Contents

Introduction 2

Revision History 4

Configuration Settings 5

Functions and Parameters 6

- HelloWorld 6
- createCustomer 6
- editCustomer 7
- getCustomerDetails 8
- getAllAlternativeStockCount 8
- getAllAvailableStock 9
- getAllBrands 10
- getAllCategories 11
- getAllDepartments 11
- getAllDiscontinuedBarcodes 12
- getAllHeadQuartersStock 12
- getAllInfo 13
  - XML Schema 14
- getAllStock 18
- getImage 19
- getItemBrand 19
- getItemDescription 20
- getItemExtendedDescription 20
- getItemID 20
- getItemImageFilename 21
- getItemPrice 21
- getItemPriceA 22
- getItemPriceB 22
- getItemPriceC 22
- getItemSize 23
- getItemStockQuantity 23
- getOrderStatus 23
- placeXmlOrder 24

XML Schema 25
placeXmlOrderCommitted 27
searchForCustomers 28
Logging 29

Introduction

The API was created to utilise functionality in Rms without having an in depth knowledge of the system.

A reference to the web service will be a url resembling:

http://storename.dyndns.org/RMSAPI/RMSAPI.asmx

Revision History

Version	Additions / Modifications
1.0	Document creation.
2.0	Added getAllInfo(), placeXmlOrder() and getHeadQuartersStock() functions.
2.1	• getAllInfo() – In *item.xsd* allow apostrophes in *filterValue*. • In placeXmlOrder() allow selling price to be specified.
2.2	• Remove placeOrder() function. • Rename getHeadQuartersStock() to getAllHeadQuartersStock(). • Add getAllDepartments() and getAllCategories() functions. • Add getImage() for downloading product images.
2.3	• Add departmentid to getAllDepartments(). • Add departmentid and categoryid to getAllCategories().
2.4	Add getAllDiscontinuedBarcodes() and getOrderStatus() web methods.
2.5	• Add getAllBrands() and getAllAvailableStock() web methods to retrieve nett stock. • Fix getAllStock() to return item.quantity.
2.5.1	The *imageLocation* setting in web.config changed to an absolute path.
2.6	• Added the following web methods: createCustomer(), editCustomer(), getCustomerDetails(). • Change placeXmlOrder() method to allow customerID to be specified in xml. • Added logging for placeXmlOrder().
2.7	• Subtract Item.QuantityCommitted from Item.Quantity where on hand requested. Applied to the following methods: getItemStockQuantity(), getAllAvailableStock(), getAllStock(), getAllHeadQuartersStock(), getAllInfo() • Add getAllAlternativeStockCount () method: Item.Quantity - Item.Committed - TransactionHold stock + Item.Content • Change getAllStock(), getAllAvailableStock() and getAllAlternativeStockCount () to accept a barcode. If the barcode is blank all items are returned.
2.8	Change editCustomer() to accept different sets of parameters.
2.9	For POS users only allow compatible stock methods to be accessed.
2.10	• Make placeXmlOrder() and getOrderStatus() compatible with POS. • Allow port number to be appended to WSDL location.
2.11	Add searchForCustomers() function.
2.12	Add placeXmlOrderCommitted() function.

Configuration Settings

The configuration settings are solely for Emporio's use. These are stored in the web.config file located in the root directory of the web service.

Name	Description	Sample Value
appSettings		
webCustomerID	The value for the customer id used when creating transaction on hold entries.	4544
imageLocation	The path where images are located.	C:\Images
connectionStrings		
rmsStore	Database connection string to store database.	Data Source=.;Initial Catalog=rms;User Id=sa;Password=password
rmsHQ	Database connection string to headquarters database.	Data Source=.;Initial Catalog=rmsHq;User Id=sa;Password=password

Functions and Parameters

All functions have an *errorMessage* parameter. This is an input/output parameter and as its name suggests, it will pass back an error message.

HelloWorld

This function simply returns the string **Hello World**. It is useful when starting to work with the web service. Once this simple function is working, the developer is ready to move on to other aspects.

Input parameters: None

Output parameter: HelloWorld() [string]

createCustomer

This function creates a new customer in RMS.

