Atlas Copco Quotation System

*QuickQuote*

**Data Management**

Contents

[Data overview 2](#_Toc363811979)

[Product Data 2](#_Toc363811980)

[Add, remove and edit product data 2](#_Toc363811981)

[Cover Letters, Notes, and Terms and Conditions 4](#_Toc363811982)

[Standard letters 4](#_Toc363811983)

[Load/Saved Letters 4](#_Toc363811984)

[Notes 4](#_Toc363811985)

[Terms and Conditions 4](#_Toc363811986)

[Product Descriptions and Technical Specifications 4](#_Toc363811987)

[Exchange rate and Currencies 6](#_Toc363811988)

[Add, change and delete currencies 6](#_Toc363811989)

[Address for footer 7](#_Toc363811990)

[Add, change and delete addresses 7](#_Toc363811991)

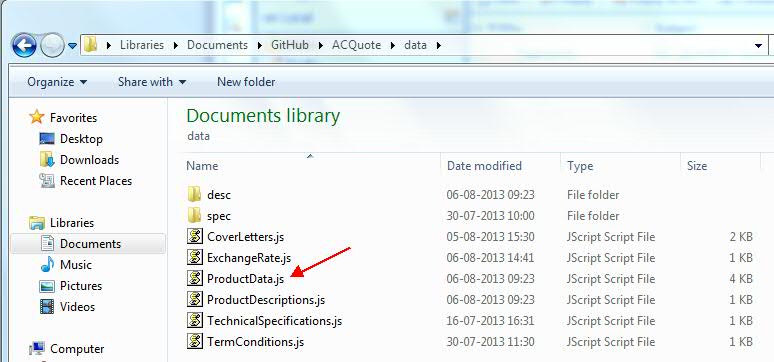
# Data overview

All data is stored in .js files in the /data folder. The format is an array delimited with [] brackets with object that are separated with {} brackets. The example below contains two object with two properties each, ‘title’ and ‘value’. The files may be opened in any simply text editor such as notepad.

*var NAME = [{title: ‘first, value: ‘11’}, {title: ‘second’, value: ‘22’}]*

# Product Data

The product data the system uses is stored in the file named: “ProductData.js” in the ./data folder. To open the file just right-click and choose a text editor like notepad.



The format is CSV JSON data after the variable named PRODUCTS. The file has to contain “var PRODUCTS = [{data}] where the [] encloses the array and the {} encloses each object in the array.

An example is given below:

*var PRODUCTS =[{"Part":57554461,"Description":"FAMILY,T3W","StdCost":358145.42,"Dep":null,"Class":"Base","BoxType":""}, {"Part":54539457,"Description":"WARRANTY,STANDARD,T3W","StdCost":0.00,"Dep":57554461,"Class":"Pump","BoxType":"Radio"}]*

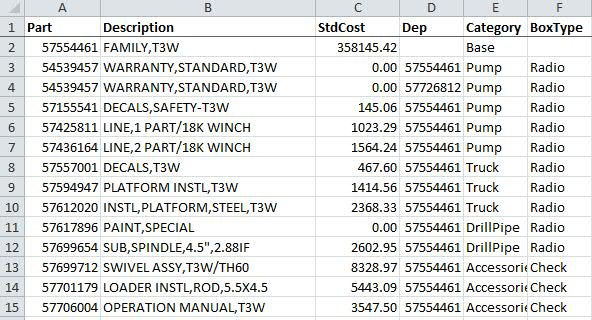
## Add, remove and edit product data

To edit the data you may use an excel file, where the columns represent the objects properties and the rows represent the objects. The following properties shall be included:

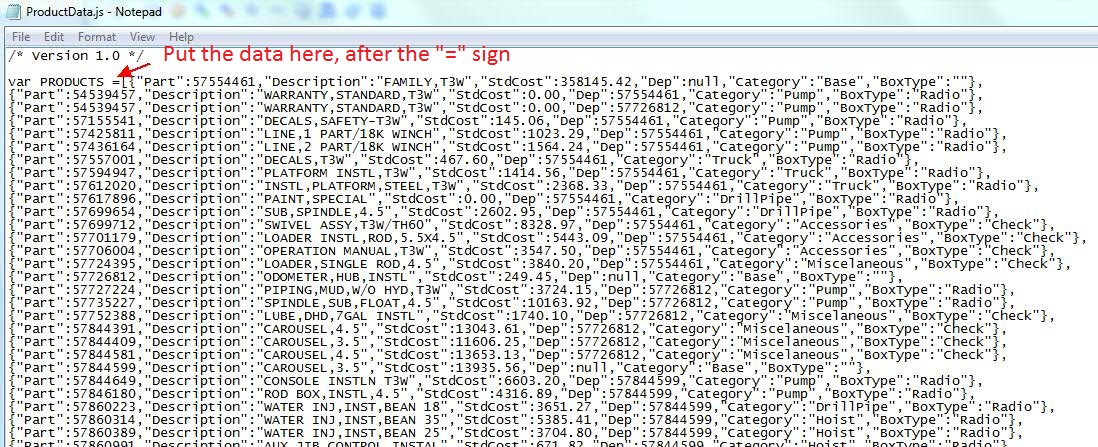
* **Part** – representing the unique part number
* **StdCost** – the cost of the product
* **Dep** – defines on which base machine the product is depending on. The unit will only show when the base it is depending on is selected. The value shall be the **Part** of the product it is depending on. If it is a base itself and thus not depending on any other unit the value shall be set to *null.*
* **Category** – Defines the category of the product, e.g. “Mud Pump” or “Accessories”. Product within the same category will appear together
* **BoxType** – defines if the selection shall be exclusive (radio box) or if multiple products within the same category may be selected.

