FTP-Host:** A text input field containing 'ftp.samplehost.com'.
- FTP-Verzeichnis:** A text input field containing '/export'.
- FTP Benutzername:** A text input field containing 'easybill-ftp'.
- FTP Password:** A text input field with masked characters (dots).
- HTTP Ping (optional):** A text input field containing 'http://www.samplehost.com/create_csv_file.php'.
- Frequency:** A dropdown menu currently set to 'Alle 10 Minuten'.

At the bottom of the form is a blue button labeled 'SPEICHERN'.

Optionally the merchant can specify an url which we call periodically to initiate the creation of the csv file and try to to fetch it afterwards.

5 REST API

5.1 Overview

We also offer a simple REST API which can be used to upload the CSV file to our servers and initiate the import process. The API is available under

<https://import.easybill.de/api/v1>

5.2 Authentication

When making HTTP calls to our REST API we check the following HTTP headers to authenticate the user:

Name	Description
X-EASYBILL-USER-ID	This is the User ID of the easybill account.
X-EASYBILL-AUTH-KEY	This is the Authentication key which is also used for accessing the SOAP API. It can be found at easybill.de under “Meine Firma” / “Einstellungen” / “Soap-Schnittstelle”.

5.3 Available Methods

5.3.1 Show an imported order

GET `/api/v1/order/:order_number.json` (or `/api/v1/order/:order_number.xml`)

This method returns all information about an imported order in XML or JSON format.

You need to specify the order number of the order you want to fetch. Example call:

`https://import.easybill.de/api/v1/order/123-1222-1222`

5.3.2 Upload orders as Custom CSV file

POST `/api/v1/orders/upload`

Use this method call to upload a CSV file (as specified in the format accord. Section 3 of this documentation). The file should be encoded and uploaded in UTF-8 encoding as a raw POST file.

By default, the orders in the uploaded CSV file are imported but we don't automatically create invoices. If you want us to create invoices for the imported orders right after uploading the file, please add the query parameter `?auto_export=true` to the API call.