Input parameters:

Parameter Name	Type	Max length	Info
firstName	string	30	
lastName	string	50	
dateOfBirth	string	10	The date in US format (mm/dd/yyy), e.g. 01/31/2011
emailAddress	string	255	
company	string	50	
address	string	50	
address2	string	50	
city	string	50	
county	string	20	
postCode	string	15	
country	string	20	
phoneNumber	string	30	
faxNumber	string	30	
newsLetter	boolean		This specifies whether the customer is on the mailing list, e.g. True or False
postalOffers	boolean		This specifies whether the customer receives postal offers, e.g. True or False
gender	string	6	Male or Female
userName	string	30	

password	string	30	
errorMessage	string		Initial value will be empty

Output parameters:

createCustomer () [integer] – This will be the customer id of the newly created customer.

errorMessage [string] – If error occurred it will contain an error message.

editCustomer

This function can be overloaded and is used to edit an existing customer in RMS.

Input parameters (Full List):

Parameter Name	Type	Max length	Info
customerID	integer		This is the primary key for a customer in RMS
firstName	string	30	
lastName	string	50	
dateOfBirth	string	10	The date in US format (mm/dd/yyy), e.g. 01/31/2011
emailAddress	string	255	
company	string	50	
address	string	50	
address2	string	50	
city	string	50	
county	string	20	
postCode	string	15	
country	string	20	
phoneNumber	string	30	
faxNumber	string	30	
newsLetter	boolean		This specifies whether the customer is on the mailing list, e.g. True or False
postalOffers	boolean		This specifies whether the customer receives postal offers, e.g. True or False
gender	string	6	Male or Female
userName	string	30	
password	string	30	
offerValue	boolean		
offerType	string		Acceptable values are newsletter or postaloffers
errorMessage	string		Initial value will be empty

- editCustomer (customerID, firstName, lastName, dateOfBirth, gender, emailAddress, phoneNumber, faxNumber, errorMessage)
- editCustomer (customerID, password, errorMessage)
- editCustomer (customerID, offerValue, offerType, errorMessage)
- editCustomer (customerID, firstName, lastName, gender, company, address, address2, city, county, postCode, country, errorMessage)
- editCustomer (customerID, firstName, lastName, dateOfBirth, gender, emailAddress, company, address, address2, city, county, postCode, country, newsLetter, postalOffers, errorMessage)

Output parameter:

errorMessage [string] – If error occurred it will contain an error message.

getCustomerDetails

This function returns the price level set for the customer.

Input parameters:

customerID [integer] – This is the primary key for a customer in RMS.

errorMessage [string] – Initial value will be empty.

Output parameters:

getCustomerDetails () [string] – This will be the price level of the customer which can be either one of the following values:

- Standard
- Price A
- Price B
- Price C

errorMessage [string] – If error occurred it will contain an error message.

getAllAlternativeStockCount

Returns the stock levels that exist in RMS in xml format. This method allows the store to use an alternative stock count (Item.SubDescription3), in addition to the normal stock count.

This takes the current stock levels and subtracts the stock needed to fulfil orders that are waiting to be processed (transaction on hold). The transactions on hold include web orders and any others that

were manually placed on hold. The committed value is also subtracted, this is any items that have been received and are on an open back order. Finally the alternative stock for the item is added.

Inactive items returned are only those have available stock.

Input parameters:

barcode [string] – If an empty string is used allitems will be returned.

errorMessage [string] – Initial value will be empty.

Output parameters:

getAllAlternativeStockCount () [string] – Sample output:

```
  <stock>
    <item>
      <barcode>5901330004070</barcode>
      <quantityonhand>0</quantityonhand>
    </item>
    <item>
      <barcode>5023652050501</barcode>
      <quantityonhand>7</quantityonhand>
    </item>
    <item>
      <barcode> 698324</barcode>
      <quantityonhand>1</quantityonhand>
    </item>
</ stock>
```

errorMessage [string] – If error occurred it will contain an error message.

getAllAvailableStock

Returns the stock levels that exist in RMS in xml format. This takes the current stock levels and subtracts the stock needed to fulfil orders that are waiting to be processed (transaction on hold). The transactions on hold include web orders and any others that were manually placed on hold. The committed value is also subtracted, this is any items that have been received and are on an open back order.

Inactive items returned are only those have available stock.

Input parameters:

barcode [string] – If an empty string is used allitems will be returned.

errorMessage [string] – Initial value will be empty.

Output parameters:

getAllAvailableStock() [string] – Sample output:

```
  <stock>
    <item>
      <barcode>5901330004070</barcode>
      <quantityonhand>0</quantityonhand>
    </item>
    <item>
      <barcode>5023652050501</barcode>
      <quantityonhand>7</quantityonhand>
    </item>
    <item>
      <barcode> 698324</barcode>
      <quantityonhand>1</quantityonhand>
    </item>
</ stock>
```

errorMessage [string] – If error occurred it will contain an error message.

getAllBrands

Returns a pipe delimited string containing the list of brands belonging to items that have WebItem=True. The list will be returned in alphabetical order.