**Note!** Make sure to remove any currency formatting from the standard cost in excel, e.g. 358,145.42 should be written like 358145.42.

Example of excel data:



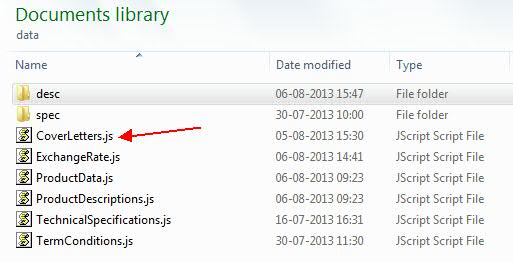
The excel file may then be converted to JSON format. This can be done e.g. by using an online tool like CSV Converter (<http://www.convertcsv.com/csv-to-json.htm>) or Mr. Data Converter: (<http://shancarter.github.io/mr-data-converter/>). Copy the excel data including the column names and paste it into the browser. Then copy the output and put it after the “var PRODUCTS =” in the ProductData.js.



# Cover Letters, Notes, and Terms and Conditions

## Standard letters

The standard cover letter are stored in ./data folder named CoverLetters.js.



Format is an array with objects (the letters) where each object has a title and content. Example below:

*var LETTERS = [{title: 'Letter 1', content: 'Lorem ipsum dolor sit amet…’},{ title: 'Letter 2', content:’Sed ut perspiciatis…’}]*

To **change** the contents of a letter just change the text after “content:”. The contents of the letters may use HTML markup. The title will be shown in the dropdown menu. To **add** a letter simply add a {} bracket after the last letter with a title and content. See example below, added letter 3 in **bold**:

*var LETTERS = [{title: 'Letter 1', content: 'Lorem ipsum dolor sit amet…’},{ title: 'Letter 2', content:’Sed ut perspiciatis…’}****, {title: ‘Letter 3’, content: ’Content of letter three’ }****]*

## Load/Saved Letters

Letter that are loaded and saved should be stored in a .txt file, preferable in the ./’Saved Letters’ folder. The content may use HTML markup.

## Notes

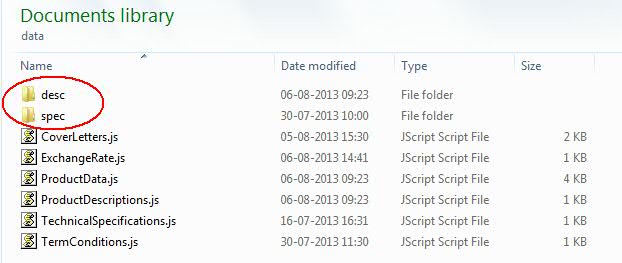
Notes may be saved and loaded the same way as the letter and use HTML markup.

## Terms and Conditions

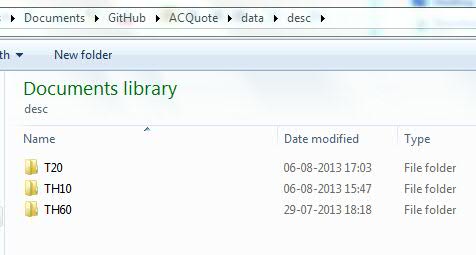
Terms and conditions are stored in TermConditions.js and behave the same way as the Cover Letters.

# Product Descriptions and Technical Specifications

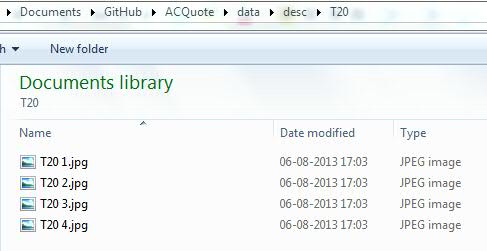
The product descriptions and technical specifications are stored as image files in the folders /data/desc and /data/spec with folder named the after the title e.g. the description of the TH60 will the stored in /data/desc/TH60 with file names TH60 1.jpg, TH60 2.jpg and so forth. Whenever a new specification or description is to be added a new folder has to be created. See picture below