Input parameters:

errorMessage [string] – Initial value will be empty.

Output parameters:

getAllBrands () [string] – Sample output:

```
Ainsworths|Alaskan Flower Essences|Albissola|Alma Win|Aloe Dent|Aloe
Pura|Alva|Animal|Antipodes|Aubrey Organics|Australian Bush Flower|Baby Buds|Balmy
Pyjamas|Barleans|Beaming Baby|becothings|Benfatto Nutrition|Bentley
Organic|BioCare|Bioforce|Biosun|Boo Boo|Bounce|Bush Flower Essences|Care
Pharma|Caudalie|Chemical Nutrition|CherryActive|Citroxx|Clearly Herbal|Comvita
```

errorMessage [string] – If error occurred it will contain an error message.

getAllCategories

Input parameters:

errorMessage [string] – Initial value will be empty.

Output parameters:

getAllCategories () [string] – Sample output:

```
<categories>
  <category>
    <categoryname>Seeds</categoryname>
    <categorycode>11B</ categorycode>
    <departmentname>Pulse Nut Fruit</ departmentname>
  </category>
  <category>
    <categoryname>Specific Blends</categoryname>
    <categorycode>24F</ categorycode>
    <departmentname>Specific Supplements</ departmentname>
  </category>
  <category>
    <categoryname>Fresh Bread</categoryname>
    <categorycode>16B</ categorycode>
    <departmentname>Bread</ departmentname>
  </category>
</categories>
```

errorMessage [string] – If error occurred it will contain an error message.

getAllDepartments

Input parameters:

errorMessage [string] – Initial value will be empty.

Output parameters:

getAllDepartments () [string] – Sample output:

```
<departments>
  <department>
    <departmentname>Pulse Nut Fruit</ departmentname>
    <departmentcode> HL-10</ departmentcode>
  </department>
  <department>
    <departmentname>Homeware</ departmentname>
    <departmentcode>HL-14</ departmentcode>
```

```
  </department>
  <department>
    <departmentname>Baby Products</ departmentname>
    <departmentcode>BS-02</ departmentcode>
  </department>
</departments>
```

errorMessage [string] – If error occurred it will contain an error message.

getAllDiscontinuedBarcodes

Returns a pipe delimited string containing all items that have WebItem=False.

Input parameters:

errorMessage [string] – Initial value will be empty.

Output parameters:

getAllDiscontinuedBarcodes () [string] – Sample output:

```
6009644651702|6009644651696|9771745433255|6009644652044|9600964465163|6009644651689|0705875100168|5028931046411|5022129040649|5022129049055|5022129046436|5028931046077|5022129040588|5022129040489|5022129043497|5022129043350|5022129043473|5028931046114|5022129042483|5022129044616|5022129044654|5022129040250|5022129025516|5022129040199|5022129020016|5022129040236|5022129040229|6009620610037
```

errorMessage [string] – If error occurred it will contain an error message.

getAllHeadQuartersStock

Returns the total stock for all stores from the hq database in xml format. The committed value is used in the calculation which is any items that have been received and are on an open back order.

Input parameter: errorMessage [string] – Initial value will be empty.

Output parameters:

getAllHeadQuartersStock() [string] – Sample output:

```
< hqstock>
  <item>
    <barcode>0083000007590</barcode>
    <quantityonhand>1</quantityonhand>
  </item>
```

```
    <item>
      <barcode>9780859699396</barcode>
      <quantityonhand>2</quantityonhand>
    </item>
    <item>
      <barcode> 9780859699396</barcode>
      <quantityonhand>0</quantityonhand>
    </item>
</ hqstock>
```

errorMessage [string] – If error occurred it will contain an error message.

getAllInfo

Used to get information such as brand, description, stock levels, etc. Returns an xml string containing the results of the query. When date values are specified in the filter these must be in US format.

Input parameters:

infoRequestXml [string] – Sample data will look like:

```
<?xml version="1.0"?>
<item xmlns="rmsItem">
  <columns>
    <column>description</column>
    <column>department</column>
    <column>brand</column>
    <column>lastsold</column>
    <column>lastupdated</column>
    <column>quantityonhand</column>
    <column>weight</column>
  </columns>
  <filters>
    <filter>
      <filterColumn>quantityonhand</filterColumn>
      <operator>greaterthan</operator>
      <filterValue>20</filterValue>
    </filter>
    <filter>
      <filterColumn>lastsold</filterColumn>
      <operator>greaterthan</operator>
      <filterValue>01-31-2011</filterValue>
    </filter>
  </filters>
  <sortColumns>
    <sortColumn>
      <sortColumnName>lastsold</sortColumnName>
      <sortType>ascending</sortType>
    </sortColumn>
    <sortColumn>
      <sortColumnName>quantityonhand</sortColumnName>
      <sortType>descending</sortType>
    </sortColumn>
  </sortColumns>
</item>
```