*In the data folder*



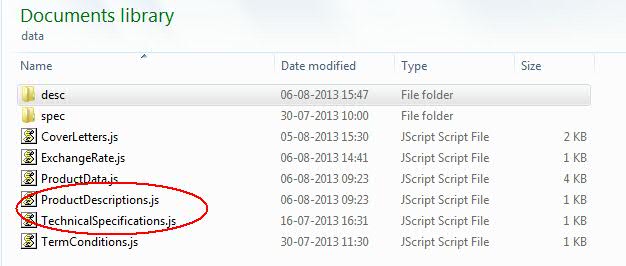
*In the desc folder*



*In the T20 folder*

**Note!** The images has to be named “title x” where x is the page number starting at 1 and be in .jpg format.

The ProductDescription.js and TechnicalSpecification.js contain information about the descriptions and specifications and have to be modified whenever a change is being made.



The data stored in the files contain information about the title, which has to be the same as the folder name, as well as the number of pages (images) that each folder contain. If there is only one folder with four images if would look like below:

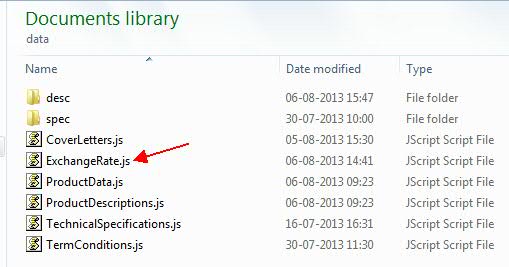
*var DESCRIPTIONS = [{"Name": "TH60", "ImageCount":4}]*

To **add** a new product folder just add a new object in {} brackets, as the example below in **bold.**

*var DESCRIPTIONS = [{"Name": "TH60", "ImageCount":4}****, {"Name": "T20", "ImageCount":4}****]*

# Exchange rate and Currencies

The file that contains the currencies and exchange rate and located in the ./data folder.



The data is stored in an array enclosed by [] brackets and each currency is enclosed by {} brackets. Each currency has two properties, Name and Rate. The Name is the abbreviation for the currency (which will be shown in the dropdown menu) and Rate is the exchange rate to the standard currency, in this example the standard currency in the USD.

var CURRENCIES = [{Name:"USD", Rate:"1.0"}, {Name:"INR", Rate:"60.0"}, {Name:"SEK", Rate:"6.75"}]

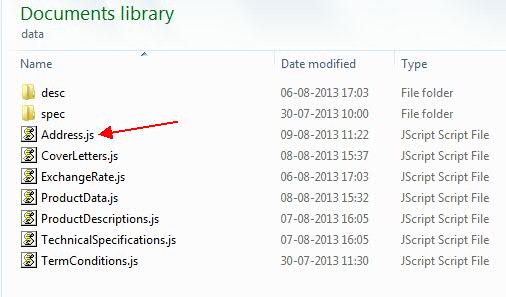
## Add, change and delete currencies

To edit the currency document just right click and open it in a text editor like notepad. A new currency can be added by adding {} brackets with a name and a rate. Below EUR is added in **bold:**

var CURRENCIES = [{Name:"USD", Rate:"1.0"}, {Name:"INR", Rate:"60.0"}, {Name:"SEK", Rate:"6.75"}**, {Name:”EUR”, Rate:”0.75”}**]

# Address for footer

The addresses for the footer is stored in Address.js in the /data folder. The Address.js file may to open with a text editor such as Notepad.

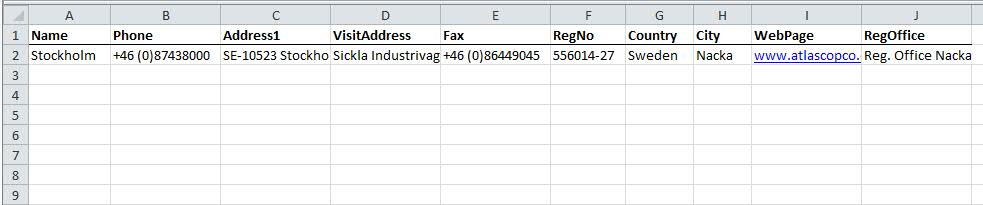


The data is stored in an array enclosed by [] brackets and each address is enclosed by {} brackets. Each property of the address is separated by a comma, see example below.

The Name property is what will be visible to the user in the dropdown menu.

## Add, change and delete addresses

To change the address you may use an excel file for input, similar to what you use for the ProductData. To add an address simply add a line in the excel file. Example of excel data below:



The excel file may then be converted to JSON format. This can be done e.g. by using an online tool like CSV Converter (<http://www.convertcsv.com/csv-to-json.htm>. Simply copy the excel data into the web browser and convert it, then copy the converted data and paste it in the Address.js file after the “var ADDRESS =”, same principle as for the other data files.