SOAP Request

```
<?xml version="1.0" encoding="utf-8"?><soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body>
<getAllInfo xmlns="http://emporiouk.com/RMSAPI">
<infoRequestXml>&lt;?xml version="1.0"?&gt;
&lt;item xmlns="rmsItem"&gt;
  &lt;columns&gt;
    &lt;column&gt;description&lt;/column&gt;
    &lt;column&gt;department&lt;/column&gt;
    &lt;column&gt;brand&lt;/column&gt;
    &lt;column&gt;lastsold&lt;/column&gt;
    &lt;column&gt;lastupdated&lt;/column&gt;
    &lt;column&gt;quantityonhand&lt;/column&gt;
    &lt;column&gt;weight&lt;/column&gt;
  &lt;/columns&gt;
  &lt;filters&gt;
    &lt;filter&gt;
      &lt;filterColumn&gt;quantityonhand&lt;/filterColumn&gt;
      &lt;operator&gt;greaterthan&lt;/operator&gt;
      &lt;filterValue&gt;20&lt;/filterValue&gt;
    &lt;/filter&gt;
    &lt;filter&gt;
      &lt;filterColumn&gt;lastsold&lt;/filterColumn&gt;
      &lt;operator&gt;greaterthan&lt;/operator&gt;
      &lt;filterValue&gt;01-01-2005&lt;/filterValue&gt;
    &lt;/filter&gt;
  &lt;/filters&gt;
  &lt;sortColumns&gt;
    &lt;sortColumn&gt;
      &lt;sortColumnName&gt;lastsold&lt;/sortColumnName&gt;
      &lt;sortType&gt;ascending&lt;/sortType&gt;
    &lt;/sortColumn&gt;
    &lt;sortColumn&gt;
      &lt;sortColumnName&gt;quantityonhand&lt;/sortColumnName&gt;
      &lt;sortType&gt;descending&lt;/sortType&gt;
    &lt;/sortColumn&gt;
  &lt;/sortColumns&gt;
&lt;/item&gt;</infoRequestXml>
<errorMessage />
</getAllInfo>
</soap:Body>
</soap:Envelope>
```

The xml is validated against the following schema:

XML Schema

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
targetNamespace="rmsItem" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="item">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="columns">
          <xs:complexType>
```

```
            <xs:sequence>
              <xs:element maxOccurs="unbounded" name="column">
                <xs:simpleType>
                  <xs:restriction base="xs:string">
                    <xs:enumeration value="barcode"/>
                    <xs:enumeration value="brand"/>
                    <xs:enumeration value="size"/>
                    <xs:enumeration value="colour"/>
                    <xs:enumeration value="description"/>
                    <xs:enumeration value="extendeddescription"/>
                    <xs:enumeration value="department"/>
                    <xs:enumeration value="category"/>
                    <xs:enumeration value="binlocation"/>
                    <xs:enumeration value="weight"/>
                    <xs:enumeration value="cost"/>
                    <xs:enumeration value="price"/>
                    <xs:enumeration value="pricea"/>
                    <xs:enumeration value="priceb"/>
                    <xs:enumeration value="pricec"/>
                    <xs:enumeration value="msrp"/>
                    <xs:enumeration value="datecreated"/>
                    <xs:enumeration value="lastupdated"/>
                    <xs:enumeration value="lastsold"/>
                    <xs:enumeration value="quantityonhand"/>
                    <xs:enumeration value="reorderpoint"/>
                    <xs:enumeration value="restocklevel"/>
                    <xs:enumeration value="inactive"/>
                    <xs:enumeration value="webitem"/>
                    <xs:enumeration value="vatcode"/>
                  </xs:restriction>
                </xs:simpleType>
              </xs:element>
              <!-- column -->
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <!-- columns -->
        <xs:element name="filters">
          <xs:complexType>
            <xs:sequence>
              <xs:element maxOccurs="unbounded" name="filter">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="filterColumn">
                      <xs:simpleType>
                        <xs:restriction base="xs:string">
                          <xs:enumeration value="barcode"/>
                          <xs:enumeration value="brand"/>
                          <xs:enumeration value="size"/>
                          <xs:enumeration value="colour"/>
                          <xs:enumeration value="description"/>
                          <xs:enumeration value="extendeddescription"/>
                          <xs:enumeration value="department"/>
                          <xs:enumeration value="category"/>
                          <xs:enumeration value="binlocation"/>
                          <xs:enumeration value="weight"/>
                          <xs:enumeration value="cost"/>
                          <xs:enumeration value="price"/>
                          <xs:enumeration value="pricea"/>
                          <xs:enumeration value="priceb"/>
                          <xs:enumeration value="pricec"/>
                          <xs:enumeration value="msrp"/>
```

```
                    <xs:enumeration value="datecreated"/>
                    <xs:enumeration value="lastupdated"/>
                    <xs:enumeration value="lastsold"/>
                    <xs:enumeration value="quantityonhand"/>
                    <xs:enumeration value="reorderpoint"/>
                    <xs:enumeration value="restocklevel"/>
                    <xs:enumeration value="inactive"/>
                    <xs:enumeration value="webitem"/>
                    <xs:enumeration value="vatcode"/>
                  </xs:restriction>
                </xs:simpleType>
              </xs:element>
              <xs:element name="operator">
                <xs:simpleType>
                  <xs:restriction base="xs:string">
                    <xs:enumeration value="equal"/>
                    <xs:enumeration value="notequal"/>
                    <xs:enumeration value="like"/>
                    <xs:enumeration value="greaterthan"/>
                    <xs:enumeration value="greaterthanequal"/>
                    <xs:enumeration value="lessthan"/>
                    <xs:enumeration value="lessthanequal"/>
                  </xs:restriction>
                </xs:simpleType>
              </xs:element>
              <xs:element name="filterValue">
                <xs:simpleType>
                  <xs:restriction base="xs:string">
                    <xs:minLength value="0" />
                    <xs:maxLength value="30" />
                    <!--<xs:pattern value="^[^']*$"/>-->
                  </xs:restriction>
                </xs:simpleType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <!-- filter -->
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <!-- filters -->
  <xs:element name="sortColumns" minOccurs="0">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" name="sortColumn" minOccurs="0">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="sortColumnName" minOccurs="0">
                <xs:simpleType>
                  <xs:restriction base="xs:string">
                    <xs:enumeration value="barcode"/>
                    <xs:enumeration value="brand"/>
                    <xs:enumeration value="size"/>
                    <xs:enumeration value="colour"/>
                    <xs:enumeration value="description"/>
                    <xs:enumeration value="extendeddescription"/>
                    <xs:enumeration value="department"/>
                    <xs:enumeration value="category"/>
                    <xs:enumeration value="binlocation"/>
                    <xs:enumeration value="weight"/>
                    <xs:enumeration value="cost"/>
```

```
                        <xs:enumeration value="price"/>
                        <xs:enumeration value="pricea"/>
                        <xs:enumeration value="priceb"/>
                        <xs:enumeration value="pricec"/>
                        <xs:enumeration value="msrp"/>
                        <xs:enumeration value="datecreated"/>
                        <xs:enumeration value="lastupdated"/>
                        <xs:enumeration value="lastsold"/>
                        <xs:enumeration value="quantityonhand"/>
                        <xs:enumeration value="reorderpoint"/>
                        <xs:enumeration value="restocklevel"/>
                        <xs:enumeration value="inactive"/>
                        <xs:enumeration value="webitem"/>
                        <xs:enumeration value="vatcode"/>
                      </xs:restriction>
                    </xs:simpleType>
                  </xs:element>
                  <xs:element name="sortType" minOccurs="0">
                    <xs:simpleType>
                      <xs:restriction base="xs:string">
                        <xs:enumeration value="ascending"/>
                        <xs:enumeration value="descending"/>
                      </xs:restriction>
                    </xs:simpleType>
                  </xs:element>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
            <!-- sortColumn -->
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <!-- sortColumns -->
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

errorMessage [string] – Initial value will be empty.

Output parameters:

getAllInfo () [string] – Sample output:

```
<items>
  <item>
    <description>Flapjack Forest Berries box</description>
    <department>Sports Products</department>
    <brand>PhD Nutrition</brand>
    <lastsold>2010-04-28T11:26:48+01:00</lastsold>
    <lastupdated>2011-02-17T13:32:42+00:00</lastupdated>
    <quantityonhand>126</quantityonhand>
    <weight>0</weight>
  </item>
  <item>
    <description>Promax Bars Choc Orange</description>
    <department>Sport Products</department>
    <brand>MAXIMUSCLE</brand>
    <lastsold>2010-05-04T15:55:41+01:00</lastsold>
    <lastupdated>2011-02-17T14:09:13+00:00</lastupdated>
```

```
    <quantityonhand>24</quantityonhand>
    <weight>0</weight>
  </item>
</items>
```

SOAP Response

```
<?xml version="1.0" encoding="utf-8"?><soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body>
<getAllInfoResponse xmlns="http://emporiouk.com/RMSAPI">
<getAllInfoResult>&lt;items&gt;
  &lt;item&gt;
    &lt;description&gt;Flapjack Forest Berries box&lt;/description&gt;
    &lt;department&gt;Sports Products&lt;/department&gt;
    &lt;brand&gt;PhD Nutrition&lt;/brand&gt;
    &lt;lastsold&gt;2010-04-28T11:26:48+01:00&lt;/lastsold&gt;
    &lt;lastupdated&gt;2011-02-17T13:32:42+00:00&lt;/lastupdated&gt;
    &lt;quantityonhand&gt;126&lt;/quantityonhand&gt;
    &lt;weight&gt;0&lt;/weight&gt;
  &lt;/item&gt;
  &lt;item&gt;
    &lt;description&gt;Promax Bars Choc Orange&lt;/description&gt;
    &lt;department&gt;Sports Products&lt;/department&gt;
    &lt;brand&gt;MAXIMUSCLE&lt;/brand&gt;
    &lt;lastsold&gt;2010-05-04T15:55:41+01:00&lt;/lastsold&gt;
    &lt;lastupdated&gt;2011-02-17T14:09:13+00:00&lt;/lastupdated&gt;
    &lt;quantityonhand&gt;24&lt;/quantityonhand&gt;
    &lt;weight&gt;0&lt;/weight&gt;
  &lt;/item&gt;
&lt;/items&gt;</getAllInfoResult>
<errorMessage />
</getAllInfoResponse>
</soap:Body>
</soap:Envelope>
```

errorMessage [string] – If error occurred it will contain an error message.

getAllStock

Returns the stock levels that exist in the RMS item table in xml format. The committed value is subtracted, this is any items that have been received and are on an open back order. This does not take into account the stock needed to fulfil orders that are waiting to be processed (transaction on hold).

Inactive items returned are only those have available stock.

Input parameters:

barcode [string] – If an empty string is used allitems will be returned.

errorMessage [string] – Initial value will be empty.

Output parameters:

getAllStock() [string] – Sample output:

```
  <stock>
    <item>
      <barcode>0083000007590</barcode>
      <quantityonhand>1</quantityonhand>
    </item>
    <item>
      <barcode>9780859699396</barcode>
      <quantityonhand>2</quantityonhand>
    </item>
    <item>
      <barcode> 9780859699396</barcode>
      <quantityonhand>0</quantityonhand>
    </item>
</ stock>
```

errorMessage [string] – If error occurred it will contain an error message.

getImage

Input parameters:

imageFilename [string] – The name of the image, e.g. SIS-7158.jpg. The image must exist in the location specified in the imageLocation setting.

errorMessage [string] – Initial value will be empty.

Output parameters:

getImage () [base64Binary] – The image will be sent back in the form of a byte array.

errorMessage [string] – If error occurred it will contain an error message.

getItemBrand

Input parameters:

itemcode [string] – The barcode of the item.

errorMessage [string] – Initial value will be empty.

Output parameters:

getItemBrand() [string] – The item brand.

errorMessage [string] – If error occurred it will contain an error message.

getItemDescription

Input parameters:

itemcode [string] – The barcode of the item.

errorMessage [string] – Initial value will be empty.

Output parameters:

getItemDescription() [string] – The item description.

errorMessage [string] – If error occurred it will contain an error message.

getItemExtendedDescription

Input parameters:

itemcode [string] – The barcode of the item.

errorMessage [string] – Initial value will be empty.

Output parameters:

getItemExtendedDescription() [string] – The item extended description.

errorMessage [string] – If error occurred it will contain an error message.

getItemID

Input parameters:

itemcode [string] – The barcode of the item.

errorMessage [string] – Initial value will be empty.

Output parameters:

getItemID() [integer] – The item identifier.

errorMessage [string] – If error occurred it will contain an error message.

getItemImageFilename

Input parameters:

itemcode [string] – The barcode of the item.

errorMessage [string] – Initial value will be empty.

Output parameters:

getItemImageFilename () [string] – The file name of the image.

errorMessage [string] – If error occurred it will contain an error message.

getItemPrice

This is the till price of the item.

Input parameters:

itemcode [string] – The barcode of the item.

errorMessage [string] – Initial value will be empty.

Output parameters:

getItemPrice() [double] – The rms till price.

errorMessage [string] – If error occurred it will contain an error message.

getItemPriceA

Input parameters:

itemcode [string] – The barcode of the item.

errorMessage [string] – Initial value will be empty.

Output parameters:

getItemPriceA() [double] – The rms pricea.

errorMessage [string] – If error occurred it will contain an error message.

getItemPriceB

Input parameters:

itemcode [string] – The barcode of the item.

errorMessage [string] – Initial value will be empty.

Output parameters:

getItemPriceB() [double] – The rms priceb.

errorMessage [string] – If error occurred it will contain an error message.

getItemPriceC

Input parameters:

itemcode [string] – The barcode of the item.

errorMessage [string] – Initial value will be empty.

Output parameters:

getItemPriceB() [double] – The rms pricec.

errorMessage [string] – If error occurred it will contain an error message.

getItemSize

Input parameters:

itemcode [string] – The barcode of the item.

errorMessage [string] – Initial value will be empty.

Output parameters:

getItemSize() [string] – The item size.

errorMessage [string] – If error occurred it will contain an error message.

getItemStockQuantity

Input parameters:

itemcode [string] – The barcode of the item.

errorMessage [string] – Initial value will be empty.

Output parameters:

getItemStockQuantity() [integer] – The amount of available stock for an item. The committed value is subtracted, this is any items that have been received and are on an open back order.

errorMessage [string] – If error occurred it will contain an error message.

getOrderStatus

Input parameters:

referenceNumber [string] – The reference number of the order.

errorMessage [string] – Initial value will be empty.

Output parameters:

getOrderStatus () [string] – The order status:

Status	Description
Processing	Waiting to be processed, exists as a transaction on hold in RMS.
Dispatched	Has been put through the till as a sale.
Deleted	The order has been cancelled or never existed.

errorMessage [string] – If error occurred it will contain an error message.

placeXmlOrder

Creates a transaction on hold using the data supplied in an xml string.

Input parameters:

orderXml [string] – Sample data will look like:

```
<?xml version="1.0"?>
<order xmlns="rmsOrder">
  <referenceNumber>001</referenceNumber>
  <comment>web order comment</comment>
  <customerID>406</customerID>
  <transactionNumber></transactionNumber>
  <items>
    <item>
      <barcode>700596100909</barcode>
      <quantityOrdered>1</quantityOrdered>
      <price>2.05</price>
    </item>
    <item>
      <barcode>0782126003034</barcode>
      <quantityOrdered>2</quantityOrdered>
    </item>
  </items>
</order>
```

SOAP Request

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body>
<placeXmlOrder xmlns="http://emporiouk.com/RMSAPI">
<orderXml>&lt;?xml version="1.0"?&gt;
&lt;order xmlns="rmsOrder"&gt;
  &lt;referenceNumber&gt;001&lt;/referenceNumber&gt;
  &lt;comment&gt;web order comment&lt;/comment&gt;
  &lt;customerID&gt;406&lt;/customerID&gt;
```

```
  &lt;transactionNumber&gt;&lt;/transactionNumber&gt;
  &lt;items&gt;
    &lt;item&gt;
      &lt;barcode&gt;700596100909&lt;/barcode&gt;
      &lt;quantityOrdered&gt;1&lt;/quantityOrdered&gt;
      &lt;price&gt;2.05&lt;/price&gt;
    &lt;/item&gt;
    &lt;item&gt;
      &lt;barcode&gt;0782126003034&lt;/barcode&gt;
      &lt;quantityOrdered&gt;2&lt;/quantityOrdered&gt;
    &lt;/item&gt;
  &lt;/items&gt;
&lt;/order&gt;</orderXml>
<errorMessage />
</placeXmlOrder>
</soap:Body>
</soap:Envelope>
```

The xml is validated against the following schema:

XML Schema

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
targetNamespace="rmsOrder" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="order">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="referenceNumber">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:minLength value="3" />
              <xs:maxLength value="50" />
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
        <xs:element name="comment">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:minLength value="3" />
              <xs:maxLength value="50" />
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
        <xs:element name="customerID" minOccurs="0" >
          <xs:simpleType>
            <xs:restriction base="xs:int">
              <xs:minInclusive value="0" />
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
        <xs:element name="transactionNumber" type="xs:string" />
        <xs:element name="items">
          <xs:complexType>
            <xs:sequence>
              <xs:element maxOccurs="unbounded" name="item">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="barcode">
                      <xs:simpleType>
```

```
                        <xs:restriction base="xs:string">
                          <xs:minLength value="1" />
                          <xs:maxLength value="25" />
                        </xs:restriction>
                      </xs:simpleType>
                    </xs:element>
                    <xs:element name="quantityOrdered" type="xs:short" />
                    <xs:element name="price" type="xs:decimal" minOccurs="0" />
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

The *customerID* element is optional, but if specified will take precedence over the *webCustomerID* setting in the web.config file.

errorMessage [string] – Initial value will be empty.

Output parameters:

placeXmlOrder () [string] – Sample output:

```
<?xml version="1.0"?>
<order xmlns="rmsOrder">
  <referenceNumber>001</referenceNumber>
  <comment>web order comment</comment>
  <transactionNumber>6061</transactionNumber>
  <items>
    <item>
      <barcode>700596100909</barcode>
      <quantityOrdered>1</quantityOrdered>
      <price>2.05</price>
    </item>
    <item>
      <barcode>0782126003034</barcode>
      <quantityOrdered>2</quantityOrdered>
    </item>
  </items>
</order>
```

SOAP Response

```
<?xml version="1.0" encoding="utf-8"?><soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body>
<placeXmlOrderResponse xmlns="http://emporiouk.com/RMSAPI">
<placeXmlOrderResult>&lt;?xml version="1.0"?&gt;
```

```
&lt;order xmlns="rmsOrder"&gt;
  &lt;referenceNumber&gt;001&lt;/referenceNumber&gt;
  &lt;comment&gt;web order comment&lt;/comment&gt;
  &lt;customerID&gt;406&lt;/customerID&gt;
  &lt;transactionNumber&gt;6061&lt;/transactionNumber&gt;
  &lt;items&gt;
    &lt;item&gt;
      &lt;barcode&gt;700596100909&lt;/barcode&gt;
      &lt;quantityOrdered&gt;1&lt;/quantityOrdered&gt;
      &lt;price&gt;2.05&lt;/price&gt;
    &lt;/item&gt;
    &lt;item&gt;
      &lt;barcode&gt;0782126003034&lt;/barcode&gt;
      &lt;quantityOrdered&gt;2&lt;/quantityOrdered&gt;
    &lt;/item&gt;
  &lt;/items&gt;
&lt;/order&gt;</placeXmlOrderResult>
<errorMessage />
</placeXmlOrderResponse>
</soap:Body>
</soap:Envelope>
```

If order creation was successful, the *transactionNumber* element will contain the newly created transaction number.

errorMessage [string] – If error occurred it will contain an error message.

Each time this method is called the parameter values are logged in a database table called *Emporio_RmsApiLog*.

placeXmlOrderCommitted

Works in the same way as placeXmlOrder():

- The input and output parameters are the same.
- The xml schema is the same.

Except:

placeXmlOrderCommitted	**placeXmlOrder**
The order is created in the transaction table.	The order is created in the transaction on hold table.
If the *customerID* element isn't specified in the order xml, no customer will be specified.	If the *customerID* element isn't specified in the order xml, the *webCustomerID* value in the web.config will be used.

searchForCustomers

This function returns a list of customer identifiers.

Input parameters:

Regarding the filter criteria parameters (emailAddress, firstName and lastName):

- To exclude one or two of them from the criteria leave the value as blank. However one of them must have a value to search on.
- Wildcards are allowed and this is represented by the percentage symbol, e.g. To search for all customers that use hotmail, set the *emailAddress*=%@hotmail.com
- When the value for more that one parameter is specified, the OR operator is used, e.g. If *firstName*=John and *lastName*=Smith, customers whose firstname is John OR lastname is Smith will be retrieved.

Parameter Name	Type	Max length	Info
emailAddress	string	255	
firstName	string	30	
lastName	string	50	
errorMessage	string		Initial value will be empty

SOAP Request: Customers with yahoo email addresses.

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body>
<searchForCustomers xmlns="http://emporiouk.com/RMSAPI">
<emailAddress>%@yahoo%</emailAddress>
<firstName />
<lastName />
<errorMessage />
</searchForCustomers>
</soap:Body>
</soap:Envelope>
```

Output parameters:

searchForCustomers () [string] – This is a pipe delimited list of customer ids, e.g. 7175|8451|1087

errorMessage [string] – If error occurred it will contain an error message.

SOAP Response

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
```

```
<soap:Body>
<searchForCustomersResponse xmlns="http://emporiouk.com/RMSAPI">
<searchForCustomersResult>66|106|113|318|2210</searchForCustomersResult>
<errorMessage />
</searchForCustomersResponse>
</soap:Body>
</soap:Envelope>
```

Logging

Presently logging is available for the *placeXmlOrder* method.In RMS there is a custom report that shows each time this method was called along with information such as input and output parameters. The report is called Rms Api Method Logging: